Emergence, Evolution, Intelligence:

Hydroinformatics

\[ \frac{\partial x}{\partial t} + \frac{\partial y}{\partial t} + \frac{\partial z}{\partial t} = 0 \]
EMERGENCE, EVOLUTION, INTELLIGENCE; HYDROINFORMATICS
Emergence, Evolution, Intelligence; Hydroinformatics

A Study of Distributed and Decentralised Computing Using Intelligent Agents

DISSERTATION
Submitted in fulfilment of the requirements of the Board of Deans of the Delft University of Technology and the Academic Board of the International Institute for Infrastructural, Hydraulic and Environmental Engineering for the Degree of Doctor to be defended in public on Wednesday, 20 March 1996 at 19:00 h

by

VLADAN BABOVIC
B.Sc., M.Sc., Dip HE (Delft)

A.A.BALKEMA / ROTTERDAM / BROOKFIELD / 1996
to Višnja
Acknowledgements

I wish to thank the International Institute for Infrastructural, Hydraulic and Environmental Engineering (IHE) for supporting this work financially. In the course of my four years as a Research Fellow at IHE I have been fortunate to meet a large number of people who have in part influenced the views presented in this thesis. Indeed, this work draws upon the work of many colleagues and students as well as other distinguished scholars. Providing even an incomplete list of the persons to whom I am indebted proved to be rather impractical. Such list would go on for several pages and might even form a separate chapter! I am therefore left with a simple, but unconditionally sincere: thank you all!
## Contents

1. Prologue ........................................ 1

2. Agents .......................................... 11
   2.1. Framework .................................. 14
   2.2. Primitive Data Types ......................... 16
   2.3. General Agent Properties ..................... 21
   2.4. Formal Definitions .......................... 27

3. Intelligence .................................... 29
   3.1. Intelligence in Animals (including *Homo sapiens*) .. 29
   3.2. Intelligence in Machines ..................... 33
   3.3. Intelligence in Societies ..................... 44

4. Emergence ...................................... 51
   4.1. Phenomenological definitions ................. 51
   4.2. Formal definitions .......................... 55
5. Searching for Solutions

5.1. Graph theoretic notation
5.2. Weak and strong search algorithms
5.3. Non-informed search
5.4. Heuristic Search

6. Evolving Solutions

6.1. Properties of natural evolution
6.2. Naive evolution
6.3. Genetic representations
6.4. Fitness landscape
6.5. Genetic Algorithms
6.6. Evolution Strategies
6.7. Evolutionary Programming
6.8. Learning Classifier Systems
6.9. Genetic Programming
6.10. Problems
6.11. Some applications of evolutionary algorithms
6.12. A machine learning algorithm based on evolutionary principles
6.13. Model induction from data

7. Emerging Solutions

7.1. Current approaches to ecological modelling
7.2. Intelligent agent approach to modelling
7.3. Description of the intelligent agent’s structure
7.4. The agent-orientated superstructure
7.5. The shift in paradigm
7.6. A related approach - cellular automata
7.7. Modelling ecosystems with intelligent agents
7.8. Food webs at the edge of chaos

8. Hydroinformatics as emergent phenomenon

References

References
The computer-controlled operating environments of such facilities as automated factories, nuclear power plants, telecommunication centres and space stations are continually becoming more complex. The situation is similar, if not even more apparent and urgent, in the case of water. Water is not only mankind's most valuable natural resource, but one which is in increasingly limited supply. 'Fresh water is the vital natural resource which supports all environmental activities, that is, the natural economy, and all human socio-economic activities, that is, the artificial economy. Therefore, life on this planet is essentially an aquaculture living in a hydro-economy.' (Matthews and Grabs, 1994). The pressure for a sustainable control and exploration of water and thus for the peaceful coexistence of human- and hydro-economies, is not only a human, socio-economic pressure, but it is the question of life and death!

Fresh, clean water is an imperative for the survival of any society, be it a village in Africa or a highly industrialised European city. Although more than 70% of the planet's surface is covered by water, less than 3% of this resource is composed of fresh water. However, the demand upon this finite supply continues to grow with the increase in the Earth's population
and the growing demands of modern industry, domestic hygiene and agriculture.\footnote{Before the agricultural revolution, some 10,000 years ago, it is estimated that the earth’s population comprised of about 100 million people. By the time of Christ, the total population grew to 250 million. This number doubled, to 500 million, by the times of first Crusaders. By the time of United States’ War of Independence, in the late eighteen century, the total world’s population was roughly 1,000 million. After the Second World War, the number was about 2,500 million, and presently it is 5,500 million. It is estimated that in coming 30-40 years the total population will grow to 10,000 million!} But if man is to avoid an environmental and social cataclysm during the next millennium, it is essential that he learns to control the demands that he places upon this precious natural resource. One way of achieving a sustainable coexistence between man and his aquatic environment is by employing information technology, although to do this effectively demands a thorough knowledge of both this emerging science and its potential applications. While the information revolution of the last 30 years has led to the general availability of sophisticated computers with ever-expanding capabilities, it has also witnessed an increasing complexity in terms of the computational ability to store, retrieve and manipulate vast information flows.

Hydroinformatics (Abbott, 1991) - the nascent technology concerned with the flow of information related to the flow of fluids and all that they convey - is probably the best possible answer yet proposed to the problem of the control of the waters, the very arteries and veins of the biosphere.

Hydroinformatics has a strong ancestry in the computational sciences, and then primarily in computational hydraulics (Abbott, 1979; Abbott and Minns, 1995), which was developed with the objective of dealing with the foundations of numerical modelling. Computational hydraulics utilises numerical modelling techniques to describe physical systems with sets of numbers and to simulate the physical laws acting upon these systems with sets of operations on these numbers. The introduction of this technology some 30 years ago necessitated a thorough reformulation of classical hydraulics to accommodate the new possibilities represented by the discrete, sequential and recursive processes of digital computation.

Computational models based on computational hydraulics have undergone a process of evolutionary change and adaptation to advances in computational machinery and have induced evolutionary changes in the engineering and managerial activities which used these models. The first generation of numerical models were quite simple, though. The computations were performed in a way similar to the existing ‘hand calculations’. Exactly the same, analytically-obtained solutions of the physical problems were coded into a computer, and exactly the same results were obtained through application of these codes, albeit faster and with higher
numerical precision. Thus, the computer was treated as some sort of a ‘super slide rule’. However, the developments within computational hydraulics necessarily changed the universe of discourse within which it was placed, necessitating that practising engineers ‘rethink the physics’ in order that an efficient use should be made of computer facilities.

The second generation of models were correspondingly characterised by the utilisation of a number of machine-friendly techniques: finite-difference methods, finite-elements methods, boundary-elements methods, fast Fourier transforms, Marker and Cell, etc. Thus, the formulation of physical problems was directed towards formalisms that were suitable for digital computer applications, thus enhancing the efficiency of the computations. These, second-generation models, however, still had very few customers, for the computational machinery of these times was almost exclusively available at a very few institutions around the world and the corresponding hydrodynamic models that were generated were custom-made, suited to solve only a particular problem, namely the one for which the model was instantiated.

Driven by further market demands, pressing for ever more powerful computational models, shorter delivery times, lower costs and other enhancements, the third generation of models emerged. These are perhaps best understood as a collections of ‘pre-fabricated’ sub-routines that can be readily and even automatically assembled to describe numerically any physical problem of a wide generic type. However, such models could still only be effectively operated by a relatively few, highly-qualified computational hydraulics specialists. A further proliferation of computational machinery and reduction in costs, and in particular the introduction of IBM’s PC/AT (based on the INTEL 80286 micro-processor and its corresponding arithmetic co-processor) that made computers available almost everywhere, greatly expanded the potential market for computational hydraulic models. However, the third generation models were far too demanding in computational-hydraulic expertise for the world of non-specialists, who could therefore not utilise these models effectively.

Thus the qualitatively new fourth generation of models appeared on the market. These models are typically user-friendly, menu-driven environments that are fault tolerant and provide default values for most of their control variables. The typical state-of-the-art commercial software product of today belongs to this, fourth generation. These models have now become standard tools in the design of many, if not most, major engineering enterprises and are used as fundamental sources of information for decision-making.

However, while traditional models of computational hydraulics provide a valuable assistance in the analysis of major engineering projects, they have one notable drawback: they can only be understood by those with sufficient experience to interpret the vast amount of numerical data that they produce. In a society where decision-makers are seldom specialists, it is vital that such assistance be made available to as many people as possible so that it must be provided in a form that can be readily understood.

In order to operate effectively, a hydroinformatics system must commonly contain numerical models to quantify the movements and changes within a body of water; it must use graphical user interfaces to present the results of computations in forms which are understandable to a wide audience; it must assist in the interpretation of these results via expert-advice systems;
and it must store this information in data- and knowledge-bases. The size and complexity of
such an undertaking is well illustrated by the computations of the motions of a moored ship
in a wave field in a harbour.

Only one aspect of these difficulties is that of dealing with the shear quantity of numbers
produced by numerical models. For example, the output obtained from a numerical model of
the type illustrated in Figure 1, may well consist of more than a billion seven-decimal-digit
numbers. Bearing in mind that only 15,000 such numbers can be printed on a square meter
of paper, the output for this particular simulation alone would cover some 70,000 square
meters. This is, approximately the area of a 10 soccer fields! On the other hand, these same
results can be presented graphically, using colours and shading, to provide concise, highly
realistic, almost 'photographic' effects - and then can be easily interpreted and understood,
and to some extent even by laymen. However, the management of large quantities of data,
as exemplified in this example, is only one of the concerns that arise within hydroinformatics
frameworks.

Figure 1 A very large number of numbers in a form understandable even to layman (illustration
courtesy Danish Hydraulic Institute)

In addition to the simulations producing water quantity and quality results, the information
necessary to describe and assess the state of any given body of water must also include a
plethora of descriptions of social, legal and environmental factors. In this context, the typical
information to be incorporated into a hydroinformatics study must include such variables as
international and national laws, local bye-laws - either temporary or permanent - and any
applicable physical, chemical and biological parameters. Added to this, flows of water,
sediment, chemicals and other waste-borne substances must be calculated and measured, and
the sites and water quality parameters of the area's water-users identified and incorporated
into the system.

The eventual physical, social and environmental impacts resulting from any action upon the
water body must also be determined before the execution of a project. An important feature
of a hydroinformatics system is that it allows the use of those numerical simulations which are subject to constraints expressed in natural language (such as applicable legislation, contracts, agreements, etc.). Lastly, the locations and production rates of heat, chemical and biological pollution must be added to the formulation of problem, as well as the presence of any control elements in the area, such as pumps, retention basins and treatment plants.

Up to this point, hydroinformatics systems have been characterised as platforms for unifying various components; hydrodynamic simulation engines, water quality modelling systems, sediment transport and morphological calculation routines, along with various impact assessment tools, legislative constraints, etc. Such hydroinformatics systems, even if composed of only a few components, present some quite serious dangers. Such intricate systems are capable of performing a vast range of calculations, and producing very detailed outputs... that are difficult or even impossible to interpret without a number of different kinds of specialist knowledge. The purpose of computing is insight, not numbers, but it is an everyday experience that the usual output of computing consists of numbers, not insight. Should hydroinformatics succeed in fulfilling the expectations that are invested in it, it must provide means for interpreting the results obtained when using various computational engines. Otherwise, it is doomed to remain as a definite syntax with indefinite semantics.

Producing an acceptably accurate prediction often requires a wealth of knowledge, much of which can only be obtained by studying previous experiences under similar circumstances. At the same time, a hydroinformatics system is likely to be operated by a broad range of users, ranging from novices to experts, and it is most likely that some of the operators will be hampered by the average intelligibility\(^2\) of their universe of discourse. In this sense, the problem becomes one of how to transfer and translate a large body of information, knowledge and experience to (Abbott, 1986):

"... a world that knows little or nothing of the problems that wrack the computational-hydraulics experts. It does not understand their jargon and it is largely uncomprehending of their enthusiasms and their despairs. The problem is then of expressing the knowledge and capability of experts in terms of another and more average intelligibility."

Hydroinformatics facilitates this assessment process by encapsulating expert knowledge and experience and by making this information available to hydro-scientists and engineers, thereby raising the level of their professional performance. Thus, quite obviously, hydroinformatics has an equally important ancestry in Artificial Intelligence (AI) and especially in that part that deals with the modelling of intelligent behaviour and the encapsulation of professional knowledge, including expertise. The hydroinformatics system, with its AI components, can be seen as a realisation of a fifth generation working environment, and as such it is qualitatively different from all of its computational hydraulics ancestors.

Hydroinformatics makes it possible to analyse environmental problems in much more of their

\(^2\) Already in 1927, this notion was introduced by Martin Heideger, the 'philosopher of technology of our century'; see also Abbott (1991, p.95)
natural and social-economic completeness. It also enables the knowledge, and especially the expertise, of a few to be made available to many. Users are immediately and unambiguously confronted with the consequences of their actions upon the natural environment and, because various scenarios can be simulated, they are able to determine both the effectiveness or otherwise of any proposed activity. Thus the potential environmental damage that follows from a certain intervention in nature can be more accurately predicted, as can the likely consequences of a project aimed at rejuvenating an already damaged ecosystem.

Such a paradigm not only permits its users to apply a computational model of interest, but also to utilise open architectures to integrate various scientific disciplines. Such a system can make a valuable contribution to ‘consensus building’ across an entire scientific spectrum, and indeed it is the only approach that is capable of demystifying the underlying science and of making accumulated knowledge available to a larger audience. Its value can therefore be measured in both scientific and social terms. While knowledge is undoubtedly power, the power within our society frequently lies with those who have little specialist knowledge and are thus ill-equipped to make far-reaching decisions. In the past, the aquatic environment has suffered as a result of this imbalance: an intelligent solution constructed within a hydroinformatics’ paradigm should help to ensure that both man and his environment fare better in the future.

Recent advances in hydroinformatics technology, often associated with large (multi-billion-dollar) projects (Eastern Scheldt storm surge barrier in the Netherlands, Great Belt and the Sound traffic connections in Scandinavia, the Venice protection works in Italy, the various Flood-Action Plans in Bangladesh, etc.), have made new tools and methodologies available to engineers and scientists to assist in the monitoring and control of hydraulic and environmental systems, but they have inevitably raised a new range of difficulties that have to be addressed by the designers of hydroinformatics systems.

The complexity of a complete hydroinformatics system is, in most cases, much greater than any single one of its users (and, for that matter even any single designer) can comprehend. Thus, as graphically illustrated in Figure 2, an average user of a hydroinformatics system is not a single person, but rather a group of persons, and these are commonly experts, each of whom operates within his or her own area of expertise.

It must be admitted, though, that the practice of hydroinformatics clearly also provides unprecedented opportunities for introducing all manner of biases, deceptions, falsifications, and other kinds of misrepresentations of the nature and the behaviour of aquatic environments. Our hydroinformatics systems must interact not only with individual persons and groups of persons, but with highly heterogenous collections of individuals and groups, each with an own set of interests and a corresponding set of intentions. Moreover, as Abbott (1994b) points out, although many of these interests and intentions may be formulated explicitly, others will only be expressed implicitly, as in certain assumptions and conjectures, while others again may be concealed within ‘hidden agendas’ and take other more-or-less opaque forms. These problems are intrinsic to the new generation of management support systems that are an integral part of the set of modern hydroinformatics systems in general.

The view of system-design as an ethical practice is not foreign to the computer-scientific
community, and has far-reaching consequences in practice for the future of hydroinformatics. It points to a unifying influence on a wide range of technologies on the one side, and to new opportunities to tackle ever more significant and urgent problems, such as those or resource-management support, on the other side. However, the realisation of a new paradigm will in its turn require a very thorough and far-reaching re-evaluation of the present theory, drawing much more heavily than has been done so far upon the pioneering work carried out in other fields.

Figure 2 A schematisation of a ‘true’ hydroinformatics system (adapted from Price et al, 1994)

With the introduction of a hydroinformatics paradigm, more fundamental changes are taking place: changes which are dramatically affecting traditional planning, design and decision-making processes involved in water-related science and technology. Hydroinformatics changes the comprehension and helps the understanding of our relations to our environment. Bearing
all this in mind, hydroinformatics indeed represents a ‘Copernican revolution in hydraulics’ (Abbott, 1994a). However, despite the new ground already broken by hydroinformatics, present systems are far from fulfilling their full potential.

The question concerning the architectural features of a hydroinformatics system obviously appears as a central issue. What are the necessary attributes of such a system? This, in short, constitutes a principal pragmatic motivation for this work. The engineering applications necessitate a synthesis of diverse knowledge sources. In particular, they call for efficient, automated, reliable, and repeatable methods for constructing useful systems. This dissertation is devoted to a further development of hydroinformatics in the broadest possible terms. The subject of the work partially falls within the area of integrated agent-architecture design. However, this dissertation is specifically directed to hydroinformatics. This means that the greater part of the work presented here is clearly directed towards specific hydroinformatics’ needs and requirements, thus adapting the knowledge developed elsewhere and merging this with the more domain-dependent aspects.

In the following Chapter 2, the notion of computational agency, which provides societal perspective on computation, is introduced and an argument is advanced in favour of corresponding models. Natural intelligence, the realisation of which within artificial media represents the ultimate goal of computational science, and which is traditionally modelled as a centralised process, is depicted as a decentralised and distributed process in Chapter 3. By these means, a more nature- and socially-orientated approach to modelling intelligent behaviour is elaborated and analysed. Further, in Chapter 4, the most interesting properties of such a societal, or distributed paradigm are identified as ‘emergent’. It is argued that hydroinformatics itself can be best perceived as a genesis; that is as a set of emergent properties of its interacting components. The rest of this dissertation is devoted to a sequence of investigations of multi-agent conglomerations within which interactions emerge in specific, hydroinformatics’ context.

Chapters 5 and 6 are more specifically orientated towards methodologies for individual agent design. Chapter 5 describes some more traditional search techniques, whereas Chapter 6 addresses the issues related to computational models of evolution, exemplified by a rather broad category of Evolutionary Algorithms (EAs). It is argued that the interaction of a relatively few, simple agents among each other and with their common environment results in the induction of models that are capable of intelligent performance. It is even argued that as this technology comes to maturity over the coming decade, practically all our programming work will be done with the assistance of such processes of evolution of artificial agents. It is argued even further that, because of the interaction of evolving agents with their environments, they incorporate the main properties of this environment in the models which they generate, grounding them in their environments and thus preventing us from building-in our own pre-conceptions, our own, often one-sided perceptions, and our own, and again often restricted systems of values. Thus, it is asserted that only agents that are designed in such a way that they are properly grounded in their physical and social environment can really be considered ethical, and that these constitute the only category truly serving the many and varied users of hydroinformatics systems.

Chapter 7, directs the focus of interest to somewhat more complex agents, the modelling of
which has been done traditionally using ordinary and partial differential equation, or in a
manner that is described throughout this work as top-down.

Finally, the work concludes with a discussion of the architectural issues of an ‘ideal’
hydroinformatics system.

Many ideas intersect and interact during this work. Some of them may appear ungrounded and
possibly flowing too freely. Although an attempt has been made to present the line of
reasoning in (one or the other) ‘logical’ way, it is by no means claimed that this has
succeeded. In the manner which has been so brilliantly formulated by Derrida (e.g. 1967,
1968//1973), a considerable part of what is presented here as ‘grammatical’ arises more from
the rhetoric of the subject, and less from any explicit underlying logic. We have been unable
to construct this work using a ‘purely-logical grammar’ in the sense of Husserl, and we
suspect, following Derrida, that this may indeed not be possible in principle. The evolutionary
processes interact with the process of emergence; and emergence interacts with agent design;
and ecology itself interacts with hydroinformatics; and intelligence interacts with ethics;
altogether, this thesis itself can be regarded as an emergent ‘property’ of its interacting
chapters, and these as emergent properties of their sections and these again of their words and
their other signs (e.g. drawings). As will be described later, the observer, as the reader of this
text, is an essential element within the paradigm of emergence. This implies that a particular
observer is able to detect particular emergent phenomena that other observers may not
necessarily distinguish. One consequence of this is that some readers might not ‘catch-on’ at
all. The only way, and the way that has sometimes been adopted here, is to describe similar
approaches in slightly different ways so as to help this material to ‘click’ in the minds of
more of its readers.
2. Agents

... the law ought to trust people with the care of their own interest, as in their local situations they must generally be able to judge better of it than the legislator can do.


Hydroinformatics, as a discipline, is *embedded in a complex environment*. This is particularly apparent in real-time control and monitoring systems that are ‘connected’ to the real, physical world through a collection of *sensors* measuring various physical quantities, such as discharges, water levels, states of controllers (pumps, gates, syphons and valves) and rainfall distributions and intensities, and *effectors* that transfer orders from the (artificial or human) decision maker to the controllers.

As this complexity grows, so it becomes increasingly difficult to control such environments with traditional centralised management and scheduling policies that are both *robust* in the face of unexpected events and *flexible* in dealing with operational and environmental changes that may occur over time. Thus, a first major problem in making effective use of computers within a hydroinformatics paradigm is that of dealing with this ‘physical’ kind of *complexity*. As demand grows and we have to design programs of increasing functionality, we find ourselves already overwhelmed, even at this level, by the ever greater intricacy of the resulting computer systems.

The problems introduced by this kind of complexity are exacerbated by the increasing need to accommodate a number of different kinds of end users of hydroinformatics’ systems, whereby one and the same system may have to serve several kinds of managers, design
agents, plant operators, environmental regulators and others besides. We are then simultaneously confronted with problems of handling an increasing level of social complexity.

These problems become reflected as fundamental questions on the limits of software complexity. Most present-day software is human-engineered, and thus bounded by the intellectual capacities of its designers. But the requirements of practice now lead us to ask whether software could be provided with certain capacities that would allow it to adapt to its environment and so 'learn' from its environment. Or, could computer programs be built that function analogously to human organisations, for example? These human organisations are often able to deal quite successfully with problems of extreme complexity, asynchronous operation and imperfect knowledge - which often constitute insurmountable problems for conventionally-designed software agents, that are managed from one, single, centralised location in memory.

---

**Figure 3** Current AI research falls mostly into three clusters, which can be contrasted according to the degree of complexity of the environments with which they deal and the number of interacting agents that they employ. Hydroinformatics, however, imposes specific requirements that need to be addressed specifically.

One solution to this problem of dealing with both complexity of physical environment and complexity of resulting software acting in this environment, and one which has a growing appeal, is to distribute - along dimensions such as *function* and *space* - the control and
scheduling of operations to a number of intelligent, task-satisfying computational agents.\(^3\) Real-world hydroinformatics domains are likely to be increasingly populated with multiple agents, each pursuing a number of tasks. Since agents are likely to have only incomplete knowledge about their worlds (consisting of both their physical and social-institution environments and their fellow-agents) and will compete for limited and shared resources (like CPU time), it is inevitable that, over time, some of their goals will conflict. Attempts to construct complex, large-scale hydroinformatics systems in which all envisaged modes of collaboration and conflicts among agents are foreseen and catered-for in advance (and so, stored at some centralised location), are likely to be too expensive, too complex, or perhaps even physically impossible to realise given the effort and uncertainty that would be involved in accounting for all possible futures of equipment, design, management and operational situations.

Following the outline just presented in this chapter, a novel approach to modelling is considered and formalised. This approach is, we believe, the most suitable for the requirements imposed by hydroinformatics practice. The proposed paradigm interacts and intersects with the work of others, and is founded on a ‘societal view of computation, in which multiple agents interact with one another’ (Shoham, 1993). In this context, an agent is itself a computer system that is situated in a (usually) dynamic and complex environment, and attempts to fulfil a set of objectives which are assigned to it or which are chosen by the agent itself. Autonomous agents act entirely on their own accord, whereas adaptive agents are capable of acclimatising their actions while progressing through successions of attempts to satisfy the goals that they have at hand. Thus, basically, an adaptive agent adapts its behavioural repertoire and improves its actions based on a feedback from its environment that informs it about its success in achieving these goals. Ashby (1952) proposed a characterisation of adaptive behaviour as the one which maintains an organism’s essential variables within their threshold limits.\(^4\)

---

\(^3\) The notion of an agent is central to this work. This concept will be introduced in more detail later in the text. For the time being, it will suffice to equate an agent with a task-orientated computational process capable of robust and flexible interaction with an environment which may (and usually does) include fellow-agents.

\(^4\) Ashby defined the essential variables as a collection of closely and dynamically interrelated factors, so that marked changes in any one leads, sooner or later, to observable changes in the others. Ashby (1952) writes:

Thus, if we find a rat in which the pulse-rate has dropped to zero, we can predict that the respiration rate will soon become zero, that the body temperature will soon fall to room temperature, and that the number of bacteria in the tissues will soon rise from zero to a very high number. These important and closely linked variables will be referred to as the essential variables of the organism.

Thus, survival, whether for natural or for artificial agents, can be defined in terms of states encompassed by an envelope of behaviour whereby no essential variable can be situated outside this envelope.
A viable agent must be autonomous. Smithers (see Steels, 1993) captures the essence of autonomy as follows:

The central idea in the concept of autonomy is identified in the etymology of the terms: *autos* (self) and *nomos* (rule or law). It was first applied to the Greek city states whose citizens made their own laws, as opposed to living according to those of an external governing power. It is useful to contrast autonomy with the concept of automatic systems. The meaning of *automatic* comes from the etymology of the term *cybernetic*, which derives from the Greek for *self-steering*. In other words, automatic systems are self-regulating, but they do not make the laws that their regulatory activities seek to satisfy. These are given to them, or built into them. They steer themselves along a given path, correcting and compensating for the effects of external perturbations and disturbances as they go. Autonomous systems, on the other hand, are systems that develop, for themselves, the laws and strategies according to which they regulate their behaviour: they are *self-governing* as well as self-regulating. They determine the paths they follow as well as steer along them.

Traditional AI has, so far, shown little interest in such a *societal* approach to modelling. However, Huberman (1988) already refers to such computational environments as *computational ecologies*, whereas Stefik (1988) argues that such an approach will play the central role in the further development of the computational sciences, and indeed predicts that it will provide a new *knowledge medium*.

This society-orientated concept of modelling derives from a symbolic-interactionist sociology and certain other studies of modern science. The basic ideas of such interactionism place an emphasis on interactions as the primary units of analysis, and not individuals-as-such. *Individuals*, in turn, are dynamic and evolving agents with many components; they are not stable, structured, unitary entities. The context-dependent, continuously-evolving individuals are *interactively* shaped in time through the processes of dynamic selection and changing interpretation (as symbols) of stimuli. Actions of individuals are guided (but not determined) by *perspectives*, which are again learned through mutual communication. *Society* comprises individuals in *processes of patterned interactions*, embedded in, and always (re-)creating perspectives.

Let us, in continuation, first introduce an agent more fully, even if still informally, by viewing the task that an agent should be designed to fulfil from a number of different perspectives.

### 2.1. Framework (Anatomy)

It is often argued that the term ‘agent’ as used within computational science is as loosely and ambiguously defined as it is the term artificial intelligence (AI) itself. Since, however, the notion of an (intelligent) agent is central to this work, it ought to be characterised as early as possible. The general framework for distributed and concurrent computation distinguishes the agents from the environment in which they are situated. The environment may posses some non-deterministic characteristics and the time flow is expressed in terms of discrete intervals.

Within the agent itself, the controller and the mechanism are further distinguished. The
mechanism consists of the sensors and effectors that allow the agent to interact with its environment. The controller is, at least conceptually, the component that accepts inputs from the sensors and controls the effectors. The controller may maintain some state (corresponding to a perceived application of memory). The details of both the nature of the environment and the structure and implementation of the agent have been abstracted away.

![Diagram of Agent System Overview](Image)

**Figure 4 Agent System Overview**

The present model and agent architecture is intended to be applied to systems as diverse as autonomous robots\(^5\), softbots\(^6\), knowbots\(^7\) and homeostats\(^8\). The agent, as presently

---

\(^5\) The term *robotics*, in fact, first appeared in the 1942 science fiction story 'Runaround' by Isaac Asimov. The term itself derives from the Czech word for worker and had already appeared in the English language in 1923 via a translation of Karel Čapek’s play ‘R.U.R.’ (Rossum’s Universal Robots). The widespread performance and reading of the play helped replace the term in vogue at the time - *automaton* - for the now more commonly used term *robot*.

\(^6\) Softbot (software robot) is a term coined by Oren Etzioni and Richard Segal that refers to intelligent agents that interact with a real-world software environment by issuing commands and interpreting the environment’s feedback.

\(^7\) Knowbots is the term introduced by Kahn and Cerf, and is fundamentally similar to the softbots, but with a more prominent knowledge component.
conceived, may have an actual physical presence or it may just exist within the memory of a computer. The control can be implemented in software as well as in hardware.

Following this line of reasoning, and maintaining the task to be fulfilled as an ever present objective, the general agent system and the deliberative agent system (Goodwin, 1993) can be further defined.

2.2. Primitive Data Types

An agent is an entity created to perform some task or set of tasks. Consequently, any property of an agent must be defined in terms of both the task that it presents to itself and the environment in which this task is to be performed. Thus, any formal definition of an agent must include a framework for describing:

(i) an environment,
(ii) a task,
(iii) the agent, as and by itself.

2.2.1. Environment

Every agent operates in an environment. In order to define an environment, one must characterise the valid states of this environment and how these change over time. In general, the state of an environment is dynamic, changing 'of its own accord', so to speak, and in response to interactions with other agents. In a 'deterministic' environment, knowing the current state, the interaction with other agents and the laws of nature appertaining allow the immediate environment's future to be predicted with some degree of accuracy. Obvious examples of deterministic environments from the world of hydroinformatics are: hydrodynamic models, advection-diffusion models, eutrophication models, etc. These models are perhaps best perceived as platforms which can be used to provide a foundation for an agent-orientated mode of computation.

In non-deterministic environments, the best that can be done is to predict a set of possible future states. Any model of the environment must be able to account for both deterministic and non-deterministic state changes.9

---

8 Homeostat is the term introduced Ashby, who used the term to portray the family of machines that can maintain a collection of essential variables within certain, sustainable viable limits.

9 We could adopt the pseudo-Newtonian view that the world would be deterministic if we could only know its state in infinite detail. Any apparent non-determinism is then only the result of our limited information about the current state, compounding this with our limited ability to compute future consequences. This view is further, of course, incompatible with the thermodynamic foundations of information theory.

Even neglecting computational limits and entropy considerations, however the environment may still be non-deterministic if we allow for agents with free will. Either we can distinguish such other agents from the environment and model them as non-deterministic entities or we must leave them as part of
Within this framework, in order to characterise environmental changes we need only to be able to predict the next state given the current state and the current interaction with the agent. This can be done if the so-called Markov property holds, whereby the immediate state description of the environment captures all the relevant features. Even with the Markov property in place, however, it is not usually possible to predict the next state for a non-deterministic environment. The best that can be done is to predict the set of possible states and the probability of each one. When considering how to model non-determinism, there are at least two major issues to keep in mind (Goodwin, 1993). The first of these is that the model should allow the performance of agents to be compared under the same operational conditions. And, secondly, the model should provide for certain probability distributions over successor states.

2.2.2. Task

A task is that which the agent is supposed to achieve within an environment. A task specification is a description of a task in some formal language.

All task specifications presuppose an environment, a set of possible initial conditions and a method for evaluating the agent's performance.

The simplest form of task evaluation is binary: either the task is accomplished or it is not. Such an evaluation can be encoded by listing all the ways of accomplishing the task successfully. An agent is said to be successful if its behaviour, in response to conditions in the environment, corresponds to the one of its methods that actually accomplishes the task.¹⁰

2.2.3. Agent

An agent is any entity created to accomplish a task. We distinguish the agent from the environment in order to enable us to substitute one agent for another and so to compare their relative performances. Within the agent, we similarly distinguish the mechanisms from the controller, in order to allow comparisons to be made between different mechanisms and different controllers.

Not every agent needs to distinguish among all possible states in its environment, nor does it always need a capacity to perform all possible actions in an environment. Following Genesereth and Nilsson (1986), all possible states in an agent's environment \( S \) are accordingly partitioned into the \( \mathcal{F} \) disjoint subsets that an agent is able to distinguish as 'somehow different'. The function that performs this discrimination is referred to as the sensory function. Thus, more formally:

\[
\text{sense}: S \rightarrow \mathcal{F}
\]

the environment and introduce the non-determinism that they introduce in the environment.

¹⁰ Note, however, that we are not suggesting that any implementation would use such a representation. This is only meant to serve as a useful way of conceptualising this process.
A similar approach is applied to effectors. A collection of all possible actions is denoted as $\mathcal{E}$. The **effectory function** is then defined as:

$$\text{effect}: \mathcal{E} \times S \rightarrow S$$

In order to define the activity of an agent, an **action** function is introduced that maps environmental state partitions into actions to be performed by an agent. The action function then becomes:

$$\text{action}: \mathcal{F} \rightarrow \mathcal{E}$$

### 2.2.3.1. Mechanism

An agent’s mechanism determines how the agent can perceive and affect its environment. It provides the agent’s only means of interacting with its environment. The mechanism’s sensors provide information about the world and the mechanism’s effectors provide a means for changing the world. The distinction between sensors and effectors has more to do with the flow of information and agent influence than it has with any physical arrangement.

### 2.2.3.2. Controller

Very briefly, the internals of an agent can be modelled as a finite state automaton (FSA) or artificial neural network (ANN) that consists of a control function and a memory. The controller function is a mapping from a perceptory reading and a current state to a command and a next state. The exact nature of the control mapping and the information encoded in the state are determined by the agent’s architecture and implementation.

A complete agent consists of a mechanism, a controller and an initial state. The controller and the mechanism must be modelled in terms of perceptory readings and commands; that is, the controller must be able to accept the perceptory readings generated by the mechanism and the mechanism must be able to handle the commands issued by the controller. The initial state encodes any explicit, a priori knowledge that the designer has given to the agent. Different types of agents are defined as specialisations of this general agent model.

So far, very little has been said about the details of an agent’s internal structure. Rather, the only interface between the agent and its outer environment to which reference has been made is that of its mechanism, and its supporting power of cognition, such as may be simplified to a finite state automaton or an artificial neural network allowing a rather one-directional flow of information. Such a structure is sufficient for treating a large class of problems in which a large collection of simple (reactive and reflexive) interactions occur. Indeed, in some cases it may even be the best structure for this purpose. Ecological simulations presented later in this thesis will support this statement.

However, for an organisation of high complexity, such as a full-blown hydroinformatics system, such a simple internal composition of an individual agent will not suffice. Thus, closely following Hayes-Roth (1995), we shall next introduce a somewhat more competent kind of individual agent.
Such an architecture of an individual agent organises perception, action and cognition sub-systems into a hierarchy. In such a system, the role of the perceptory subsystem is not only to perceive data, but also to filter the noise and to extract and abstract the most relevant information before it is forwarded to other sub-systems.

As indicated in Figure 6, perceptory operators can, in principle and in certain specific situations, short-cut the cognitive sub-system and effect the action sub-system either directly or through perception-action coordination processes, thus reducing their operation to one of an action of an FSA or ANN as indicated earlier, in sub-section 2.2.3.2. Through the invocation of the action sub-system, an agent can affect its environment and make a move towards the achievement of its goals. The cognition sub-system provides perception, data interpretation, problem solving and plan making, and it is also responsible for the elaboration of the agent’s overall perceptual strategies and actions. All these cognitive processes communicate by passing messages and in principle they operate concurrently and asynchronously.

Perception-action short-cuts can greatly accelerate the decision-making process because, in principle, they proceed at least one order of magnitude faster than does the entire iteration of the cognitive cycle.

Figure 5 Coordination of Cognition and Perception (adapted from Hayes-Roth, 1995)
The cognition sub-system, which is the most substantial and most complex component in this architecture can be realised in many different ways. A blackboard architecture (Hayes-Roth, 1985) was one obvious possibility in this work because it is very modular and its application had been tested earlier (Babovic, 1991; Amdisen et al., 1994). However, several features of a more general cognitive sub-system should be introduced. Following (Hayes-Roth, 1995) they are: (a) perceptual inputs and internal reasoning operations produce changes to a global memory; (b) each such event triggers a number of possible reasoning operations; (c) possible operations are scheduled for execution based on active control plans; (d) control plans are themselves constructed and modified by reasoning operations; (e) possible actions and control
plans are represented in a language that supports a semantics of interpretation of action-to-plan processes.

2.3. General Agent Properties

Mitchell (1990) provided the set of characteristics that he believed to be necessary for a successful agent (see also Goodwin, 1993). Following a similar approach, this section defines five general agent ‘properties’, namely those that are successful, capable, perceptive, reactive and reflexive. These properties are relevant to any type of agent performing a task with a binary evaluation function.

**Successful:** An agent is successful to the extent that it accomplishes the specified task in the given environment.

**Capable:** An agent is capable if it possesses the effectors needed to accomplish the task.

**Perceptive:** An agent is perceptive if it can distinguish salient characteristics of the world that would allow it to use its effectors to achieve the task.

**Reactive:** An agent is reactive if it is able to respond sufficiently quickly to events in the world to allow it to be successful.

**Reflexive:** An agent is reflexive if it behaves in a stimulus responsive fashion.

*Figure 7 General Agent Properties*

The relationship between three of these properties is depicted in Figure 8. An agent is capable if its effectors are able to accomplish the set task. A perceptive agent also possess the sensors needed to determine how to operate its effectors for achieving the set task. A successful agent is a perceptive agent with the task-appropriate controller.

There has been wide disagreement as to what the term reactive means when applied to an agent. The American Heritage Dictionary defines ‘reactive’ to be ‘tending to be responsive or to react to a stimulus’. In AI, a common definition of ‘reactive’ is *responding quickly and appropriately to changes in the environment*. This will be satisfactory within the present context only to the extent that we accommodate other agents within ‘the environment’.
When considering the achievement of a binary task, the agent must respond sufficiently quickly and appropriately to changes in its environment if it is to realise the task. An agent that is successful is then, almost by definition, sufficiently reactive. The definition of reactive becomes more interesting, however, when relative task achievements are considered later.

### 2.3.1. Reflexive agent

A term that is often confused with 'reactive' is 'reflexive'. An agent is said to be *reflexive* if it responds only to an immediate stimulus. Such agents can also be called 'stimulus-responsive' agents.

Reflexive agents do not need to maintain any memory: the history of the agent need play no part in determining its actions. Genesereth and Nilsson (1986) refer to this kind of agent as a *tropistic*\(^\text{11}\) agent. In Section 2.2.3, which introduced something of the agent’s anatomy, and

---

\(^{11}\) In this context, a tropism is a tendency of an animal or plant to act in response to an external stimulus.
in particular in subsection 2.2.3.1 in which the mechanisms of the agent were introduced, this agent appeared as exceedingly simple.

The reflexive\textsuperscript{12} agent is fully defined as a 6-tuple:

\[ < S, I, E, \text{sense, act, effect} > \]

2.3.2. Deliberative agent

A deliberative agent possesses an internal model of its world and uses this model to reason about the affects of its actions in order to select actions that it unambiguously predicts will accomplish its immediate task, as schematised in Figure 9.

\begin{table}[h]
\centering
\begin{tabular}{ll}
Predictive: & An agent is predictive if its model of how the world works is sufficiently accurate to allow it to predict correctly how it can achieve the task. \\
Interpretive: & An agent is interpretive if it can correctly interpret its sensor readings. \\
Rational: & An agent is rational if it chooses to perform commands that it predicts will achieve its goals. \\
Sound: & An agent is sound if it is predictive, interpretive and rational. \\
\end{tabular}
\caption{Deliberative Agent Properties}
\end{table}

\textsuperscript{12} A reflexive (tropicstic) agent with a memory, as exemplified by a certain internal state of an agent, is then referred to as an hysteretic agent (Genesereth and Nilsson, 1986). For this purpose, a collection of internal states \( I \) is introduced. Each such internal state influences the agent's actions, so that the function action in this case becomes:

\[ \text{act}: I \times J \rightarrow E \]

At the same time, the 'memory update', as exemplified by an alternation of internal states, takes place, so that

\[ \text{internal}: I \times J \rightarrow I \]

Putting the pieces together, a hysteretic agent is defined as an 8-tuple:

\[ < I, S, J, E, \text{sense, act, effect, internal} > \]
An agent's internal model of its environment must provide a certain minimum, basic functionality. In order to reason about the consequences of actions, the model must predict how its actions will affect its external state. The model must also be able to derive information about the external state from its sensor outputs. A sensor model is also needed in order to predict which perceptions it may expect in its predicted future external states. In addition to the model, the agent needs to make an estimate of the current external state. It is from this estimated external state that the agent makes its inferences or its projections of the consequences of its set of potential actions. The result of an agent's deliberation process is a plan to accomplish its task. The agent needs to maintain a representation of the plan in order to be able to release the chosen actions at the appropriate times. The representation of the plan also allows the agent to elaborate and revise the plan further as new information is gathered and more computation is done. This plan must however be unambiguous.

[The restriction to an unambiguous plan arises from problems that have long been recognised in decision theory generally and in social-choice theory in particular. Ambiguous plans and impossible plans form a central area of study in these disciplines, often centred around the so-called Arrow impossibility theorems (Arrow, 1951; see Kelly, 1978). We shall not discuss this area of difficulty in the present work: we have enough of such areas here already!]

*Figure 10 Deliberative agent system (adapted from Goodwin, 1993)*
2.3.2.1. Deliberative properties

A deliberative agent depends on its model of its world in order for it to accomplish its task. The properties defined for deliberative agents characterise the accuracy and the suitability of the model for its task and how well the agent uses its model. Predictiveness is a characterisation of the model's ability to infer information about the state of the world from its agent's sensors. Independently of the accuracy of the model, an agent is said to be rational if it behaves in a consistent way and in accordance with its model of its world (see again, however, even if only by way of comparison, Kelly, 1978). A rational agent with a correct model is said to be sound.

2.3.3. Predictive Agent

An agent is predictive if its model of its world allows it to predict the results of its actions. A correct prediction relation must predict all the possible external states that could result while not predicting impossible states. In this ideal case, the projective function is the same as the composition of the mechanism's effect function with the environment's consequence function.

![Deliberative Agent Properties](adapted from Goodwin, 1993)

*Figure 11 Deliberative Agent Properties (adapted from Goodwin, 1993)*

Given an environment and a mechanism, we can determine the correct projective relation. The definition of this relation is however complicated by the fact that we do not want it to depend
on any hidden state: the projection function is not required to be 'clairvoyant' (or extrasensory). The functional relation can be created by composing the effects and consequence functions to determine which external state will result from a given initial external state and action. It is in this sense that the projection, even as it causes a mapping of one external state into another external state, can also be said to establish a relation between the states. The correctness of the prediction relation assures that any state that is indistinguishable from a given initial state can only produce again a state that is indistinguishable from that produced by the initial state.

It is also possible for an agent to have separate models of how the world works and how the agent’s mechanism works. These individual models would correspond to the effect and consequence functions. Having separate models would allow the agent to reason about what interactions are required to produce a desired result and then to reason about how to produce this interaction. For simplicity, it is supposed that the two models can be composed to provide a single relation.

2.3.4. Interpretive Agent

An agent is interpretive if its model of its sensors allows it to interpret the perceptions its receives correctly. The interpretation must of course then be ‘reasonably’ accurate. The relation should include all the possible relations between its external state (environment) and the perception of this state by the agent, while excluding those that are not possible.

2.3.5. Rational Agent

An agent may now be defined as rational if it adopts plans that it predicts will succeed before plans that it predicts will not succeed. For a more formal definition we yet again refer to Kelly (1978, p.19). A deliberative agent can predict the future by repeatedly applying its projective and interpretive relations to its current estimate of the external state. The agent can use this ability to predict the future in order to simulate the execution of its current plan. If such a simulation leads to the accomplishment of the task, then the plan is predicted to succeed.

In order to define a rational agent, we need to be able to determine the result of having the agent simulate a plan. There can be multiple possible predicted steps for any given state because in this case the agent’s projective and interpretive operations are strictly speaking relations (and so relating within one set) and not functions (mapping from one set into another set).

During the planning process, the agent revises its plan while attempting to improve it. Occasionally, the agent will adopt a revised plan. An agent is then rational if the revised plan that it adopts is predicted to be at least as successful as was the old plan. A plan is predicted to be at least as successful as another plan if it is predicted to succeed under a super-set of conditions under which the second plan is predicted to succeed. This definition of rational is of course very restrictive. A preferable definition would allow a rational agent to prefer plans where the predicted probability of success or the expected utility is higher. This requires the introduction of one or more probability distributions over possible events, which is a
complication that will in fact be introduced later in this work.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{task_hierarchy}
\caption{Task Hierarchy Overview (adapted from Goodwin, 1993)}
\end{figure}

2.3.6. Sound Agent

A sound agent has been defined as one that is predictive, interpretive and rational. That is, its projective and interpretive relations are valid and it selects plans that it predicts will succeed under a variety of conditions.

A sound agent is not necessarily successful. The agent's mechanism may not be capable or perceptive enough to accomplish the task that the agent is set or which it sets itself. In such a case, there may be no plan available that can achieve this task. Even if a successful plan exists, the agent may not be able to generate it. Just because an agent can decide correctly whether a plan will succeed or not does not mean it can itself generate a successful plan. The agent's plan generator may not be complete. It is also possible that it may take the agent too long to generate the successful plan: a complete plan generator will eventually generate a successful plan, if one exists, but by the time it has done so some deadline for the performance of the task may have passed.

2.4. Formal Definitions

Following closely Wooldridge and Jennings (1995) the notions of weak and strong agency are introduced. Thus (Wooldridge and Jennings, 1995):

"A weak notion of agency [assumes the existence of a] hardware- and software-based computer system that possesses the following properties:
• **autonomy:** agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;

• **social ability:** agents interact with other agents (and possibly humans) via some kind of *agent-communication language* (Genesereth and Ketchpel, 1995);

• **reactivity:** agents perceive their environment, (which may be the physical world, a user interacting through a graphical user interface, a collection of other agents, the INTERNET, or perhaps some others or some or all of these combined), and respond in a timely fashion to changes that occur in it;

• **pro-activeness:** agents do not simply act in response to the environment, they are able to exhibit goal-directed behaviour by *taking the initiative*.

A stronger notion of agency: For some researchers - particularly those working in AI - the term 'agent' has a stronger and more specific meaning than that introduced above. These researchers generally understand an agent as a computer system that, in addition to having the properties listed above, is either conceptualised or implemented using concepts that are more usually applied to humans. For example, it is quite common in AI to characterise an agent using *mentalistic* notions, such as knowledge, belief, and obligation (Shoham, 1993). Some AI researchers have gone even further, and considered *emotional* agents (Bates, 1994). (Lest the reader suppose that this is just a pointless anthropomorphism, it should be observed that there are some sound arguments in favour of designing and building agents using imitations of human-like mental states [even though there are clearly serious limitations to this: see, for example, Dreyfus (1972)].)

In a similar vein, but more concisely, Hayes-Roth (1995) descriptively introduces ‘*Intelligent agents* [as being able to] perform three functions continuously: perceptions of dynamic conditions in their environment; actions to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions. Conceptually, perception informs reasoning and reasoning guides action, although in some cases perception may drive action directly. This abstract definition allows for a great variety of biological and artificial agents whose capabilities range from the extremely limited and stereotyped to some exhibiting a quite sophisticated and versatile behaviour.’

In the continuation, a short analysis from a more biological perspective of intelligent behaviour in natural agents - animals, including *Homo sapiens* - is presented, followed by an analysis of the by-now classical approaches taken towards modelling this intelligent behaviour, together with an outline of the causes of their frequent failures. A more novel approach towards modelling intelligent behaviour, namely one based on a society of interacting specialist-agents, is outlined immediately after this.
3. Intelligence

If a machine is expected to be infallible, it cannot also be intelligent.

Alan Turing

In order to be understood, the intelligence of an individual must be observed and analysed within its social and cultural context. In contrast to traditional AI, which addresses intelligence as a phenomenon attributed solely to the individual agent, the present approach is founded on the belief that intelligent behaviour is inextricably linked to its cultural context and cannot be properly addressed when studied in isolation. This work attempts to study intelligent behaviour within its natural habitat - society, together with emergent phenomena occurring at the level of society.

3.1. Intelligence in Animals (including Homo sapiens)

Before Darwin’s time, it was most commonly assumed that the behaviour of animals was under the control of blind instinct. This was flattering to man, who prided himself on his possession of reason. This sharp divide between man and the brutes\(^\text{13}\) was challenged by

\(^{13}\) Plato was probably the first who argued that man’s actions were so much the results of his reasoning. Similar views were incorporated into Christian doctrine largely through the writings of Thomas Aquinas (1224-1274), who wrote that:
Darwin and his theory of natural selection. In his illustrious book *The Descent of a Man* Darwin argued that 'animals possess some power of reasoning' and that 'the difference in mind between man and the higher animals, great as it is, certainly is one of degree and not of kind'.

The notion of the soul, as the essential indwelling, individualising and animating principle of human bodily existence, as this was carried over from the Scholastics by Descartes, was probably one of the earliest approaches to the description and explanation of intelligent behaviour to survive into the 'modern' period. Descartes followed up the proposition that all natural phenomena are in the last resort of a physical nature and that they might even be explained mechanically or in mechanical terms, so that the behaviour of animals, supposedly having no souls, might be reduced simply to the behaviour of automata.

Within a somewhat different context and using a different vocabulary, essentially the same notion of a single, centralised governor that managed the rest of the body came to persist in and dominate early medical physiology, later psychology and, most recently, artificial intelligence. Zoology, on the other hand, established quite a different approach, even though based on an analysis of the behaviour of some quite primitive animals that have no central nervous system and which yet, despite this fact, exhibit rather complex behaviours. Following this perspective, an animal's internal 'architecture' was responsible for a rich behavioural repertoire which was essentially composed of a number of sub-systems, each of them responsible for a number of primitive activities (like perception, vision, memory, etc.).

From the work of Darwin onwards, it became generally accepted that in animals, including *Homo Sapiens*, intelligent behaviour originated through a process of natural selection. Given

"Man has sensuous desire, and rational desire or will. He is not absolutely determined in his desires or actions by sense impressions as is the brute, but possesses a faculty of self determination, whereby he is able to act or not to act... The will is determined by what intelligence conceives to be the good, by a rational purpose... man is free because he is rational, because he is not driven into action by an external cause without his consent, and because he can choose between the means of realising the good or purpose which his reason conceives." (*Summa Theologica*)

14 Descartes actually proposed that man is under the dual authority of his soul and of his body. In his treatise, *Passions of the Soul* (see McFarland and Bösser, 1993), he stated that:

"[passion] disposes the soul to desire those things which nature tells us are of use, and to persist in this desire, and also bring about the same agitation of spirits which customarily causes them to dispose the body to the movement which serves for the carrying into effect of these things."

However, from the point of view of current 'post-modern' theology (Barth, 1960, III, 4, p. 519):

"To live as man is to fashion nature through the spirit, but also to fashion the spirit through nature. It is the subjectivisation of the object, but also the objectivisation of the subject. In its appearance of the inward in the outward, but also the substantiation of the outward through the inward. It is the besouling of the body, but also the embodying of the soul."
the collection of phenotypical traits in an animal, the forces of evolution promote those genetically encoded phenotypes that are more capable of adapting to a current environmental situation. The ultimate objective is always to reach some kind of a goal, and the natural-evolutionary forces respect only the exigencies of this ultimate outcome. From the perspective of evolution, every behaviour can be considered intelligent so long as it produces 'the correct answer', irrespective of the methods, means and ways whereby the satisfactory answer is produced. Should the opposite be claimed, namely that the more involved processing necessary to solve a problem is more advanced or even more intelligent than some other, simpler mode of managing the situation, one would find oneself in the absurd situation of promoting certain class of solutions simply because they required more 'computing power' in order to be solved. This is, of course, simply the economic argument of 'Occam's razor' in another sense.

McFarland and Bösser (1993) offer an illustrative example taken from the animal kingdom:

"... in pointing to a particular intelligent behaviour in a particular species, we are not saying that other species are stupid in comparison. The other species may have solved the problem in a way that does not involve much intelligent behaviour. For example, many marine animals are sessile filter feeders. They do not have any sophisticated foraging behaviour, and they do not show much intelligent behaviour in connection with feeding. What they do have is an efficient, intelligently defined filtering and pumping system, which enables them to draw in water and extract the food particles from it. Such animal are well adapted to their environment, largely by virtue of their morphological design. Similarly, in comparing pigeons and bats, we can see that pigeons are much better at navigation, whereas bats are much better at foraging at night. It makes no sense to try to compare the general intelligence of these two animals, but it does make sense to ask to what extent their special abilities depend on intelligent behaviour rather than on morphology."

The general point in this respect is that it is not satisfactory to label certain mechanisms or certain organisms as intelligent and others, at the same time, as non-intelligent. It is most suitable, for our purposes, to relate intelligence (i.e. mechanisms for controlling behavioural sequences) to the evaluation of animal, robot or whatever other such activities. Thus intelligence can also be perceived as a conceptual characterisation of individual behavioural responses to conditions in a neighbouring environment and an internal state. Intelligence is then a strategy that is directed to a certain aim or set of aims, and this aspect will become particularly pronounced during the discussion of cognition which will be presented shortly.

Another point to be emphasised here is design. The evolutionary forces can themselves be depicted as being directed to designing instruments. Modern evolutionary theory implies that organisms tend to develop characteristics that offer no competitive and reproductive disadvantage with respect to other rivalling organisms within a specified stable environment. Thus, organisms, such as animals, being 'guided' by evolutionary forces, develop more-or-less optimal characteristics (at least as concerns the characteristics that satisfy the constraints imposed by their varying environments) with respect to their own immediately prevailing circumstances. This does not necessarily imply that each and every organism is itself a perfect, optimal specimen, since, firstly, the natural environment changes and the optimal
configuration migrates in principle faster than does the evolutionary adaptation, so that organisms tend to lag behind, and, secondly, there is a considerable diversity in the genetic constitutions in animals of the same species, and indeed one that may prove to be useful in some future states of the environment so as to provide these specimens with comparative advantages later in the evolutionary process.

Our main intention in this, albeit superficial, biological interlude is to introduce the notion of a biological designing agent that will subsequently be exploited so extensively in this work. One of the main propositions of this thesis is that certain natural principles should be introduced into the designing of artifacts that were traditionally engineered by humans. Moreover, we want to emphasize that the design of artifacts capable of performing intelligently cannot be considered without consideration of the parts from which the artifact is assembled. Thus, given the available apparatus, the sessile feeders perform extremely well by any standards of 'intelligence'.

Traditionally, and again very anthropocentrically, intelligent behaviour and cognitive ability are often linked. We, as representatives of the species *Homo sapiens* are animals that are capable of cognitive processing. It is not very good scientific thinking to assume that all other animals (and for that purpose, plants) possess cognitive capabilities similar to our own.

Cognition, according to the *Encyclopædia Britannica*, means ‘knowing’, in the widest sense of the term. In psychology it is used to denote one of three ultimate functions or processes of consciousness, the others being feeling and conation (i.e. the active aspects of mind, including desire, volition and willing). Cognition includes every mental process that can be described as an experience of knowing as distinguished from an experience of feeling or of desiring and willing; it includes in short, all processes of consciousness from which knowledge is built up. In one of its most familiar and fully developed form it provides the judgement, in which a certain entity (known logically and grammatically as a ‘subject’) is discriminated from other entities and characterised by some concept or concepts. Although cognition is readily distinguishable from feeling and conation, in the actual flow of mental life the three types of experience are always found together, and not separately, even though one of them is usually predominant in any one total experience, and another in another - and this property facilitates their mutual discrimination. Psychology, as a descriptive science, is not concerned with such epistemological question as that of how external objects can be revealed in subjective experiences; it simply takes at their face value these cognitive experiences in which objects appear to be known somehow and leaves the critical problems to epistemology and, even to logic.

In other disciplines, again, and in particular in theology, it becomes essential to distinguish cognition from knowledge. Thus, for Barth (I, 1, 1960, p.188; see Abbott, 1994, p.11), ‘knowledge arises from men and women when, knowing, they are affected by the object known. They no longer exist without it, but with it... Its truth has come home to them, has become their own... This event, this confirmation, in contrast to mere cognisance, we call knowledge. Cognisance becomes knowledge when men becomes a responsible witness to its content’.

Cognitive processing provides clear advantages in an often rapidly changing environment.
Should the environment be fixed, with constant properties, cognitive capabilities would be similar to some superimposed supplementary processing layer that could be much more easily and more effectively implemented in biological 'hardware', through the play of reflexes. Thus, there would be no comparative advantage to learning even ever-repeating events. There would not even be any purpose in learning the environmental changes that occur on a cyclic basis, since those could be just as successfully tackled through using endogenous biological clocks, as indeed occurs in many organisms under very finely tuned cyclic (e.g. lunar, tidal) conditions. Thus, in principle, those environmental changes that can be accurately predicted over long periods of time can be better handled by introducing pre-programmed codes in the organism's composition. A similar approach should in this case be followed in the design of computational agents.

During its lifetime, almost every natural agent is confronted with a number of unforeseen circumstances that cannot be dealt with using preprogrammed patterns of behaviour. For the individual agent that can utilise past experiences, various forms of learning, like habituation and stimulus substitution, provide advantages and call for a form of cognition. Thus, cognition, as we may perceive it for the purposes of this work, is a form of phenotypic adaptation. It has to be emphasised that, even though the basis for cognition is realised genetically, cognition covers the forms of adaptation that cannot be genetically inherited.

Cognition, as used in animal-behavioural studies, generally refers to mental processes that are presumed to be occurring ‘within’ the animal, but which cannot be observed directly. Animal psychologists are particularly interested in cognition. The process involved seems to imply the presence of some kind of capacity for mental abstraction on the part of the animal. The study of problem-solving, and especially tool-using, in primates has led many scientists to believe that some of our fellow animals, such as the chimpanzees, are capable of forming a cognitive map or model of their external environment. Sometimes these cognitive processes themselves involve social relationships. The many recorded instances of cooperative behaviour, tool using and so-called detour negotiation in certain primates, leave little doubt that these also have some sort of insight into situations which is difficult to explain in terms of ‘ordinary’ learning.

We have deliberately drawn a distinction between cognition and intelligence. Cognition, stated bluntly, represents a possible means to an end, whereas intelligence involves an assessment of performance in terms of some functional criteria. Should human intelligence be utilised as a basis for a models of intelligence in artificial media, the distinction between intelligence and cognition should be maintained. ‘As is true of animal behaviour, many aspects of human behaviour do not involve cognition... Because it is so difficult to attain agreement as to what human intelligence involves, it might be more profitable to consider the nature of intelligent behaviour and define intelligence. This approach also might enable us to compare the intelligent behaviour of humans, animals and machines' (McFarland and Bösser, 1993).

3.2. Intelligence in Machines

The quest to design intelligent control in artificial systems dates back a long time. Long before the advent of the electronic computer, the engineer James Watt (1736-1819) popularised the use of a mechanical feedback control as a way of automatically regulating the
velocity of rotation of steam engines, thereby controlling their energy intake and power. These feedback control devices, known as governors, were designed to refine actions and produce stability in dynamical systems through a process called negative feedback. In other words, by feeding some chosen output from the system back to its input, a means was provided of comparing actual and intended performance so that compensatory changes in the input could be made. A similar example is provided by the thermostat, where a change in the temperature of the main object causes a change in a signal which in turn causes the heat supply to become more or less intense or more or less effective.

_Cognitive Science_, the science of ‘virtual machines’ that are capable of exhibiting a behaviour that can be regarded as intelligent, is a relatively young discipline, being just over forty years old. Varela (1992) offers a concise overview and divides the development of this science into several stages: its foundational years, the arise of the cognitivist paradigm, and the contemporary stage characterised by such innovative approaches as emergence (as an alternative to symbol manipulation) and enaction (as an alternative to representation).

The initial excitement of the foundational years of the 1940s and 1950s provided a great impetus for this new discipline. Virtually all the topics that remain at the focus of interest, even today, were identified. ‘The ‘founding fathers’ knew very well that their concerns amounted to a new science, and christened it with a new name: cybernetics. This name is not in current use, and many cognitive scientists today would not even recognise the family connection’ (Varela, 1992, p. 236). This founding stage initiated an extraordinary assortment of directions of multidisciplinary research, most of which are still active (Varela, 1992):

- the use of mathematical logic to understand the operation of nervous systems;
- the invention of information processing machines (as codings of digital computers), subsequently opening the path for the creation of artificial intelligence;
- the establishment of the meta-discipline of system theory, which has had an impact on many branches of science, such as engineering (system analysis, control theory), biology (regulatory physiology, ecology), social sciences (family therapy, structural anthropology, management, urban studies), and economics (game theory, social welfare theory);
- information theory as a statistical theory of signal and communication channels;
- the first examples of self-organising systems.

Let us stay in these foundational, golden years for just a while longer and let us try to clarify the intentions and ambitions of cybernetics. Following works of the investigators like Turing, McCulloch, Pitts, Ashby or Wiener, we can recognise the intention of creating models of and explanations for mental phenomena which would result in a science of mind.

The artifacts of cybernetics were based on analog computer technology, developed during World War II for the purposes of supporting military machinery and through developments

---

15 The term cybernetics derives from the Greek _kubernetes_ meaning ‘governor’. It was coined by Norbert Wiener of the Massachusetts Institute of Technology (MIT) in order to name a unified study of control and communication in animals and machines.
in precision bombing, aircraft navigation, training anti-aircraft guns, and other such activities. The early cyberneticists clearly recognised a similarity between the control of such machinery and internal regulatory control in living systems (such as the homeostatic control of the glucose level in the blood, or the body temperature in humans, as apposite examples).

Towards the end of that war, the first digital computers emerged and were used for military-scientific computations, including code-breaking and artillery calculations. This new category of programmable, digital machines provided unprecedented opportunities. Pioneers like John von Neumann and Alan Turing raised the hope that the ability to think rationally could be embodied in some such machine.

In the *Automatic Computing Engine*[^16] (ACE lecture, delivered to the London Mathematical Society on February 20, 1947) on the nature of programmable digital machines, Alan Turing meticulously predicted many of the developments that would take place in the years to come. We will use some of the messages delivered at this lecture to accentuate the present line of reasoning. Firstly (see also Michie, 1993), Turing suggested that *logic was the* discipline needed to interpret mental activity and went even further in saying that:

> "I expect that digital computing machines will eventually stimulate a considerable interest in *symbolic logic and mathematical philosophy*; ...in principle one should be able to communicate in any symbolic logic, provided that the machine were given instruction tables which would enable it to *interpret that logical system.*"

He continued further to depict human intelligent performance as something that was strictly conscious, *(i.e. capable of being represented at a symbolic level). Turing claimed that, to the extent that the assumption of the possession of consciousness in other people is not refutable,

---

[^16]: In the very same lecture, Alan Turing proposed yet another paradigm, and foresaw one of its most significant sociological implications, namely that of *machine learning* (Michie, 1993):

> "Let us suppose we have set up a machine with certain initial instruction tables, so constructed that these tables might on occasion, if good reason arose, *modify those tables*. One can imagine that after the machine had been operating for some time the instructions would have been altered out of all recognition, but nevertheless still be such that one would have to admit that the machine was still doing very worthwhile calculations. Possibly it might still be getting results of the type desired when the machine was first set up, but in a much more efficient manner. In such a case one would have to admit that the progress of the machine had not been foreseen when its original instructions were put in. It would be like a pupil who had learnt much from his master, but had *added much more by his own work.*"

But then Turing as well forecasted that a certain category of people, namely those attending the digital programmable machines, might be:

> "... unwilling to let their jobs be stolen from them in this way. In that case they will surround the whole of their work with mystery and make excuses, couched in well chosen gibberish, whenever any dangerous suggestions are made... This topic [of automatic programming] naturally leads to the question as to how far it is possible in principle for a computing machine to simulate human activities..."
we conveniently assume that they are conscious. We should therefore be equally ready to abandon any kind of solipsism when assessing the capabilities of thinking machines (see Michie, 1993):

"... according to the most extreme form of this view [that thought is impossible without consciousness] the only way by which one could be sure that a machine thinks is to be the machine and to feel oneself thinking. One could then describe these feelings to the world, but of course no one would be justified in taking any notice. Likewise according to this view the only way to know that a man thinks is to be that particular man. It is in fact a solipsist point of view. It may be the most logical view to hold but it makes communication of ideas difficult. A is liable to believe 'A thinks but B does not', whilst B believes 'B thinks but A does not'. Instead of arguing continually over this question it is usual to have the polite convention that everyone thinks... I think that most of those who support the argument from consciousness could be persuaded to abandon it rather than to be forced into a solipsist position. They will then probably be willing to accept our [Turing] test."

The famous, but much disputed Turing test is a culmination of these ideas. There is an obvious thread running between these ideas and the further development of rational-reasoning - from cybernetics to cognitivism. One immediate consequence of these early ideas was that man’s cognitive facilities became replaced conceptually by a machine based on symbols, and this machine, roughly speaking, came to be understood as a program (code) that was domain independent.

In 1960 John McCarthy introduced the term Artificial Intelligence to describe the combination of all the efforts that were being made to produce a thinking machine. By the mid-sixties, a number of programs had been designed that could play both chess and checkers, solve algebraic problems and prove theorems in geometry. The AI of those times could solve such problems in a rather narrow domain, but in certain cases their performance was even then of an extremely high level. Today, several chess playing programs are officially rated as chess masters, even if not grand masters, while backgammon and checkers programs are now winning in many national and international competitions. So-called expert systems, understood as programs encoding decisions in symbolic, if-then-else rule-formats, are attractive business activities for the companies which produce and use them in domains like computer

17 In 1950, Alan Turing proposed the following criterion for determining whether a machine could think. This method has become known as ‘the Turing test’. To conduct this test, we would need two persons and the machine to be evaluated. One of the people would play the role of an interrogator. The interrogator would be in a separate room from the computer and the other person, with whom, however, he or she could communicate by typing questions and receiving typed responses. The interrogator could ask questions of either the person or the computer, but would not know which of them was which. The interrogator would know them as A and B, and would aim to determine which of them was a person and which was the machine. If the machine were to succeed in leaving the interrogator unclear as to which was which as a result of playing this game, then we could conclude that the machine was able to think. We should add that the machine is allowed to do whatever it can to fool the interrogator. So, for example, if asked the question: "How much is 3563 times 754283?", it is allowed to wait for several minutes and then respond with the wrong answer.
configuration, infectious disease diagnosis or plant scheduling.

Thus, the foundational years were superseded by a stage represented by a cognitivist paradigm that, roughly speaking, declared that cognition and intelligence could be defined in terms of computations of symbolic relations. The cognitive activity itself is then defined as an information processing activity grounded in the rule-based manipulation of symbols. This manipulation of symbols, it must be emphasised, is not founded on the meaning of the symbols, but only in their syntax. Semantic interpretability (or meaningfulness) is one of the defining features of the 'language of thought' (Fodor, 1975) which forms the prevailing perspective in the symbolic model of the mind. Thus, as a particularly extreme example of this ideology, 'the mind is a symbol system and cognition is symbol manipulation. The possibility of generating complex behaviour through symbol manipulation was empirically demonstrated by successes in the field of artificial intelligence' (Harnad, 1990). Based on Newell (1980), a symbol system is (Harnad, 1990):

- A set or arbitrary physical tokens (scratches on paper, holes on tape, events in a digital computer, etc.) that are
- manipulated on the basis of explicit rules that are
- likewise physical tokens and strings of tokens. The rule-governed symbol-token manipulation is based
  - purely on the shape of the symbol tokens (not their 'meaning'), i.e. it is purely syntactic and consists of
  - rulefully combining and recombining symbol tokens. There are
  - primitive atomic symbol tokens and
  - composite symbol-token strings. The entire system and all its parts - the atomic tokens, the composite tokens, the syntactic manipulations (both actual and possible) and the rules - are all
  - semantically interpretable: the syntax can be systematically assigned a meaning (e.g. standing for objects, as describing states of affairs).

According to this symbolic-model-of-mind paradigm, symbolic structures are capable of catching such mental phenomena as beliefs or thoughts. This, symbolic, level is a segregated functional level on its own, with its own separate grammar which is different and independent from the grammar of the physical realisations of the constituting processes. Fodor (1975) even claimed that the 'meaning of symbols comes from connecting the symbol system to the world in "the right way"'.

The above definition introduces a number of critical components of a symbolic system. We shall here paraphrase the discussion by Harnad (1990) who emphasises that it is not enough for a phenomenon to be interpretable as rule-based to be regarded as symbolic. Already Wittgenstein (1953) distinguished between implicit and explicit rules as the difference between 'following' a rule (explicitly) and 'behaving in accordance with' a rule (implicitly). The decisive qualities are the compositeness of 'atomic' symbol tokens and the criteria of systematicity. Thus the fact that a behaviour is 'interpretable' as a collection of rules, does not mean that it is based on a collection of symbolic rules. The system, in order to be a symbolic one, must be, strictly speaking, a semantic interpretation based on an explicit representation, syntactically manipulable with a systematically-assigned meaning.
Symbolists believe that cognition, being only a process of symbol manipulation (or 'symbol shuffling' to use M.B. Abbott's phrase), is an autonomous functional 'module' that need, supposedly, only be connected to the objects of the world (or, more accurately, the representation of the objects in the world) for the cognitive processes to function in the world. Some, rather insular, forms of symbolic reasoning, like geometry or algebra, do indeed allow for this independence. 'Unfortunately, this radically underestimates the difficulty of picking out the objects, events and states of affairs in the world that symbols refer to, i.e. it trivialises the symbol grounding problem' (Harnad, 1990). In order to provide an intelligent behaviour, the cognitive apparatus, and thus its reasoning facilities, must be connected to the context of meaning.

Let us continue along these, symbolic-cognitivist, lines just for the moment so to arrive at what Newell (1981, 1982) refers to as the knowledge level. According to this view, there are two primary levels of characterisation related to a model of intelligence in any agent, whether natural or artificial: the symbol level and the knowledge level.

The symbol level here corresponds to the program used by an intelligent agent together with its model of its representation of the world, which then becomes the world.

The knowledge level, however, is located 'above' this symbol level; it is more complex and is represented as a collection of actions which can be conducted by an agent, the set of goals which provide the agent's driving force and towards which it strives, the knowledge necessary to achieve these goals through the application of appropriate actions and, finally, a principle of rationality.

The principle of rationality now comes to state that, if an agent possesses the knowledge that is applicable within the context of a particular problem, that agent will apply this knowledge in order to achieve one of its goals or satisfy one of its needs. Thus, in order to model an aspect of intelligent behaviour according to this view, one has to identify goals, possible actions and necessary knowledge and then implement them at the symbol level. At the symbol level, in turn, there should be a formalism for the representation of knowledge and a certain computational (algorithmic) interpretation of such a principle of rationality.

"When [we AI researchers] say, as we often do in explaining an action of a program, that the 'program knows K' ... we mean that there is some structure in the program that we may view as holding K"

(Newell, 1981, p.15)

Thus, Newell basically claimed that the knowledge level can be reduced to the symbol level, and to a corresponding collection of data structures. Angeline (1993) refers to these models of intelligence, 'where there is a direct correspondence between the knowledge described at the knowledge level and the content of representations at the symbol level, as literal knowledge level models of intelligent behaviour'.

The symbol-cognitivist postulates have strongly influenced many early approaches to artificial intelligence. Some of these, due to their pragmatic performance, are applicable and are being
applied even today. However, this approach must be criticised along (at least) two basic lines. Firstly, the symbolic computation must be examined, as a 'platform' for 'intelligent computations', (Is 'symbol shuffling' really the core of intelligent behaviour?) and, secondly, the adequacy of this representation of the real world must be considered.

It is understandable that the symbol-cognitivist approach is popular. It appeals to our senses since it is easy to grasp and it corresponds to what we like to think that we normally do. It produces results, at least so long as we ascribe only modest ambitions to an intelligent agent. Such a situation holds, for example, in strongly constrained universes of discourse. Each discrete physical item in the system has its corresponding cognitive interpretation and corresponding meaning. The cognitive mapping, if you will, is complete. However, this, symbolic approach is increasingly seen to be 'naive, full of methodological pitfalls, and [it] violates some fundamental behavioural principles' (McFarland and Bösser, 1993). Should the universe of discourse becomes unconstrained, syntax is the only remaining power supposed to account for the rest of the world (beyond constraints), and this is clearly, not enough. As early as 1911, A.N. Whitehead wrote:

"It is profoundly erroneous truism... that we would cultivate the habit of thinking what we are doing. The precise opposite is the case. Civilisation advances by extending the number of important operations which we can perform without thinking about them."

Abbott (1994, unpublished correspondence) finds much stronger words to illustrate the symbolic orientation in AI in particular:

"Most of the acolytes of AI started out already with higher order languages, such as LISP, with its conceptual and notational relations to Church’s lambda calculus, so that they were already launched on the way of illusion at this most basic level of working language. By concentrating on production systems [i.e. the symbolic paradigm in AI] and working with 'meaningless expressive signs' it became possible to avoid the real problems involved in using the programmable digital machine to aid and to enhance our learning capabilities."

Instead of placing the onus for representing a core of activities on symbols, another approach is to begin with somewhat simpler, sub-cognitive elements which can be interconnected, or can interact in some way\(^\text{18}\). These atomic elements are only competent in their own, in

\(^{18}\text{Many skills can be learned without their prior description in a symbolic form. As Posner (1973) demonstrates (and everyone of us can easily and independently verify):}

"If a skilled typist is asked to type the alphabet, he can do so in a few seconds and with very low probability of error. If however, he is given a diagram of his keyboard and asked to fill in the letters in alphabetical order, he finds the task difficult. It requires several minutes to perform and the likelihood of error is high. Moreover, the typist often reports that he can only obtain the visual location of some letters by trying to type the letter and then determining where his finger would be. These observations indicate that experience with typing produces
principle very limited and not necessarily symbolic, domain of expertise. These local processing units, because of their dense inter-connectivity, can emerge spontaneously, without any central processing or coordinating element that would correlate and organise the entire operation of reaching a mutually satisfactory state for all atomic elements. 'This passage from local rules to global coherence is at the heart of what used to be called self-organisation during the foundation years. Today, different people prefer to speak about emergent of global properties, network dynamics, or even synergetics. Although there is no unified formal theory of emergent properties, the most obvious regional theory is that of attractors in dynamical systems theory. These are not the property of an individual components, but of the entire system, yet each component contributes to its emergence and characteristics' (Varela, 1992).

Strengths of symbol systems

1. **Symbolic function:** Symbols have the computing power of Turing machines and the systematic properties of a formal syntax that is semantically interpretable

2. **Generality:** All computable functions (including all cognitive functions) are equivalent to a computational state in a Turing machine

3. **Practical successes:** Symbol systems' ability to generate intelligent behaviour is demonstrated by successes of Artificial Intelligence

Weaknesses of symbol systems

1. **Symbolic function:** Symbol systems are subject to the symbol grounding problem

2. **Generality:** Turing power is too general. The solutions to Artificial Intelligence's toy problems do not give rise to common principles of cognition but to a vast variety of ad hoc symbolic strategies

Figure 13 Strengths and weaknesses of symbol system (adapted from Harnad, 1990)

This alternative orientation, which is the one adopted to a large degree in this work as well, is that in which cognition emerges as a global state of competence in a network of simple interacting components. It performs the computation through the application of rules providing a local succession at the level of these simple components and through introducing a corresponding interaction of these instructions executed locally. When the emergent properties (and resulting emergent structures) can be seen to correspond to a specific cognitive task, a successful solution is said to have emerged.

Searle (1992), however, proposes, beyond this, that consciousness is a higher-order emergent...
... consciousness is a causally emergent property of systems. It is an emergent feature of certain systems of neurons in the same way that solidity and liquidity are emergent features of systems of molecules. The existence of consciousness can be explained by the causal interactions between elements of the brain at the micro level, but consciousness cannot itself be deduced from or calculated from the sheer physical structure of the neurons without some additional account of the causal relationships between them.

In this new, emergent paradigm, the symbols have no role in the classical sense of the word, or, to say the least, the role of symbols is downplayed radically. Cognition is perceived not as a process of symbol manipulation but is represented in terms of dynamic patterns of activity. If we, just for the moment, return to requirements for the symbol system, it becomes obvious that the emergent paradigm meets neither the compositeness nor the systematicity criteria. The interactions of atomic components in the emergent paradigm do not, in principle, result in 'simple', linearly-combined structures, and are not decomposable according to some formal syntactic rules. The sub-symbolic nature of atomic constituents acquires meaning through association with the global state of the emergent system, and is not located in a collection of particular symbols. Thus the syntactically-semantically grounded classical cognitivism is replaced by the system's global state for which a semantic interpretation is provided by and through the observer only.

The symbol-cognitivist approach can conveniently be described as top-down. This implies that the description of the system behaviour is located at some central position ('at the top'), from which all instructions are sent further to the lower levels. In the emergent, bottom-up, paradigm, it is the interplay of local interactions that results in the higher (emerging) capabilities.

The answer to the question concerning the 'best' architecture to represent processes of cognition is, however, neither black nor white. The relation between the symbolic and the emergent should provide certain integrated modes of operation, or simply be used at different stages. However, the only viable path 'from sense to symbols' is from the world (as is usually represented by our own observations, and in particular our measurements) to our symbols.

Varela (1992) states:

"In my view the most interesting relation between emergent and symbolic descriptions is the one of inclusion, that is, the view of symbols as a higher level of description of properties embedded in a underlying distributed system. The case of the so-called genetic code is paradigmatic, and I will use it here for concreteness. For many years biologists considered protein sequences as being instructions coded in DNA. However, it is clear that DNA triplets are capable of predictably specifying an amino acid in a protein if an only if they are embedded in the cell's metabolism, that is, in the midst of thousands of enzymatic regulations in a complex chemical network. It is only by the emergent regularities of such a network as a whole that we can bracket out this metabolic background, and that triplets become codes for amino acids. In other words,
Figure 14 (a) A Punch cartoon that depicts the cognitivist hypothesis. To catch its prey, the kingfisher has, in its brain, a representation of Snell’s law of refraction. (b) Another interpretation of the cartoon indicates how the symbolic levels can be seen, as arising from the network of interacting atomic components, in the observer (adapted from Varela, 1992)
the symbolic description is possible at another level of description. Clearly, it is possible to treat such symbolic regularities in their own right, but their status and interpretation is quite different than if taken at face value, with independence of the substratum from which they arise."

Thus, a free-floating, disconnected symbolic, cognitive machine cannot be constructed, since it raises the un-resolvable problem of acquiring a cognitive apparatus based on the intrinsic meaning of the symbols and implicitly accepting a reduction to the level of machine code in a programmable computing machine. As Harnad (1990) points out:

"In an intrinsically dedicated symbol system there are more constraints on the symbol tokens than merely syntactic ones. Symbols are manipulated not only on the basis of the arbitrary shape of their tokens, but also on the basis of the decidedly non arbitrary ‘shape’ of the iconic and categorical representations connected to the grounded elementary symbols out of which higher-order symbols are composed... The symbol meanings are accordingly not just parasitic on the meanings in the head of the interpreter, but intrinsic to the dedicated symbol system itself. This is still no guarantee that our model has captured subjective meaning, of course. But if the system’s behavioural capacities are lifesize, it is as close as we can ever hope to get."

The foundational ideas of cyberneticists, like Turing, Ashby of Wiener, all of whom cherished the hope that an ability to perform rationally might be one day transferred to a machine, were inherited by the early AI research community which adopted the view of problem-solving as a formal theorem proving, or as a heuristic search over a space of goal and sub-goal structures.

Unlike the sensory-motor-level and the biologically-inspired artifacts produced by cyberneticists, AI researchers have by and large tended to concentrate on building narrowly-defined, competence-orientated systems based on the formalisation and the simulation of human intellectual reasoning processes (as physical symbol systems).19

19 Within this paradigm we may further distinguish an unembodied path (Brooks, 1991) which is associated with programming intellectual activities such as chess playing, and an embodied path, which is concerned to provide machines with senses and sensitivities. The unembodied path runs through most AI work, whereas the embodied path is associated particularly with one of AI’s sub-fields, already introduced as robotics.
Strengths of connectionism

(1) *Nonsymbolic function*: As long as it does not aspire to be a symbol system, a connectionists network has the advantage of not being subject to the symbol grounding problem.

(2) *Generality*: connectionism applies the same small family of algorithms to many problems, whereas symbolism, being a methodology rather than an algorithm, relies on endless problem-specific symbolic rules.

(3) *‘Neurosimilitude’*: the connectionist architecture seems more brain-like than a Turing machine or a digital computer.

(4) *Pattern learning*: Connectionist networks are especially suited to the learning of patterns from data.

Weaknesses of connectionism

(1) *Nonsymbolic function*: Connectionist networks, because they are not symbol systems, do not have the systematic semantic properties that many cognitive phenomena appear to have.


(3) *‘Neurosimilitude’*: Connectionisms’s brain-likeness may be superficial and may (like toy models) camouflage deeper performance limitations.

*Figure 15* Although connectionist systems with their globally distributed dynamics cannot be simply equated with self-organised emergent systems, some of their properties can be presented in this way (adapted from Harnad, 1990).

In this work, however, due to the reasons just explained, we put the physical symbol system hypothesis to one side and adopt emergence as our governing principle.

### 3.3. Intelligence in Societies

Intelligence, like knowledge, is essentially a social phenomenon. Animals live by following certain collections of genetically coded or learned rules that, although they can sometimes be expressed in terms of strings of symbols, mainly require for their proper representation collections of tokens that are below this level and which are thus *sub-symbolic*. Social
organisations benefit from a community of kin, and in many cases not only or even necessarily their own kin but also, or even preferably, assemblies of agents that somehow share their interests and which benefit from aggregation. In societies, a collection of regulative policies emerges, apparently more or less spontaneously, and we can, for the time being, refer to this policy formation as one that proceeds through a shared culture.

The culture is, simultaneously, the result and the cause of intelligent behaviour. Agents in a society behave more-or-less intelligently and use these intelligent skills to propagate their interests within the society. Quite obviously, this is a two-way process. On the one side, individual behavioural habits emerge and blend into the culture, but at the same time the culture constraints some of the actions of an individual, and implicitly promote the interest of the group above the individual interests. As the social organisation grows and adapts to an ever changing environment, the social rules become more and more complex, the culture more intricate, and the demands placed upon the capabilities of the individual agents - their intelligences and learning proficiencies, and consequently abilities to absorb and acclimate to the culture - increase drastically. *Homo sapiens* is an extreme example of social and cultural intricacy, this being the species that requires the longest training and learning developmental period of all animals (see, in this respect, Gould, 1982).

Traditionally, AI has been more interested in modelling and analysing the behaviour of the individual agent than it has concerned itself with a group of agents. This explains why, to date, much AI research has been dedicated to studying single functions, such as perception, planning, learning, natural language processing, or belief modelling. In general, little attention has been devoted to the problems of integrating and deploying these capabilities in autonomous systems capable of operating effectively in complex environments.

Operating in the real-world means having to deal with multiple events at several levels of granularity, both in time and space. So, while agents must remain reactive and reflexive in order to maintain their vital functions, some amount of strategic knowledge or a certain predictive decision making will be required if agents are to coordinate their actions with other agents and handle complex, time-constrained goals while keeping their long-term options open. Agents, however, cannot be expected to model their surroundings in every detail as there will be simply too many events to consider, a large number of which will be of little or no relevance in any one given situation in any case. What is required, in effect, is an architecture that can cope with uncertainty, react to unforeseen events, and recover dynamically and gracefully from poor decisions, as well as learn the causes of these poor decisions so that they can be avoided in the future. All this, of course, comes on top of all the other tasks that the agent was originally assigned to do. Not surprisingly, it is becoming widely accepted that neither purely deliberative nor purely non-deliberative control techniques are capable of producing the range of effective, robust and flexible behaviours desired of intelligent agents.²⁰

---

²⁰ The term *deliberative* implies that the agent possesses reasonably explicit representations of its own beliefs and plans and the goals that it uses in deciding which action it should select at a given time. Conversely, *non-deliberative* implies that the agent's beliefs, plans, and goals are implicitly embedded or pre-compiled into the agent's structure by its designers.
3.3.1. Why decentralise?

Natural evolution, being a very persistent but, for us, blind force, has promoted the development of socialisation. However, whether it is due to our ignorance of the investigations of Thomas Aquinas, or Descartes, or our own arrogance or ignorance generally, the benefits of socialising have only recently, and even then rather modestly, started to enter the engineering arena. The utilisation of a distributed, multi-agent proficiency is the exception rather than a rule. The introduction of parallel, distributed, decentralised, non-hierarchical computation in fact necessitates a fundamental paradigmatic shift.

![Diagram](image)

**Figure 16** Traditional AI has addressed complex agents in simple environments, while reactive and behaviour-based approaches have dealt with simple agents in noisy and uncertain worlds. This work attempts to bridge this division. (adapted from Mataric, 1994)

The obvious question arises. Why be a member of a group? Why socialise if this restricts and constraints the private interests?

Socialising indeed provides many benefits. Looking through the biological magnifying glass, for example, we see that, by utilising the power of social learning, it is possible to compensate for many genetic deficiencies. For example, the exchange of survival-relevant information within the group, increased chances of mating and survival and the possibility to attack larger prey are all examples of benefits of grouping, and so of socialising. Through social organisation, agents are capable of adapting their behaviours over very short time scales. Certain social interactions may be created and destroyed in this way within a single generation. In such a way, the manoeuvre discovered by an individual may be easily transferred to others within the society and passed on, even to offspring. We shall shortly
identify this sort of progression as one of Lamarckian evolution.

At the same time, and again referring to the animal kingdom, it is well known that insect colonies, for example, are capable of collective 'problem-solving'. Deneubourg et al. (1992) asked whether a colony of termites might be able to build Notre-Dame de Paris if properly programmed, or whether ants could build the Golden Gate Bridge. He demonstrated that many structures in nature (such as wasp and termite nests) are erected by some 'primitive' agents that collectively accomplish such very complex assignments. These tasks are accomplished by so-called swarms.

A swarm is defined, within our present context, as a collection of autonomous agents which are capable of communicating either directly or indirectly (through affecting their local environment in such a way that the effect is perceived by other agents). Through such a collective action, swarms are capable of distributed problem solving. The functioning swarm results in functional collective patterns which are characterised through differentiation and spatio-temporal organisation of the agents within the swarm and also the parallel organisation of the material elements in the environment upon which each agent acts. The collective, resulting behaviour can be associated with swarm intelligence.

Nature demonstrates that collections of simple agents can perform tasks that are complex. From a software engineering (pragmatic) perspective, accordingly, it is easier to write a simple computer program, instantiate a number of these programs and simply observe how these interact and themselves solve more complex problems. Moreover, it is much easier to handle the complexity of such a specialist-agent than it is to handle the complexity of an entire swarm.

Such an approach offers other advantages as well, such as robustness when confronted with unexpected events, flexibility in the ways in which tasks are carried out, autonomy in the sense that agents have their own goals and can prioritise among them, and effectiveness, in the sense that the overall problem is partitioned into smaller, easy-to-handle sub-problems. At the same time, on parallel computers of MIMD or SIMD nature, the distribution of problem-solving to a number of computational agents may result in super-linear reductions in computational requirements.

From the more epistemological perspective, a decentralised problem-solving (that is, an intelligent-agent based approach) is the only possible approach to modelling a certain class of systems in which the dynamics of the interacting parts result in a complex and sometimes chaotic behaviour. Such an approach to simulation is more 'natural' and consequently results in a richer behavioural repertoire of emerging systems.

The construction of systems that are composed of a number of intelligent agents is also an objective of so-called Distributed Artificial Intelligence (DAI)-the branch of AI concerned with concurrent and distributed computations within AI. In this area, the compulsory reading starts with Bond and Gasser (1988). There are three major research directions within DAI. Distributed Problem-Solving (DPS) considers the problem of the division of labour and knowledge among agents in collective problem-solving. A sub-field of DAI, known as Multi-Agent (MA) systems, deals with the coordination of intelligent behaviour among a collection
of autonomous, intelligent agents. Lastly, although a discipline in its own right, Parallel AI (PAI) is concerned with parallel architectures, languages and algorithms suited for tackling certain specific problems of AI.

Although initial work in DAI has been concerned with fully knowledgable, resource-unbounded agents that operate in highly structured, artificial environments, more recently the main thrust of research has begun to move towards realistic, less-well-structured domains comprised of groups of agents which are constrained by limited resources and capabilities and have only a bounded rationality^{21}. In such cases, agents are expected to satisfy constraints imposed by their environment (so that they are ‘satisficing’), rather than to optimise their own state of existence.

Decentralized AI (DzAI) is concerned with activities of autonomous agents in a multi-agent world. Thus it can be regarded as an instance of MA. However, a distinction between DAI and DzAI should be carefully drawn. DAI is in principle concerned with the collaborative problem-solving of global (common) problems by a societies of entities. DzAI does not impose the global problem to be solved, but rather orientates itself towards a more powerful and better autonomous agent design as well as the simulation and analysis of social congregations of these agents, and in this way it seeks to allow emergent phenomena to arise from out of these interactions.

Most recently, a number of researchers from the fields of AI, DAI, robotics and Artificial Life have started to turn their attention to the problem of designing and implementing integrated control architectures for embedded intelligent agents, and so to the problems of realising intelligent agent architectures (IAA). Specifications of correctness for embedded agents amount to specifications of their interactions with their environment, posing such questions as: what action should the agent take when the environment has a particular configuration? Any model of the control for such an agent must express the possibility for the agent to respond appropriately to diverse and often rapidly-changing situations. The approach that is promoted in the present work has a strong inclination towards the analysis of the AI side of hydroinformatics systems. Thus, it basically addresses intelligent behaviour and the ways in which it can be modelled and applied in hydroinformatics systems.

3.3.2. Deliberative Architectures

Strongly inspired by certain successes of classical AI planning systems such as STRIPS (Fikes and Nilsson, 1990), deliberative systems are often supposed to act in a sense-act-plan loop, with the intention of generating a correct sequence of plans, the execution of which will result in a certain desired goal state. The deliberative planners assume the availability of complete and up-to-date knowledge about the world in which they are situated, as well as reasonably correct predictions of the developments of this world in the future. Examples of studies of such systems are the Intelligent Resources Bounded Architecture (IRMA) developed by

^{21} Bounded rationality implies that the complexity of agent’s environment is much greater than the complexity of the agent itself.
Bratman and his research team (Bratman, et al., 1988), the behaviour hierarchies of Durfee and Montgomery (1990) and a computational framework which Shoham (1993) has called *agent-oriented programming* (AOP). Agents are defined as entities possessing formal versions of mental states; that is, formal versions of knowledge, beliefs and the like.

From a software-engineering point of view, AOP can be regarded as a specialisation of *object-oriented programming* (OOP) in which the objects - in this case, agents - can possess various mental states about themselves and about one another. From a more formal point of view, AOP can be regarded as a specialisation of a formal language (a temporal epistemic logic of belief) augmented with various modal operators such as commitment, choice (commitment to oneself) and capability.

Some earlier works of the present author also belong to this, deliberative-architecture category. The *blackboard-based* multi-agent system, called Den Røde Tråd - 'The Red Thread' (Babović, 1990) was developed with a purpose of performing rapid real-time control and fault diagnosis in urban drainage systems. One application of this work is in successful practical use even today (Amdisen et al, 1994).

### 3.3.3. Non-deliberative architectures

Non-deliberative architectures stem from a well-known observation by Simon (1963) that a complex behaviour of an agent may not be a reflection of its own complexity, but rather an evidence of the complexity of the environment in which it is situated. Such systems are often said to be *situated*- reactive- or behaviour-based. Examples of non-deliberative architectures are Brooks' *subsumption architecture* (1991), Rosenschein and Kaelbling's *situated automata* (see Kaelbling, 1987), *reactive action packages* (RAPs) (Firby, 1987) and Maes' (1990) *dynamic action selection*.

All these architectures have a common objective in analysing complex behaviour as a consequence of a relatively small number of simple behavioural rules. Thus it can be concluded that non-deliberative architectures have a strong inclination towards studies of emergent computation.

### 3.3.4. Hybrid architectures

Both approaches presented so far - deliberative and non-deliberative - may exhibit specific drawbacks under certain conditions. In general, deliberative architectures are regarded as time-consuming and extremely difficult to implement under real-time constraints. Moreover, the fundamental assumptions of complete informedness of the agents sometimes appear unreasonable. Thus, deliberative-agent systems are quite fragile in real-world situations. Non-deliberative architectures, on the other side, are often rather robust, but, should the environment in which non-deliberative agents are situated diverge enough from the originally intended one, the agent may end up producing a quite unfeasible behaviour. While such an agent might be robust in the sense that it to avoids harmful or fatal behaviour, it is not clear and can in no way be guaranteed that a non-deliberative agent is sufficiently capable of carrying out its proposed long-term objectives.
To overcome these deficiencies, several *hybrid* architectures have recently been proposed. These architectures have the objective of blending the most useful properties of both approaches within one unified framework. One ‘early’ example is provided in work by Georgeff *et al.* (1987) who proposed a *Procedural Reasoning System*. Cohen *et al.* ran the *Phoenix project* (Cohen *et al.* 1989; Howe and Cohen, 1990) where the objective became an investigation of a real-time *behavioural ecology*. This work addressed the functional relationships among agents, their designs, the environments in which they operate and the resulting behaviours. The latest example of a hybrid system approach that is presented here is the so-called *Adaptive Intelligent System* (AIS) (Hayes-Roth, 1995), as has already been introduced briefly earlier in this work.
4. Emergence

Intuitively, emergence is a notion used to describe the appearance of new structures and properties. Emergence is intimately associated with interacting entities when the result of the interactions can be said to emerge on the shoulders of these interactions. In order to define the notion of emergence more precisely, the primitive entities, through the dynamics of interaction of which emergent properties appear, must first be defined. At the same time, in order to keep a record of this new quality, a class of artifices for identifying and scrutinising emergent properties must be introduced as well.

4.1. Phenomenological definitions

Forrest (1991c) explicitly set down the requirements for emergent computation, basing it on the requirements proposed by Hofstadter’s (1982) work, situated within the realms of sub-cognition. Hofstadter accentuated the absence of certain information at ‘lower’ levels of cognition which can exist at ‘higher’ levels as a consequence of the collective activities of these ‘lower’ sub-cognitive processes. Following this proposition, Forrest (1991c) defined the necessary constituents of an emergent computation, as follows:
(i) A collection of agents, each following explicit instructions;

(ii) Interactions among the agents (according to the instructions), which form implicit global patterns at the macroscopic level, i.e. *epiphenomena*;

(iii) A natural interpretation of the epiphenomena as computations.

It has been stated that each agent must follow its instructions explicitly. These are, in fact, collections of primitive, atomistic rules assigned to each individual entity (agent). These primitive rules, are, however, of only secondary interest within the emergent-computational framework. It is the epiphenomena that are the purpose and the main motivation for any such 'societal' approach to computation. Epiphenomena are intuitively seen to be situated at a 'higher' level than the primitive, atomistic rules.

One of the predicaments of emergent computation is the establishment of the way and the methodology to capture the emergent properties. Quite obviously, the results of emergent computation are only implicit in the atomistic rules and there is no obvious output location where they can be accessed. Of course in certain cases the emergent computation is accomplished with a certain class of emergent phenomena in mind, and the methodology for approaching these can be explained accordingly. But, in general, the design of a 'perceptory apparatus' capable of interpreting an entire range of results of emergent computations is far from being a trivial task.

Why emergent computation? This question must be addressed with great care. After all, the Church-Turing thesis postulates that a sequential Turing machine can implement any definable computation and execute any set of explicit instructions. 'Thus, the concept of emergent computation cannot contribute magical computational properties' (Forrest, 1991c). The question concerning the fundamental difference in the way the problems are solved in explicit and emergent computations necessitates a definitive answer.

In certain cases, in fact, explicit computation offers the most definitive advantages. These are the cases in which macroscopic processes can be assumed to be perfectly-ordered, mechanistic processes. There is a long tradition in physics, for example, of studying systems in ways which control complexity. Firstly, a simple, archetypal problem is identified and basic mechanisms identified and explained, and then the complexity of any one problem under consideration is increased. For example, in the case of dynamics, the physicist can start out by studying of two point-bodies in a Newtonian mechanical universe, describing the elementary rules for the interaction of these two bodies, and then shifting the focus of the problem to say, celestial mechanics or condensed matter physics. The techniques of statistical mechanics applied to studies of the non-linear dynamics of systems with many degrees of freedom, and for corresponding computer simulations, are well established methodological procedures for advancing an analysis. However, such, fundamentally statistical, averaging

---

22 An ideal gas is an example of a system in which the abstraction of the individual elements and their interactions makes tractable the derivation of global properties. From the idealised properties of molecules in such a gas, macroscopic properties are derived which describe real gases within certain
applies for a narrow range of problems within rather confined limits of application.

This, Kepleristic and Newtonian, idealism holds for a certain class of problems, but the most intriguing, difficult and probably most important problems that currently confront us do not conform to these conditions. Most of the current problems are complex, chaotic and can not (or at least, until now were not) solved by the application of explicit computations. Over the past two decades, a different approach to complex systems modelling has originated. This is an approach which builds upon the large base of ideal models of (physical, ecological, psychological) systems, orientated towards the analysis of the global properties that result from the sum of many simple interacting units. If we use ecology, just for the moment, as an example, we observe that, traditionally, ecological systems have been modelled in a certain centralised fashion by means of ordinary or partial-differential equations that are solved and applied to the this system, as if they were the beginning and the end of all understanding. All presently available indicators, however, show a deteriorating capacity of our planet to sustain its current rate of increase of industrialisation and more ‘civilised’ human life. Ecology, of course, is not the only discipline to blame (if ecology can be said to be blamed at all), but the question is whether current ecological modelling practice is able to help develop the knowledge necessary to allow us to interact responsibly with our environment. Clearly, the classical, centralised modelling approaches are ‘failing to produce those adequate environments of large-scale, unbounded and interconnected complexity that exists as a fundamental property in the world’s ecosystems’ (Jørgensen, et al., 1992). All the fundamental reasons for the failure of centralised modelling basically in this area boil down to one: ecosystems are composed of a large number of individual, complex entities (i.e. agents), each of which acts on its own accord with its own set of goals and objectives, and these have really nothing at all in common with the differential equations, which we sometimes use to model them. These, essentially non-Newtonian agents live and die, reproduce, fight and cooperate, learn and forget, and, obviously, differential equations are a poor and non-expressive apparatus for capturing all these features. Of course, models based on differential equations describe with a beautiful simplicity certain singular and isolated aspects of the ecosystems’ dynamics, but the scaling of these descriptions from any such singular event to the real-world dynamics of large ensembles is where the real problems start. The agent-orientated approach to ecological modelling will be described in considerable detail later in this work.

Through the promotion of emergent computation as a paradigmatic approach to modelling, we are, de facto, advocating an alternative and more natural (although often rather radical) way of thinking. For her part, Forrest (1991c) emphasised:

limits. For these models, the techniques of statistical mechanics, in which a description of the typical behaviour and interactions of a system’s elements is used to determine overall behaviour, prove invaluable. The macroscopic properties of system with degrees of freedom of the order of Avogadro’s number can be derived if the interactions are simple enough.

23 The way of thinking! This is the key phrase. Through centuries and, for that matter, even millennia, we, Homo sapiens, have observed the world around us in ways that are largely, and even primarily dependent on our own, social situations. Most of these perspectives are deeply built in our own cognitive apparatus and shape our overall way of thinking. The following few excerpts from
"First, implementing computations directly as emergent patterns may provide implementation efficiencies because of the need for less control over the different components. Second, flexibility is important for systems that must interact with complex and dynamic environments, e.g. intelligent systems. For these systems, it is impossible to get enough flexibility from explicit instructions; for realistic environments it is just not possible to program for all contingencies ahead of time. Therefore, the flexibility must appear at the emergent level. Third, the advantage in representation arises in systems for which it is difficult to articulate a formal description at the emergent level. Finally, the grounding issue arises if the emergent phenomena are intended as real phenomena or models of real phenomena (as in cognitive modelling). In this circumstance, the intended interpretation of a purely formal model (e.g. symbolic models of artificial intelligence) becomes problematic since the model is not connected to (grounded in) the domain of interest (by, for example, a sensory interface). Emergent-computation models can address this problem by using low-level explicit instructions that are directly connected to the domain."

Searle (1992), on the other hand, defines emergent properties of a system in the following way:

Abbott (1995, unpublished manuscript) should help in setting this scene:

"There is nothing we know that is more clearly and self-evidently permanent than the starry sky. We have only to lift our eyes on clear night to behold a sight that is essentially indistinguishable from that seen by Kepler, by Aristotle or, for that matter, by Neanderthal man... The celestial process that we so easily observe is, to all intents and purposes, the whole process: no other process intervenes to influence its course. The celestial motions constitute the most perfect examples, the very prototypes of completely-macroscopic and perfectly-ordered mechanical processes, with a corresponding permanence or persistence of motion that cannot be equalled by any mechanical system on the Earth itself.

Indeed, it was just because of their perfection that these motion were described as celestial, and heaven itself was placed metaphorically in the tracts of the starry skies, so that even into our own century we could speak, with Poincaré, for example, of a Méchanique Céleste, or a celestial mechanics...

Astronomical predictability, as formulated by Kepler alongside the astrological, but generalised by Galileo and Newton within the modern-scientific paradigm, made possible the establishment of mechanics, as the first realisation of the programme of the European Enlightenment. It thereby, provided the French encyclopaedists with their grand example of the power of rationality and so served in its turn to fan the fires burning under the upper deck, while, through Kant and Hegel, it led to a break with the whole European tradition of Christian, including scholastic philosophy. It thus led, ultimately, to the subversion of social institutionalised religious belief itself. Although the solution of certain problems of celestial mechanics presented formidable analytic and computational difficulties - and some of its problems continue to pose such problems even today - the laws of mechanics as enunciated by Newton and Euler continue to provide one of the surest and most firmly established branches of science."
"Suppose we have a system, \( S \), made up of elements, \( a, b, c \ldots \) In general, there will be features of \( S \) that are not, or not necessarily, features of \( a, b, c \ldots \) Let us call such features 'system features'... Some system features can be deduced or figured out or calculated from the features of \( a, b, c \ldots \) just from the way these are composed and arranged (and sometimes from their relations to the rest of the environment). But some other system features cannot be figured out just from the composition of the elements and environmental relations; they have to be explained in terms of the causal interactions among elements. Let's call these 'causally emergent system features'."

In order to further distinguish these features from those of 'casually emergent system phenomena' Searle describes a second type of emergent property:

"[The above definition] of causal emergence, call it 'emergent1', has to be distinguished from a much more adventurous conception, call it 'emergent2'. A feature \( F \) is emergent 2 iff \( F \) is emergent1 and \( F \) has causal powers that cannot be explained by the causal interactions of \( a, b, c \ldots \) If consciousness were emergent2 then consciousness could cause things that could not be explained by causal behaviour of the neurons."

4.2. Formal definitions

Thus, emergence can be viewed as a process of constructing a new structure from a collection of existing ones. Baas (1994), provides a general procedure for the construction of new structures on the basis of existing ones, as follows. Firstly, the family of existing structures is defined as:

\[
\mathcal{S}_i, \ i \in J
\]

Then a certain set of observational mechanisms, \( \mathcal{O}bs \), are introduced with the purpose of acquiring the structures:

\[
\mathcal{S}_i, \ \mathcal{O}bs(\mathcal{S}_i)
\]

Finally, elements of \( \mathcal{S} \) are subject to certain interactions which are collectively referred to as \( \mathcal{I}nt \). The new structure \( S \) is then defined by the triplet:

\[
S=R(\mathcal{S}_i, \mathcal{O}bs(\mathcal{S}_i), \mathcal{I}nt)_{i \in J}
\]

where \( R \) stands for the result of the construction process. Baas (1993) refers to \( S \) as a second-order structure, as opposed to the elements of \( \mathcal{S} \) which are considered to be first-order structures, the primitives of the theory. The definitions above can be naturally extended to allow for third- and any higher-order structures. For the sake of clearer nomenclature, first-order structures will be referred to as \( \mathcal{S}^1 \), whereas a family of first-order structures will be referenced as \( \mathcal{S} \). Similarly, all possible second order structures, defined as
are elements of the family $S_i$, $i_2 \in J$.

Given such an apparatus, Baas (1993) conveniently defines emergence as:

$P$ is an emergent property of $S^i$ iff $P \in \text{Obs}(S^i)$; but if further $P \notin \text{Obs}(S^i)$, we call $S^i$ an emergent structure, and the process is called first-order emergence.

Further to this again, two different manifestations of emergence can be distinguished:

A. **Deducible or computable emergence** means that there is a deductional or computational process or theory $D$ such that $P \in \text{Obs}(S^i)$ can be determined by $D$ from $(S, \text{Obs}(S), \text{Int})$.

B. **Observational emergence** arises if $P$ is an emergent property but cannot be deduced as in A.

Working along these lines, many different phenomena of interest have been observed. A variety of interesting problems arises in the biological and social disciplines, and in computational studies in general. For example, the problem can be posed of how memory and learning can emerge from a collection of 'neurons' (McCulloch and Pitts, 1943; Hopfield, 1982), or, in a somewhat different vein, how these dynamical systems can be studied as collection of agents according to a set of rules in the game-theoretical sense (Von Neumann and Morgenstern, 1944; see also, in this respect, Maynard Smith, 1982).

In many computational, economic and biological systems, the comprising entities or agents have additional, and in no-way Newtonian, properties. The agents may form plans or be programmed to plan ahead. As a result, their actions appear to be (or are) intentional. For example, beliefs about the future can be programmed into the design of computational agents or evolve genetically in biological (and some non-biological) ones. Alternatively, social agents can form expectations because of their intentional inclinations. The property of intentionality is absent in the objects that are within the purview of physics and chemistry, by and for themselves, but it applies to that which creates the object, as something towards which a thought (or a fragment of a code) is directed.

The agents make plans to achieve their goals based on their knowledge of causes, their memories of the past and their predictions for the future, as well as on their biases and their beliefs. Such agents may also be capable of learning, possibly by adopting strategies that enhance their performance. Or they may simply appear to learn if the agent types evolve over generations according to some fitness function. Along these lines, a computational analogue of genetic evolution called genetic algorithms (Holland, 1975) has become a common general purpose search technique in which a population of solution 'agents' are bred until a satisfactory solution is reached. In this way, through learning and planning, agents can to some extent adapt to changing environments made up of other agents and external events.

Integrating several ideas about emergence within an AI framework, Angeline (1993) proposed a notion of **Emergent Intelligence** (EI):
"Emergent intelligence (EI) is any problem-solving algorithm that uses an integral component of a problem-solving process such that the amount of explicit knowledge required to implement the same problem-solving activity is reduced. The intention is that appropriate knowledge emerges at the knowledge level as an epiphenomenon of the problem-solving process. Rather than providing all the necessary knowledge as explicit knowledge to the problem solver, EI algorithms allow knowledge that is pertinent to the specific problem-solving instance to emerge from the natural dynamics of the problem solver interacting with the task environment."

A dynamical formulation of agent interactions requires a different approach when the behaviour of the entities depends on their beliefs and knowledge as well as the rules governing their interactions. The agent’s predictions for the future must enter explicitly into the dynamic rules governing the evolution of the system. In general, an agents’ expectations of the future can be recast as a prescription of how to act in the present based on imperfect present and past knowledge in a way that takes the beliefs of the agent into account implicitly.

The present work is a study of the performance of a collection of agents and is an exploratory one. One part of it is thus inevitably phenomenological, and so essentially pre-scientific, but hopefully it is also to some extent scientific in the attempts that it makes to recognise and categorise certain phenomena and utilise them in the design of useful artifacts.

This work could indeed well be concluded, at this point, for the essence of the most important ideas has now been presented. The rest of the work merely demonstrates the way in which these ideas are realised, and themselves emerge fully, once they have been instantiated in code.
5. Searching for Solutions

The objective of artificial intelligence is to create machines that can act in complex environments in a way that is similar to that attributed to humans, or even better. Thus far, the field has produced many useful artifacts with various rudimentary traces of intelligent behaviour. However, it is generally accepted that this ‘ultimate goal’ of AI remains distant, and indeed almost certainly unattainable in its full generality.

Probably the most original notion within the entire field of AI can be attributed to John McCarthy (1958). He proposed a system (he called it Advice-Taker) that would decide how to act through applying formal reasoning procedures (inferencing) over a body of explicitly-represented knowledge. Such a system would not be so much explicitly programmed to do something as much as it would be provided with the necessary pieces of knowledge, and on the basis of these it should infer the rest. This approach explicitly distinguishes between what the system has been told and what the system infers. Thus, the entire computational problem is defined by a formal language, which is a declarative framework for expressing knowledge, a knowledge base, which is a collection of symbolic structures expressed in a formal language, and an inference engine, which is a collection of specifications as to what should be extracted from the semantics of a formal language, given a knowledge-base as input.
Following this notion, an entire category of problem-solving software emerged: *production systems*. Production systems consist of (Nilsson, 1980):

- a *global database*
- a collection of *production rules*, and
- a *control system*.

Production rules consist of precondition and action parts. When the preconditions are matched against some aspects of the global database, the rule may be applied, thus altering the content of the global database. The control system decides which production rules to apply in which situation, applies conflict resolution strategies and terminates the inference process when appropriate conditions are met.

Generalising the properties of production systems even further within the framework of an entirely symbolic artificial intelligence, two central directions can be recognised: *knowledge representation* and *search methodology*. These two research directions are even today often regarded as the most rudimentary, but they are still the *only* two constitutive elements in AI (Ginsberg, 1993, Angeline 1993). Ginsberg (1993, p.38) pronounces, very concretely:

"The intended role of knowledge representation in artificial intelligence is to reduce problems of intelligent action to search problems."

The process of the embodiment of the problem domain within the knowledge representation formalism can be equated to the formation of a set of axioms that are formulated with respect to a given problem domain. Similarly, the application of a knowledge representation formalism on a problem domain can be represented in the visual form of a *graph*, which can be explored, albeit using other sign vehicles, by employing a suitable search algorithm.

The very first AI programs were founded on heuristic search, such as the chess programs of Shannon (1950), *Logic Theorist* of Shaw and Simon (1956), Samuel’s checkers program (1959) and Slagler’s symbolic integrator of 1961 (Feigenbaum and Feldman, 1963). All these domains typify the notion of intelligence that underlies this particular paradigm and all are well suited to be solved by procedures that apply search techniques.

Applying the *rules of inference* which specify which conclusion can legitimately be deduced from existing axioms, can indeed be understood as a *search* problem.25

---

24 Minsky (1961) divided the AI problems into five main areas: search, pattern recognition, learning, planning and induction. Out of these five sub-domains pattern recognition, learning and planning are said to be direct descendants of search and knowledge-representation.

25 Leibnitz was the first proponent of Artificial Intelligence in this sense. In the future, he predicted, if a dispute arose between two philosophers, they would reach for pencils and paper and say *calculemus* (let us calculate). Even his invention of the first calculator with multiplication and division and his associated research into binary notations were motivated by his dream of 'automating philosophy'. The situation of search at the core of Artificial Intelligence is perfectly in accordance with his views. The whole process is implicit however in Aristotelian logic, and the first to see the resulting
The one, most general rule of inference is so-called *modus ponens*, which simply states that "If a, then b". Ergo, if a holds, it is legal to conclude that b holds as well. This can be written as:

\[
a \rightarrow b
\]

\[
a
\]

\[
b
\]

The similar version, *modus tolens*, performs the inference in the reverse direction, thus given that b holds, it is legal to conclude that a holds as well. Similarly, this can be written as:

\[
a \rightarrow b
\]

\[
b
\]

\[
a
\]

There are many possible ways to apply modus ponens, modus tolens, and rules of inference in general, in a given problem domain. Since there are so many of these possibilities, the choice of which one to select is solved by *searching through the space of solutions*. Thus, from this point of view, the *deduction of a solution is regarded as a search problem* (Ginsberg, 1993, p. 39). The initial information (or starting vertex in a graph) is a point in a search space (graph) from which the search process is initiated.

The effectiveness of the exploration of this space can than be equated with the performance of a search technique. Systematic, non-systematic, exhaustive, non-exhaustive, random search, hill-climbing, simulated annealing, evolutionary algorithms and many other techniques can all be easily depicted as methods that perform a ‘walk’ on a landscape that is composed of solution elements.

Correspondingly, every problem-solving activity can be regarded as corresponding to a task of searching for, or constructing an object with required characteristics. The most rudimentary computer-program that performs this problem-solving must, in turn, encompass the following sub-processes (Pearl, 1984):

- provide a *symbol structure* or *code of representation formalism* which can represent any admissible candidate object in the search space;
- provide certain computational *procedures* that are capable of *transforming* the encoding of one object to that of another in order to inspect the candidate object;
- provide some *scheduling* method which arranges these transformation so as to generate

misconceptions in Aristotle’s work and to explicate their restrictions and divisive consequences was Nicholas de Cusa, in the late XIII century. However, it is better to differ the discussion of the pivotal role of Nicholas’ thought to another place.
the desired object as quickly as possible, and thus provide an effective control strategy.

A systematic control strategy is one which satisfies the completeness requirement\textsuperscript{26}, the non-redundancy requirement\textsuperscript{27} and the informedness requirement\textsuperscript{28}. Such a control strategy requires a certain memory (to store information on already-visited nodes - pointers) and the cost (as a measure of computational effort) necessary to establish which node to visit next. Both memory and cost are crucial constituents to be discussed with respect to the efficiency of an algorithm at the meta-level. The algorithms we outline in the sequel are discussed in terms of the properties just mentioned and with a particular regard for algorithmic efficiency at the meta-level.

![Figure 17 An overview of search algorithms](image)

Search algorithms can be decomposed into at least five different constitutive parts. The first one is problem state representation, the role of which is to code for entire solutions of the elements of complete solutions using some knowledge representation formalism. The second part defines search operators that extend or alter partial solutions. Credit assignment estimates a 'cost' of complete or partial solutions. The control structure of the algorithm defines an order in which search operators will be applied - such as breadth-first, or depth-first. Finally, there is a termination criterion, which defines a degree of satisfaction with the solution proposed by algorithm.

5.1. Graph theoretic notation

As has been outlined earlier, search in the most abstract form is performed on a graph. It will thus be advantageous to introduce some more formal graph-theoretic notation at the outset of this section.

\textsuperscript{26} The completeness requirement states that, given an encoding that can represent a method for generating a desired solution, a search algorithm can reach this desired solution.

\textsuperscript{27} The non-redundancy requirement requires that repetitious computations will not be necessary.

\textsuperscript{28} The informedness requirement insists that the search technique uses the possibility-limiting information in order to restrict the way in which solutions are generated.
A graph is a pair of sets $A$, $V$ and a pair of mappings $\phi$, $\psi$ that connects the two sets. The upper mapping $\phi$ corresponds to the function defining a source vertex, and the lower mapping $\psi$ defines a target vertex; thus $A \xrightarrow{\phi} V$. It will later become apparent that the 'upper arrow' in principle can be equated with the application of search operators, whereas the 'lower arrow' is equated with mechanisms that eventually occupy the memory within the algorithm and keep track on already visited nodes.

5.1.1. Graphs or trees?

It is often useful, in the graph-searching context, to introduce the notion of the tree, $T$. A tree is a graph in which each node has only one parent node. A number of methods exist which generate trees of a required properties from a given landscape $L$ or a graph $P$ (Chen, 1990; see also, with another particular reference to hydroinformatics, Kutija, 1995).

![Figure 18](image)

**Figure 18** A graph encoding for all possible configurations between target configurations 0000 and 1111

In the following chapters we shall perform searches on trees rather than on more general graphs, so that it will be impossible to reach the same node using each of several paths from the root node. This limitation is only introduced for convenience of exposition, however, and in fact it will not be valid in many practical cases, while, by exploring the properties of
graphs while assuming that they are trees, we may obviously come in such cases to violate the non-redundancy requirement. A quick reference to figures 18 and 19 will suffice to indicate the degree of redundancy introduced through the representation of a problem as a tree instead of as a more general graph. In order to maintain the algorithm's performance at a high level and to avoid repetitious calculations, a transformation of graphs into trees must none the less be essayed.

One way of doing this is to maintain a list of already visited nodes and simply avoid visiting these nodes in the subsequent computations. This list is often referred to as list of closed nodes. The other, non-visited, vertices in the graph are referred to as open nodes. Searching the graph would then involve determining whether the node belonged to the open or the closed list, and, depending on the outcome of this test, one could either expand this node or not. Such a test, clearly introduces a certain additional cost into the use of the search algorithm, and consequently reduces its performance.

Recently, Ginsberg (1993) has proposed a novel methodology called dynamic backtracking to explore, or search, a graph as a search tree. The fundamental idea is that when backtracking from a non-successful node, a different path should be used from the one used to generate the node. Dynamic backtracking is intended to be a systematic, as opposed to a non-systematic search method. This systematicity requirement, however, introduces in its turn quite a heavy memory requirement and indeed one that is of the order of magnitude of $O(bd^2)$, where $b$ denotes the branching degree and $d$ the depth in the graph. For fuller and a more detailed information, reference should be made to Ginsberg (1993, p.64).

5.1.2. Searching a graph

A sequence of vertices $v_1, v_2, ..., v_k$ where each $v_i \in V$ is an offspring of the $v_{i,j} \left[ v_i = N(v_{i,j}) \right]$, $\forall i \in \{1, 2, 3, ..., k\}$ is called a path, of a length $k$, from node $v_i$ to node $v_k$. The cost of a such a path is normally understood to be equal to the sum of the weights of the arcs traversed as induced by $\phi$. In some special cases, it might prove useful to base the calculation of the cost on the average weight, or as the maximum of all the arc costs along the path.

Pointers are normally set up from each generated node back to its ancestor. These pointers form a network of paths which facilitates the mechanism of tracing back the explored pathways. The pointers collectively form a memory of a search process.

A search procedure, or policy, is a recipe for determining the order in which nodes are to be generated. One can distinguish between blind or uninformed search and informed, guided or directed search. In the former, the order in which the nodes are expanded depends only on information gathered by the search, but is not affected by the character of the unexplored portion of the of the graph, and not even by the goal criterion. The latter uses partial information about the problem domain and about the nature of the goal to help guide the search towards the more promising regions.
Naturally, the set of vertices in the graph that is being searched can be decomposed, at any time, into four sub-sets:

- nodes that have been expanded,
- nodes that have been explored but not yet expanded,
- nodes that have been generated but not yet explored,
- nodes that are still not generated.

A search algorithm on such a graph, or in general a problem-solving algorithm’s action, can be equated with the *reachability* problem, which can be posed as follows:

*Given a graph, $\Gamma(\mathcal{A} \rightarrow \mathcal{V})$, and two vertices $x, v \in \mathcal{V}$, determine whether there exist a path, $x \rightarrow \rightarrow \ldots \rightarrow v$.*

A problem-solving algorithm is the one that characterises these paths in one of the following three ways:

$$R = \{ v \in \mathcal{V} \mid x \rightarrow \rightarrow \ldots \rightarrow v \} \quad i.e. \ v \ is \ reachable \ from \ x$$

$$R = \{ v \in \mathcal{V} \mid x \rightarrow \rightarrow \ldots \rightarrow v \mid m \leq n \} \quad i.e. \ v \ is \ reachable \ from \ x \ in \ at \ most \ n \ steps$$

$$R = \{ v \in \mathcal{V} \mid x \rightarrow \rightarrow \ldots \rightarrow v \mid v \} \quad i.e. \ v \ is \ reachable \ from \ x \ in \ exactly \ m \ steps$$

The operators that expand the vertices are in effect those that apply the rules of inference to a search space in order to solve the problem. With generate-and-test, according to definition, it is the goal state that is fixed; thus search progresses in some *backwards* fashion, from a goal state to its preconditions (axioms). In principle, there would be no indispensable difficulties in supplying the initial state and proceeding towards the goal in a *forward* fashion, similarly to applying modus ponens. Which of the two directions to choose is a meta-level decision, and the decision for the more successful one will depend on such properties of the fitness landscape as the branching factors $\mathcal{R}(v) \forall v \in \mathcal{V}$ in the related directions.
The process of the exploration of the graphs and in particular of the special category of graphs that we shall soon identify as fitness landscapes, whether by evolutionary or by any other means, is one of the central issues of the present work. Therefore, in the continuation, some time is spent on the analysis of the performance of certain search algorithms.

5.1.3. Optimisation or constraint-satisfaction

One more aspect of the search must be accentuated. The search can be performed in order to locate many different things, thus satisfying many different kinds of requests. In the optimisation view of the search problem, the objective is not only to locate a formal object satisfying an established set of criteria, but also to ascertain that it possesses qualities unmatched by any other object in the search space.

Another view on graph exploration is the one that is often described by the neologism of satificing (originally coined by H. Simon, 1956) which is otherwise called 'constraint satisfaction'. In this case, what is desired is to discover any qualified object with as little search effort as possible. Thus, the search can be for the least expensive path between a starting configuration, \( x \) and a certain end configuration that satisfies the requirements, minimising the number of steps \( m \) in the search procedure, or minimising (maximising) the sum of the 'weights' encountered on the arcs along the path, or searching for the least average weight along the paths, or whatever else may be required. There can indeed be a large number of different measures of success. This situation can be expressed by writing:

\[
R = \{ v \in V \mid x \rightarrow_0 x \rightarrow_1 y \rightarrow_2 z \rightarrow_3 \ldots \rightarrow_m v \}
\]

with the objective to minimise \( m \).

Most problems can be posed in terms of both constraint-satisfying and optimisation tasks. The real-world problems are generally very difficult, since in most of the cases encountered in practise it is not known if any 'zero-cost' solution (the unique optimal solution) exists. Even if it can be proved that such a solution does not exist, a large number of more expensive configurations must be examined before some other optimal solution is established.

Heuristic\(^{29}\) information pertaining to the quality of the solution may be helpful in reducing the effort involved in finding such a solution, even in those cases in which the quality of the solution is of minor importance. Often, by focusing the search effort on finding the shortest path, it is possible to insure against useless explorations in the large space of possibilities. This small-is-quick principle plays a major role in heuristics for 'satisficing' search.

Often the difference in complexity between an optimisation and its 'satisficing' is very substantial. The so-called NP-hard (Garey and Johnson, 1979) problems are good examples of this. To find some solution that satisfies the constraints is usually almost trivial, but to

\(^{29}\) The word heuristic comes from the Greek heurikein, meaning 'to discover', which is also the origin of eureka, best known from Archimedes' reputed exclamation, heurika (for 'I have found'), uttered when he had discovered a method for determining the purity of gold.
locate a truly optimal solution is often extremely demanding. In such cases, it is a good heuristics to relax the optimality requirement and trade it off with a reasonably good solution, using only a reasonable search effort. Whenever the acceptance criterion tolerates a neighbourhood about the optimal solution, we refer to such an approach as one of semi-optimisation, or occasionally, and less correctly, as quasi-optimisation.

Semi-optimisation problems fall into one of two categories. If the boundaries of the acceptance neighbourhood are sharply defined (e.g., requiring that the cost of the accepted solution is within a specified fraction of the optimal cost), the task is referred to as near-optimisation. If we further relax the acceptance criterion and insist only that the solution be near optimal with a sufficiently high probability, we refer to the resulting type of problem as approximate-optimisation.

Most practical problems are of a semi-optimisation nature, requiring some trade-off between the quality of the solution found and the cost of searching for such a solution. The basic problem in handling a semi-optimisation task is to devise an algorithm that guarantees bounds on both the search effort and the extent to which the optimisation objective is compromised. A more ambitious task would be to equip the algorithm with a set of adjustable parameters, so that the trade-off between the quality of the solution and the search effort can evolve towards the most acceptable configurations.

5.2. Weak and strong search algorithms

In general, problem-solving strategies in artificial intelligence are divided into strong and weak search methods. Weak methods do not make any assumptions about the domain they are transversing. Strong methods do. Consequently, since the strong methods utilise the auxiliary information provided through application of the assumptions in order to improve the efficiency of the search procedure, they usually outperform the weaker methods in many classes of practical problems.

However, strong methods can be successfully applied only to classes of problems where the assumptions can be justified and conveniently introduced. Since weak methods can in principle assume almost nothing, they have in principle an almost universal applicability.

Weak methods are not primarily intended to be used directly as solution methods, but rather as foundations for establishing stronger methods (Rich, 1984, p.55):

"[The] efficiency [of weak methods] when applied to a particular domain is often highly dependent on the way they exploit domain-specific knowledge since in and of themselves they are unable to overcome the combinatorial explosion to which search processes are so vulnerable."

Problem-solving techniques with wide or universal applicability are thus referred to as 'weak' methods; the reason is that their universality typically makes them less effective than approaches that have been carefully designed to solve specific problems. Many operational AI systems developed with a specific problem-domain in mind use special-purpose algorithms that, although useful only in these narrow domains, do have a significant impact in these domains.
Searching for Solutions

There is something more profound stemming from this distinction between weak and strong methods. There is a trade-off in problem-solving between spending time actually solving the problem and spending time deciding how to solve the problem. The effort expended in actually trying to solve the problem is referred to as a base-level activity, whereas the work spend establishing what to do is called a meta-level activity. Since time spent on the meta-level cannot be spent on the base level, and since the system has no real interest in the meta-level effort other than in helping it decide how to solve the base-level problem, the general form of the above observation can be re-stated as follows (Ginsberg, 1993):

"In efficient problem-solving, time spent at the meta level is recovered by corresponding reductions in the amount of time required to solve the problem at the base level."

This is referred to as the base-level/meta-level trade-off.

There appears to be a large qualitative difference between human problem-solving and machine-based problem-solving. Humans are usually extremely good at making meta-level
decisions, like choosing the direction of a search. Machines, on the other hand, usually have tremendous difficulties in doing this. We have not yet found a way to transfer our own, and even for us, extraordinary capabilities to this machine of our own creation, the computer.

Probably, most of the 'blame' here has to be placed on differences in the nature of problem-solving between humans and computers. Problem-solving in humans is rather non-deterministic and, if only for this reason, very efficient at the meta-level. It appears that we possess 'mental-computational' qualities that simply rule-out those 'expensive' portions of the search space that have small probabilities of hosting the solution. This non-determinism is natural to humans, but alien to computers. And, as has now been shown in many case studies and experiments, computers do outperform human problem-solving in cases where meta-level decisions are not important, whereby a 'brute-force' search applied directly on the problem domain can be the more efficient.

5.3. Non-informed search

Non-informed search here implies a category of search algorithms which do not utilise feedback from the search space in order to guide future steps. Such an approach results in robust techniques that cannot get trapped in local optima.

5.3.1. Non-deterministic, non-systematic search

Non-systematic search characterises a category of search algorithms that do not keep track of already visited points, thus these are algorithms that do not require memory. This apparent advantage introduces the possibility of a redundant search: thus a single point might be visited on several occasions during the search procedure.

5.3.1.1 Generate-and-Test

Generate-and-test (Newell and Simon, 1972) is at one and the same time the weakest and the most general possible search method and all other search methods can be considered as specialisations of this method. These authors introduced their technique within the context of General Problem Solving (GPS) in the following fashion:

"The main methods of GPS jointly embody the heuristic of means-ends analysis. Means-ends analysis is typified by the following kind of common-sense argument:

I want to take my son to nursery school. What's the difference between what I have and what I want? One of distance. What changes distance? My automobile. My automobile won't work. What is needed to make it work? A new battery. What has new batteries? An auto repair shop. I want the repair shop to put in a new battery; but the shop doesn't know I need one. What is the difficulty? One of communication. What allows communication? A telephone ... and so on."

A strongly similar argumentation was used by Aristotle more than 6 centuries BC (see also Norvig, 1992, page 111). In the Nicomachean Ethics (Book III.3, 1112 BC) and in particular
in the chapter entitled *The nature of deliberation and its objects*, Aristotle debates in the following persuasive manner:

"We deliberate not about ends, but about means. For a doctor does not deliberate whether he shall heal, nor an orator whether he shall persuade, nor a statesman whether he shall produce law and order, nor does any one else deliberate about his end. They assume the end and consider how and by what means it is attained; and if it seems to be produced by several means they consider by which it is most easily and best produced, while if it is achieved by one only they consider how it will be achieved by this and by what means this will be achieved, till they come to the first cause, which in the order of discovery is last ... and what is the last in the order of analysis seems to be the first in the order of becoming. And if we come on an impossibility, we give up the search, e.g., if we need money and this cannot be got; but if a thing appears possible we try to do it."

This is an interesting basis as a foundation of GPS and intelligent behaviour in general. Accordingly, the end object (goal state) is fixed and then some of the options are tried within an existing representational scheme until the object is composed, or all options exhausted and the NIL (empty) object is generated instead.

In the initial excitement of 1957, when the method so devised appeared to provide a grandiose vision - a single computer program that could solve any problem, given a suitable description (or representation) of the problem - Simon went so far as to make this statement about his creation:

"It is not my aim to surprise or shock you ... But the simplest way I can summarise is to say that there are now machines in the world that think, that learn and create. Moreover, their ability to do these things is going to increase rapidly until - in the visible future - the range of problems they can handle will be coextensive with the range to which the human mind has been applied."

Almost forty years later, it is clear that GPS has not fulfilled these claims. This failure is due to many reasons, some of which will be explicated in the sequel, but we may immediately emphasise the combinatorial explosion in the solution space which results in such a computational effort at the domain level that it degrades the GPS’s performance to a totally unacceptable level.

A random search procedure, such as generate-and-test, generates a single element with a random probability. It assumes only the completeness principle and does not satisfy the other two. At the meta level, generate-and-test is very inexpensive. It posses no memory, so that search can be redundant, and, since the offspring nodes are created randomly, the computational cost can be kept rather low. Generate-and-test is the most basic of all search strategies; therefore its results are often used as a basis of comparison with other search techniques. In order to improve upon this most basic technique, we must first improve the performance of the function which generates the new objects.
procedure Generate_and_Test;
begin
  object := Generate();
  while not (Test (object))
    object := Generate();
  return object;
end.

Figure 21 Pseudo-Code for the Generate-and-Test Weak Search Method

The test that relates the target configuration and the generated object is rather simple. For constraint satisfaction problems it involves testing the satisficing. In cases of optimisation, it often requires a transformation of an object into a more appropriate representation for evaluation, using credit assignment (i.e. a means to establish the value of a fitness).

Figure 22 Breadth-first search

5.3.2. Deterministic, systematic search
5.3.2.1. Breadth-first search

Breadth-first search is a blind, systematic search technique. It performs a search on a tree in the fashion that is schematised in Figure 22, examining all the nodes at a certain depth in the tree, and only then proceeding to the next level.

Such a search technique is not affected by specific properties (such as topological features).
of the search tree so that it guarantees success under all circumstances... but then, of course, subject to a high computational cost.

Should the branching degree, $B(v)$, be constant over the entire tree, say $B(v)=b$, and the goal node be located at a depth $d$, then the minimum memory requirement is of order of magnitude of $b^{d-1}$, since the algorithm must store in the memory all the already-visited nodes. Thus the memory requirement grows exponentially with respect to the depth of the search tree.

![Diagram of memory requirement for breadth-first search](image)

*Figure 23 Memory requirement for breadth-first search (adapted from Ginsberg, 1993)*

Computational cost can be estimated for such an idealised tree with a constant branching factor if one aggregates the number of all the nodes until reaching the fringe $d-1$ and adds to this number half the number of the nodes constrained within the deepest fringe, $d$. After some elementary summation algebra, this cost amounts to (see also Ginsberg, 1993):

$$\frac{b^{d+1}+b^d+b-3}{2(b-1)}$$

5.3.2.2. Depth-first search

Similarly to breadth-first search, depth-first is a blind, systematic search technique. Again, its
performance is best summarised by referring to an illustration, as in Figure 24.

It can be seen that the memory requirement grows linearly with a tree depth and is equal to $d(b-1)+1$. Clearly, the memory requirement is at its highest just one node before the examination of the goal node.

![Figure 24 Schematisation of a depth-first search algorithm](image)

The cost necessary to perform the computation equals to (for a full reference and cost computation, a reference may be made to Ginsberg, 1993):

$$\frac{b^d}{2(1+\frac{1}{b})}$$

In order to conclude, the depth-first approach is rather more expensive in computation, but much cheaper in terms of memory than is the breadth-first search technique. The performance of both techniques can be considerably improved by combining the best of the two approaches into a so-called ‘iterative deepening’ and ‘iterative broadening’ methodology (Korf, 1985b) that will be explained in the continuation.
5.3.2.3. Beam Search

One of the main disadvantages of the informed search techniques lies in the satisfaction of the non-redundancy requirement. Both depth-first and breadth-first search techniques have substantial memory requirements which effectively debar these approaches for the exploration of trees with extensive connectivities and large branching factors $B_v$. The problem is particularly apparent for breadth-first search. In order to avoid this, a variant of breadth-first was proposed (Lowerre and Reddy, 1980) in which the number of nodes to be expanded from a visited vertex is limited to a certain finite number, $c$.

Inside the loop, the algorithms continuously expand the nodes. The fitness values are then sorted and only the $c$ best-performing vertices are transformed into the queue that will be expanded in the next iteration.

Accordingly, beam-search sacrifices completeness in order to better prune the search space (tree). Nevertheless, it uses the cumulative information gathered as feedback from the search space through the continuous application of the evaluation function, and this feedback is not utilised with generate-and-test, breadth-first and depth-first search methods. Thus the traversal of the search space is, so to say, informed. The benefits in terms of reductions in both memory and computational requirements are obvious, and are also clearly dependent on the connectivity of the landscape and corresponding reductions in the branching factor $B(v) - c$.

In search spaces where the evaluation function is a good indicator of global optima with large basins of attraction, beam-search becomes an extremely efficient search method. The memory requirements for the beam-search are:
We see that, as compared to the breadth-first search, the 'only' difference is that the value of the branching factor $B(v)=b$ is replaced with $c$, a constant value of beam length.

\[
\frac{c^{d+1}+c^d+c-3}{2(c-1)}
\]

5.3.2.4. Iterative Deepening

Iterative deepening is a technique introduced by Korf (1985b), and represents an attempt to reduce memory requirements encountered with breadth-first search. Since the greater part of the cost involved in the systematic search is related to the calculation at the fringe, instead of 'memorising' all the nodes that have been visited, these are simply regenerated. The idea behind the technique stems from the observation that the analysis should be directed towards just those cases in which depth-first search performs poorly, namely, those where the depth of the goal node is less than the depth of the tree as a whole. This situation arises in those cases in which the tree is of infinite depth, or in the cases in which the first solution found is at a 'deeper' location than the shallowest solution. Iterative deepening assures that the goal depth and tree depth match.
procedure iterative_deeipinging
begin
  \( c = 1 \);
  while termination condition not satisfied
  perform depth-first search up to depth \( c \);
  \( c = c + 1 \);
end.

Figure 27 Pseudo code for iterative deepening

The idea, then, is to examine all the nodes up to a certain depth in one iteration. This \textit{cutoff} depth is increased by unity after the iteration and the process is repeated.

Figure 28 Schematisation of iterative deepening algorithm

Iterative deepening guarantees the location of the shallowest solution. The memory requirement is equal to that of depth-first search. The total number of nodes examined in this approach equals (Ginsberg, 1993):

\[
\frac{b^{d+2} + b^{d+1} + b^2d + b^2 - 4bd - 5b + 3d + 2}{2(b-1)^2}
\]

In the expression (X) above, the terms:
are the ones that contribute mostly to the memory size.

Similarly, in the pure breadth-first search the expression for cost is dominated by:

\[
\frac{b^{d+1}}{2(b-1)}
\]

The ratio of the two shows that iterative deepening is

\[
\frac{b+1}{b-1}
\]

'slower', or more costly to perform, than breadth-first. However, for larger branching factors, the performance of the two algorithms become almost identical. It appears that repetitions of the calculations at lower depths do not lower the performance of the algorithm.

In one very definitive sense, iterative deepening is an **optimal blind-search procedure**. For a fixed branching factor, \( B(v) = b \), it requires a memory of order of magnitude \( o(d) \) and cost \( o(b^d) \). Excluding luck and heuristics, it is impossible to do better. If an algorithm is expected to find a solution at depth \( d \), then the memory must be \( o(d) \). And since the goal configuration is located at the fringe, which is of the size \( b^d \), it cannot be expected that the blind algorithm can perform better than \( o(b^d) \).\(^{30}\)

### 5.3.2.5. Iterative broadening

Although iterative deepening is optimal in the sense that it is impossible to decrease the computational cost further than \( o(b^d) \), the performance of iterative broadening procedures often improve in practice, due to the manner in which the tree is examined.

---

\(^{30}\) Ginsberg sharpens the question even further (Ginsberg, 1993, p.60):

Consider any algorithm that takes time \( o(b^d) \) to run; how much space does it need in order to distinguish its own internal states?

If all the internal states of the algorithm are to remain distinct, it clearly needs at least as much space as it would need to count to \( b^d \). But counting to \( b^d \) needs \( \log_2(b^d) \) bits; since \( \log_2(b^d) = d \log_2(b) \) is \( o(d) \) for fixed \( b \), we see that any blind search algorithm needs space \( o(d) \) whether it is expected to return a path to the answer or not.
Similarly to iterative deepening, iterative broadening is viable when $c < R(v) = b$. The maximum amount of time is approximately $b^{d+1}/d$ for large $b$, and is approximately $b^d$ for large $d$. Iterative broadening is a useful technique because it concentrates attention on the exploration of the part of the search space that possibly contains the goal configuration, while scheduling non-promising portions of the search space last.

```plaintext
procedure iterative_broadening;
begin
  c=2;
  while termination condition not satisfied
    c=c+1;
  end
end
```

**Figure 29** Schematisation of iterative broadening algorithm

**Figure 30** Pseudo-code for iterative broadening
5.4. Heuristic Search

5.4.1. Deterministic Search

5.4.1.1. Hill Climbing

Hill-climbing is a search strategy based on local optimisation, and is one of the most popular techniques among human problem-solvers. It implies, similarly to the climber who wishes to reach the peak of a mountain by as short a path as possible, a choice of the direction of steepest ascent from the current position. Hill climbing is typically initiated from a random point in the search space and then proceeds in this, ‘greedy’, best-first fashion. In steepest ascent (descent), all the neighbouring points are examined and the most profitable one is selected for advancement.

Figure 31 Hill-climbing. On the left (a) standard hill-climbing, and on the right (b), hill climbing that makes use of macro operators (adapted from Ginsberg, 1993)

In terms of graph-searching, hill-climbing corresponds to a repetitious expansion of a vertex, inspection of newly generated offspring vertices, followed by choice and expansion of the best among the successors, while retaining no information on ancestors or siblings.

Such a computational simplicity implies numerous drawbacks. Firstly, due to the lack of memory the redundancy requirement is not guaranteed. Secondly, certain deceptive features of problems may attract the climbing procedure to the deep, and sometimes even infinite search paths that in fact provide no solution. Thirdly, as soon as a peak or local minimum is detected, no further improvement is possible on the basis of local perturbations inherent to \( \phi \) in hill climbing and the process must be re-initiated.

A further problem with hill climbing involves following ridges. As indicated in the Figure 31a, there is a definitive direction which we would like an algorithm to choose, but the graph provide no such single, legal move. The possible solution, as indicated in Figure 31b, implies

---

31 Strictly speaking, there is a subtle difference between hill-climbing and a best-first search technique. This is more a question of definition than anything else. However, despite the differences, we shall not describe best-first separately here, but suppose that it can be subsumed under hill-climbing.
a combination of several steps. These sequences of moves are referred to as macro operators (Korf, 1985). However, the discovery of these macro operators is a search problem in its own right; the space of the macro operators grows exponentially with the difficulty of the problem and it is not obvious which criteria should be used to create these macro operators.

Hill-climbing is an example of an irrevocable search strategy, since it possesses no means to re-track previously made decisions, even though they may have offered more promising alternatives. This is due to the lack of memory in such a narrowly defined hill-climbing procedure. Obviously, an algorithm constructed purely on this principle does not possess a facility to store information about previously visited vertices.

```
procedure Hill_Climbing;
begin;
  best := NIL;
  while (not (Test(Object)) and not(best = Object)) do
    begin
      best := Object;
      for all children Objects of best
        if (Evaluate(Object)) >= Evaluate(best)) then
          best := Object;
    end;
end.
```

Figure 32 Pseudo code for hill-climbing

Hill-climbing is very useful in the presence of a highly informative fitness landscape which repulses the search process from local extrema and quickly progresses towards a global optimum. However, in most realistic search problems, which are commonly noisy and sometimes even discontinuous, its performance is rather poor.

As has been highlighted earlier, the main objective in hill-climbing and best-first search-flavoured techniques in general, is to find a goal as quickly as possible. If, however, the objective is altered so as to find the shallowest goal, for example, then a different approach is viable. In such a cases, a technique simply called A* (Hart, et al., 1968) and an iterative deepening version of A*, IDA* for short (Korf, 1985a) are the most appropriate ones. Since these techniques involve the use of so-called admissible heuristic functions, a reference to admissible heuristics may be made, to Gaschnig (1979). Although very useful algorithms in many circumstances, these techniques are not covered here, if only because our objective here is simply to present an outline of alternatives to the procedures that are our main concern in the present work.

5.4.2. On the Power of Non-determinism

In a deterministic algorithm, each string induces a unique path. The acceptance of a string of length n is tested in $n$ steps. In a non-deterministic algorithm on the other hand, testing depends on luck: an acceptable path may be there, but the same string may lead to dead-
locked or to non-final (sub-optimal) states by other paths. In any case, testing still fails or succeeds in at most \( n \) steps. A deterministic search through a non-deterministic automaton pursues all paths induced by a string. If each state allows 2 non-deterministic choices, testing will take \( 2^n \) steps. Thus, the passage from non-deterministic to deterministic testing involves an exponential jump in complexity.

The upshot of this discussion on the power of non-determinism is that we do not know how to avoid the exponential jump in complexity that occurs along the way to determinism. This statement does not prove that this jump is unavoidable, but only that we do not know any move better than this! Non-determinism is natural for us, but alien to computers. Is it then a genuine power within the context of digital computation? Computability is insensitive to non-determinism. Complexity is probably sensitive!

The thread that runs through this chapter on systematic, blind search is that memory requirements grow exponentially as the search progresses. Moreover, the more recent research achievements of many authors indicate and recommend that, once a portion of the search space has been examined without success, one should jump to a distant portion of the search space that might be more promising. The conclusion is that one should abandon the memory and examine the search space in a certain non-deterministic, non-systematic way. Although the guarantees of non-redundancy, completeness and informedness cannot be fulfilled, a large number of empirical studies confirm the superiority of the non-systematic methodologies over systematic ones for difficult and large search problems. Roughly speaking, these difficult search spaces are so large that it is practically impossible to use a systematic, brute force, approach. Moreover, since the search spaces are so large, the probabilities of repetitiously exploring isolated parts of the search space (that is, of getting stuck in one part of the search space) are generally very low when proceeding non-systematically.

Thus, the emphasis is on finding the right balance between the exploration and the exploitation of the search space. Exploration of the search space would correspond to the investigation of new, unexplored portions of the search space, whereas exploitation would proceed through the utilisation of the knowledge and information already detected in the search space to help search for better configurations. These two techniques are counter-productive, the one to the other, and a good search algorithm must find a reasonable trade-off between the two. Quite obviously, random (non-systematic, non-informed, non-deterministic) search is an excellent approach for exploration. In the other vein, hill-climbing (deterministic, informed) search is exploiting the search space's feedback successfully, while performing very little exploration.

In the continuation, some techniques that utilise the power of non-determinism in combination with successful exploitation mechanisms are presented.

### 5.4.2.1. Scatter Search and Scatter-Taboo Search

Scatter search (Glover, 1977) is a technique which maintains a population of potential solutions. This search technique uses a preferred subset of reference points to generate new trial points (offspring). The offspring are created by constructing a so-called weighted linear combination, which can be understood as a version of multi-crossover in which more parents
(usually more than two) contribute to the generation of successors. Memory requirements in the scatter search are constant over the algorithm's execution, and are equal to the number of the generators that are retained as 'parents' for the production of subsequent generations of solution points.

Later work (Glover, 1989; Glover, 1990), was directed towards blending the scatter search and taboo search procedures. Taboo search is a technique which restricts the selection of a new offspring through the introduction of an additional memory storage in which certain, prohibitive (taboo) moves are accumulated.

After the initialisation and evaluation, the scatter-taboo search algorithm classifies potential solutions into several sets: (1) the set of elite historical generators $\mathcal{H}$ consisting of some fixed number of best solutions, as obtained through the search process; (2) a set of taboo generators $\mathcal{H} \subseteq \mathcal{T}$, consisting of solutions currently excluded from consideration; (3) a set of selected current generators $\mathcal{S}$ consisting of the best elements of $\mathcal{H} - \mathcal{T}$, and (4) a set of selected current generators $\mathcal{S}$ consisting of the best elements of all possible moves $\mathcal{S}$. The $\mathcal{S}$ is created as collection of all points generated from $\mathcal{H}$.

```
procedure scatter-tabu
begin;
    t=0;
    initialise P(t);
    evaluate P(t);
    classify P(t);
    while termination condition not satisfied
    begin
        t=t+1;
        create R(t);
        evaluate R(T);
        select P(t) from P(t-1) and R(t);
        classify P(t);
    end;
end.
```

Figure 33 Pseudo-code for scatter-tabu search

Memory in this technique increases quasi-linearly throughout the search process and is proportional to the (constant) number of generators that are used to create offspring, as augmented by historical generators after every generation. The cost of the technique is low and involves the generation of the offspring through multi-crossover.

It is interesting to regard scatter search and scatter-taboo as versions of randomised beam search. Similarly to beam search, scatter search maintains a collection of points that are used

---

32 In the original texts Glover refers to the technique as tabu search. In this thesis, however, the British English spelling is adopted; thus taboo search.
as 'seeds' for offspring. One of the major differences between the techniques is that scatter search supports a population of points, whereas in beam search only one configuration is an active point. Otherwise, both techniques use this point or these points to explore only \( c \leq R(v) \) offspring from the configuration point \( v \).

The other major difference between the techniques is the method used for the generation of the offspring points. While generating offspring, scatter search does not utilise information from the problem domain in order to infer which one to visit next. These points are generated randomly. Beam search utilises the feedback from the problem domain and expands only the \( c \) most promising nodes. Thus, both techniques sacrifice completeness in favour of pruning the search space.

5.4.2.2. Simulated Annealing (SA)

Simulated annealing\(^{33}\) (Kirkpatrick et al., 1983) was derived from statistical mechanics for finding the near globally-minimum cost in large optimisation problems. In the continuation we shall briefly describe the underlying processes; however, for more complete information, a reference may be made to, among others, (Davis, 1987; Davis and Steenstrup, 1987 and Ackley 1987). Since the inspiration for the technique arises from statistical mechanics and the analogy is drawn on the basis of the behaviour of a system of particles in thermal equilibrium at temperature \( T \), the behaviour of a system of particles can be modelled using such statistical thermodynamic techniques as those developed by Metropolis et al. (1953). Using these techniques, Geman and Geman (1984) have shown that, as cooling time approaches infinity, the distribution of stable configurations converges to a Boltzmann distribution. Within this set of stable configurations, the greatest interest accrues to low energy configurations. In physical systems, this is due to their favourable physical properties, but, due to the nature of the Boltzmann distribution, the number of such low-energy, stable configurations is very low as compared to the total number of stable configurations. In the case of metal casting, it was discovered heuristically that, when the temperature of the system was gradually lowered, step by step, while spending 'enough' time at each temperature, and especially near the 'coolest' lowest energy configuration, the probability of attaining this 'best' configuration approached unity.

In a similar way to metal casting processes, simulated annealing searches for the configuration that possesses certain minimal properties. The analog to the best energy in the physical system is the objective function, the temperature becomes an exogenous algorithm control parameter, the particle configuration corresponds to the variable configuration influencing the objective function, and the cooling process becomes analogous to the process of searching for the near-optimum configuration.

\(^{33}\) The name of the procedure is the result of an analogy with metal-casting techniques. When molten metal is poured into a mould, it gradually cools into a solid that will retain its overall shape. In order to make the final solid as non-brittle as possible, the temperature should be reduced gradually; when this is done and the metal is solidified, it is said to have annealed. In simulated annealing, when the problem-solver's attention cannot be attracted away from of the (local) maximum with which it is currently dealing, the algorithm's result can be said to have annealed in a similar way.
We observe at this point again how in this descriptive representation of our 'artificial' or 'virtual' world we are constructing a *rhetoric* by analogy, and with this and inseparably from it we are taking our *grammar* from one field of human endeavour to another. As we continue doing this, however, we induce a *generalised grammar*, applying over several and often disparate domains of discourse. And then, to the extent that we introduce further a *logic* into this, so we construct a purely logical generalised grammar of the kind introduced by Husserl and investigated further again by Derrida (see Abbott, 1992, and Abbott and Minns 1996).

The pseudo-code in Figure 34 is composed of three parts. The first part is concerned with the generation of a new point. Thus, in a $D$-dimensional parameter space with parameters $a_i$ on ranges $a_i \in [u^p, v_i]$, the algorithm utilises the last saved point (the memory of simulated annealing) $\omega_k$ to generate new points. This new point can be generated in many different ways. Ackley (1987) anneals binary strings and uses 'bit-flipping'. Ingberg (1993) uses a distribution defined by the product of distributions for each parameter $g_i(\omega_k, T_i)$ in terms of random variables $\chi_i \in [0, 1]$, where:

$$\omega_{k+1} = \omega_k + \chi_i(v_i - u_i)$$

and temperatures $T_i$ (and again by an obvious analogy to statistical mechanics):

$$g_i(y_i, T_i) = \frac{1}{2(|y_i| + T_i) \ln(1 + \frac{1}{T_i})}$$

The second part of the algorithm encodes the so-called *acceptance probability density function*. This is typically, and as stated in the pseudo-code as well, a *Boltzmann test*:

$$IF \exp\left[\frac{F(\omega_{k+1}) - F(\omega_k)}{T}ight] > random[0, 1]$$

Finally, the third part of the algorithm encodes the *temperature reduction schedule*. In principle, the temperature $T$ is lowered at every step. In his very fast simulated reannealing (VFSR) model, Ingberg (1993) proposes:

$$T(i) = T_0 \exp\left(-c_k^{1/D}\right)$$

where $T_0$ represents some initial temperature and the subscript $i$ denotes the total number of generated points in standard SA, and the total number of accepted points in the VFSR variant of SA.
procedure Simulated_Annealing
begin
  t=0;
  initialise temperature $T$;
  select a current configuration $\omega_1$ at random;
  evaluate $\omega_1$;
  repeat
    repeat
      select a new configuration $\omega_2$ in the
      neighbourhood of $\omega_1$ by altering $\omega_2$;
      if $(F(\omega_1) < F(\omega_2))$
        then $F(\omega_1) \leftarrow F(\omega_2)$
      else if random $[0,1) < \exp((F(\omega_1)-F(\omega_2))/T)$
        then $F(\omega_1) \leftarrow F(\omega_2)$
    until(termination-condition)
  T=g(T,t)
  t=t+1
until(stop-condition)
end.

Figure 34 Pseudo-code for simulated annealing

The SA algorithm terminates at some small value of $T$: this stop-criterion examines whether the system is 'frozen', i.e. whether hardly no further changes are to be accepted.

Bearing in mind the requirement for slow cooling in the process of metal casting and given our faith in the logic of our generalised grammar, it becomes obvious that the determination of the cooling scheduling should be the key to the successful simulated annealing algorithm. On the one side, a rapid temperature decrease lends itself to a fast algorithm, without however guaranteeing a near-optimal solution. A very slow cooling results in a slow algorithm. The other critical component for efficient simulated annealing is a competent methodology for the generation of new configurations.

Thus, simulated annealing generalises hill-climbing and eliminates its main disadvantage: the dependence of a solution on a starting point. The purpose of simulated annealing is to avoid the problem of local extrema by taking an occasional, undirected step (modelled through the Boltzmann acceptance test) in a direction other than the most promising. Thus, SA avoids the irrevocable and 'greedy' character of hill climbing.

To conclude this short discussion on simulated annealing and to make it comparable with other search methodologies, the memory and cost requirements of SA should be estimated. The memory requirement can be made equal to the number of configurations that are retained for the generation of offspring.
6. Evolving Solutions

This chapter is in praise of the solution of a mystery and this is the mystery of complex design. Darwinism today is a widely accepted doctrine that explains the process of complex design as a process of adaptation and alteration of lineages of information within their environment. The beautiful, and even elegant solution presented originally in 'The Origin of Species by Means of Natural Selection' (1859) provides an explanation of the origin of entities as complex as Amoeba, Cichasoma maculicaundae (which happens to be a fish) or Homo sapiens. Moreover, the solution provides an explanation of the origin of complex structures generally (even if these are organisms in this particular case) 'from first principles' and thus exemplifies the most extreme form of reductionism. In Darwin's own words (1859):

"I believe that animals have descended from at most only four or five progenitors and plants from an equal or lesser number.

Analogy would lead me even one step further, namely to the belief that all animals and plants have descended from the same one prototype... Therefore I should infer from analogy that probably all organic beings which have ever lived on this earth have descended from some one primordial form, into which life was first breathed."

(emphasis added)
This is to say that, ever since some first form of life - some first self-replicating entity capable of (genetically) transferring information - was created, some three billion years ago, it is the same ‘immortal’ germ-line that has been permanently altered, adapted and ultimately propelled even to the most complex forms of life presently living on this earth.

Let us put to one side, for the moment, the obvious objections to this view that this is only an ideological veneer that has been superimposed upon nature, that our internal time consciousness does not permit any meaning to be given to ‘three billion years’ other than through one or the other analogue device recording chronological time, etc., etc., etc., ... and let us accept this view in all its nineteenth century naïvety.

Then, most obviously, natural evolution has been extremely successful in creating many ‘useful’ things. Technology can accordingly be nothing but jealous about the successes of Natural Evolution. The grace and, for example, minimal energy consumption of a fish swimming in water cannot be matched by any human creation. The rate of energy consumption for a given speed of any modern submarine, let alone any surface vessel, exceeds that of any fish by many, many times.

Our present idea is merely that of plagiarising ideas that arise in the descriptive apparatus of Darwinian evolution. Our objective is to use the creative power of this apparatus to construct entities that are useful in our water management and engineering applications. We wish also to explore the possibility of initialising the evolution process with only small a number of primitive proto-organisms. These organisms will reproduce, sexually and asexually, battle for resources and ultimately arrive at capacities where they will be well adapted to their environment, just as fish are well adapted to water.

Natural evolution can best be understood as a process of adaptation. At the same time, adaptation can easily be associated with a ‘growth of knowledge’. Clearly, the knowledge of an entity adapting to its environment can be supposed to grow constantly.34 In its essence, Darwinism is a theory of cumulative knowledge processing - and in the present context it can be conceived as a theory of knowledge accumulation.

It took a long time for a theory of evolution to establish itself. Today’s most widely accepted theory, originated by Darwin, appeared some 200 years after Newton’s Principia, for example. One wonders how it was possible that the explanation of such a central question escaped attention during such a long span of human history. How could it happen that Archimedes, Galileo or Leibnitz did not find a reply to this question, as one of the most fundamental to our own very existence. The explanation had to wait for Darwin’s immensely powerful yet beautifully simple interpretation of the mechanism of evolutionary change.

The main reason for such a long and difficult maturing of evolutionary thought may perhaps

34 Within this evolutionary context, it is supposed that, through inheritance of genetic material and the accumulation of (genetically expressed) knowledge, the evolving entity learns to maximise its utilities in a given environment. The phenotype is represented only as an embodiment of genetic information that has been accumulated and tested over many generations.
be sought in the nature of the appearances of our world. We are surrounded by artificial, engineered entities. Most of the world around us is by now adapted by engineers and other technologists to suit our purposes better. It is very difficult and intellectually demanding to accept the fact that certain, apparently blind forces of nature perform similar engineering task more successfully than ourselves. It is even more an affront to our arrogance to accept the fact that we are only an intermediate outcome of such an evolutionary process, and that our 'superior' skills are only very limited. Sigmund Freud often remarked that the train of great revolutions in the history of science has but one common, ironic, feature: each such revolution knock human arrogance off another pedestal, further undermining our opinions of our own self-importance. In Freud’s three examples: Copernicus moved our home from the centre to a periphery; Darwin then relegated us to a ‘descent from an animal world’; and, finally (in one of the least modest statements of intellectual history), Freud himself ‘discovered’ the ‘unconscious’ and explored the doctrine of a fully rational mind as a form of myth.

Natural evolution can as well be ‘understood’ as a search process, the search being directed towards a configuration that will maximise the time before extinction, and correspondingly maximise the use of the available, life-supporting resources in an environment in the longest run. Later in this work, some effort and space will be devoted to establishing more formal links between search algorithms, as presented in Chapter 5, and evolutionary algorithms.

In this chapter, however, the successes of such an evolutionary process approach are merely reported. In the continuation, the properties of natural evolution are briefly outlined, several evolutionary algorithm techniques are briefly described and, finally, some results of a few numerical experiments that illustrate the power of simulated evolution are presented.

6.1. Properties of Natural Evolution

Natural evolution as nowadays studied by evolutionary geneticists has become an extremely complex subject. Its characterisation in a few lines is impossible and this is, also, a task for which the author is ill equipped. Here, only the outline of the most fundamental properties is presented, anticipating that these properties will enable a model of evolution to capture the creativity of nature. However, it must always be borne in mind that such a model can never be more that just a crude simplification of the processes that occur in nature.

6.1.1. Explaining the very improbable

But, before everything else, is there any such thing as evolution? How does it happen that any thing evolves? Does it really evolve at all? Could the different species not just as well have been set into the world by a certain creative force and simply ‘parachuted’ into the environment to establish the world as we know it. The existence of an evolutionary process, even as it cannot be proven, cannot be axiomatised so very easily.

The most common approach to evolution is one that denies its existence altogether: we may call this Creationism. God proclaimed ‘Fiat Lux’ and then flooded His new universe with
brightness. According to literal (non-allegorical) interpretation of the Biblical text, all species were created separately, and remain for ever immutable (that is, they cannot give rise to other species).

Creationism has a zero 'corroborability' (sensu Popper, 1959), since there are no non-tautologous consequences of it. That is, if all species arose separately and not by descent, then, in the absence of a theory of a creative mechanism itself, we cannot predict anything about what relationship should exist between the species, present or past, so that creationism simply does not address the problem of relations between species. Creationism, by its very nature, has no interest in these relations, since it posits that they do not exist.

Some readers may have recognise that the title of this paragraph is borrowed from a famous essay by an Oxford biologist, Richard Dawkins, whose remarkable Blind Watchmaker addresses this issue more fully. For the present purposes, we may simply summarise the 'literalist' theological arguments, and in particular those advocated by the eighteen century theologian William Paley. His Natural Theology - or Evidences of the Existence and Attributes of the Deity Collected from the Appearances of Nature, published in 1802, is one of the best-known expositions of the 'Argument from Design', always the most influential of the so-called 'arguments for the existence of the God'. This work, which is indeed a philosophical masterpiece, even if it does not stand up at all under the criticism of the 'post-modern' theologies of such as Bultmann and Barth, begins with a now famous passage:

"In crossing a heath, suppose I pitched my foot against a stone, and were asked how the stone came to be there; I might possibly answer, that, for anything I knew to the contrary, it had lain there for ever: nor would it perhaps be easy to show the absurdity of this answer. But suppose I had found a watch upon the ground, and it should be inquired how the watch happened to be in that place; I should hardly think of the answer which I had before given, that for anything I knew, the watch might have always been there."

Evidently, Paley distinguishes the physical, natural things, like stones, and some other socially-purposive things, like watches, that exist to fulfil certain definitive intentions. Quite logically, this realisation of a certain design undertaken to realise a particular function implies:

"... that the watch must have had a maker: that there must have existed, at some time and at some place or the other, an artificer, or artificers, who formed it for the purpose which we find it actually to answer; who comprehended its construction, and designed its use."

Paley continues, and even depicts the work of nature, for

"... every indication of contrivance, every manifestation of design, which existed in the watch, exists in the works of nature; with the difference, on the side of nature, of

35 There are difficulties with the exact interpretation of the notion of 'the species' in biology.
being greater or more, and that in a degree which exceeds all computation."

*Touché!* Nobody can possibly disagree with such an argument *per se*. And so, as Paley builds this argument, so he anatomises the ‘machinery of life’. The cherished example is the one of the human eye, the same one that will be later much used by Darwin. Just as the telescope was designed with a purpose of assisting vision, the same argument should be used to explain the existence of an eye, which was designed with the purpose of acquiring this vision.

These persuasive arguments can be faulted from one perspective, but this provides such a compellingly critical viewpoint within the framework of modern science that we are forced to reject such a theory. The analogy between the machine (watch, telescope) and living organs (the eye) and organisms is unacceptable in the sense that the *creator* of the watch has a certain clairvoyance, seeing a future purpose of the *creation* in his mind. The forces of natural selection appear in this respect, however, to be *so much* simpler. For, so far as we can see, these forces posses no foresight, and thus effectively, no vision. They only answer to the forces of chemistry and physics that do not plan for a future but are *self-organised* and it is this self-organisation that in a certain special way assures their present existences. The best conclusion of present-day physical and biological science to this counter-argument is that of Dawkins’ (1986) to the effect that:

"If it can be said to play the role of a watchmaker in nature, it is a *blind* watchmaker."

Of course, and again, from the point of view of a ‘post-modern’ theology, the cogency of the argument of both sides is entirely spurious: theology always leaves open the ‘absurd possibility’ (in the words of Kierkegaard) that it is we who are blind, and indeed are necessarily blind, to the workings of the ‘watchmaker’. Bearing this in mind, we shall in the rest of this chapter describe Paley’s argument as literal creationism or *naive creationism*.

### 6.1.2. Phylogenetic tree perspective

In concerning the Origin of Species, it is quite conceivable that a naturalist, reflecting on the mutual affinities of organic beings, on their embryological relations, their geographical distribution, geological succession, and other such facts, might come to the conclusion that each species had not been independently created, but had descended, like varieties, from other species...

Darwin (1859, Introduction)

*A phylogenetic tree perspective* is a particular point of view upon how species arise from other, and apparently ‘different’ species. This view makes some reasonably non-tautological predictions and is therefore corroborable and testable. The basic prediction is that any given unique, phylogenetic tree must be compatible with *all* manifestations of that tree. Within the overall framework of modern science, this perspective has a (much) higher corroborability than does *naive creationism*. 
Gould (1983, Essay 19) argues forcefully for the view that the phylogenic tree perspective should not be called a ‘theory’ at all but rather a ‘fact’ - on the basis that it has been ‘confirmed to such a degree that it would be perverse to withhold provisional assent’. Gould reserves the term ‘theory’ in this context for hypotheses relating to the mechanisms regulating the development of the phylogenetic tree. Again, we close these remarks with Dawkins’ observation:

"If a single, well-verified mammal skull were to turn-up in 500 million-year old rocks, our whole modern theory of evolution would be utterly destroyed."

(Dawkins, 1986)

6.1.2.1. The Classical Darwinian Evolutionary Paradigm

The most widely accepted evolutionary theory remains the Darwinian, which asserts that the development of living forms is fully accounted for by statistical processes operating within populations and species, and that these statistical processes are restricted to those of reproduction, mutation, competition and selection.

The dominant, conservative view in evolutionary thinking over the last few decades has regarded evolution as being driven by a ‘field of forces’. Selective pressures act on the genetic variety of a population, producing changes over time so as to optimise a ‘fitness potential’. The adaptionist or Darwinian attitude adopts this process of natural selection as the main factor in organic evolution. Orthodox evolutionary theory does not deny that there is a number of other factors operating in evolution; it simply plays down the importance of these other extraneous factors, and accounts for the observed phenomena solely on the basis of optimising ‘fitness’. The origin of the living organisms and their diversity is referred exclusively to this external referent which is regarded as a ‘landscape’ over which a code can navigate along trajectories of ever-improving fitness.36

Reproduction is an obvious property of extant species. Most species have an unlimited reproductive potential: in situations in which the population size is not restricted by environment and resources located in the animal’s surroundings, the population size would grow exponentially. Since, however, these resources are of only limited magnitude, they can accommodate only a population of a certain size, so that a competition for resources must arise. Through the resulting competition, ‘weaker’ candidates are eliminated. Some of them die, some of them get injured and some of them simply do not possess that excess of energy that would serve to promote them to dominant positions within their population. The more fit candidates, on average, use their advantageous characteristics to propel their own interests within their environment. There are many distinctive ways by which benefits are obtained: advertisement, aggression, territory protection, etc., most of which provide advantages with respect to reproduction. Thus the more fit individuals typically reproduce more often and

36 We refer here to an evolving entity as a ‘code’. The code, in this sense, stands for the set of instructions, in their most general form, that have to be executed in order to advance over a landscape. Even the body necessary to execute the program is coded for.
more successfully than do the rest of the population, thereby proliferating, and through this process with all its attendant features they attain to an advantageous position. As Darwin expresses this feature:

"This presentation of favourable variations and the rejection of injurious variations, I call Natural Selection. Variations neither useful nor injurious would not be affected by natural selection, and would be left a fluctuating element, as perhaps we see it in the species called polymorphic."

Implicitly, we assume, so to say, that elephant begets elephant, that the offspring is ‘similar’ to its progenitors. The modern theory of heredity was initiated in the mid-nineteen century and much has now changed. But since some of the earlier notions of the theory of heredity remain exceedingly valuable within the context of artificial evolution, these concepts should still be introduced in this place.

The first steps in evolutionary genetics per se were made by August Weismann. A German biologist, who worked in this field at the end of the last century, Weismann recognised two independent processes that emerge from the cell division of a fertilised egg. This led to the development of the dual concepts of independent germ and soma lines. The germ line identifies a sequence of transfers of inheritable material from parents to offsprings which is potentially immortal. The soma line is represented by adult bodies, which are obviously of a limited life-span. The important deduction that follows from such a suggestion is that it is the flow of information that matters, not the flow of energy. Following this line of reasoning, one can easily distinguish an organism’s genotype, which is constituted of genetic material inherited from its parents, and the organism’s phenotype, which is a composition of characteristics of the individual living entity.

Another important observation that arose from Weismann’s findings was that exchange of information occurs at one organisational layer and selection at another. Basically, it is the genotypes that are re-combined and altered in an evolutionary process. However, the principle of survival of the fittest, appears to operate only on the phenotypes. It is the more adapted

---

37 This unfortunate term - survival of the fittest - has often been used to discredit the Darwinian theory of evolution on the basis that the ‘fittest’ are themselves defined, implicitly or explicitly, as those that survive, so that this ‘principle’ is tautological. However, as Dawkins (1982) explains:

"This tautology is not a property of Darwinism itself, but merely of the catch phrase that we sometimes use to describe it."

Moreover, the term ‘survival of the fitness’ is not Darwin’s invention. It is well known that Darwin was strongly influenced in his vocabulary and overall way of thinking by the mores of the industrial society in which he lived, as recorded in the works of such as Comte, Herbert, Spenser, Marx and Wallace, but the following letter, quoted by Dawkins, sheds another light on this:

My dear Darwin,

I have been so repeatedly struck by the utter inability of number of intelligent persons to see clearly,
adult body that is ‘favoured’ by the forces of selection to reproduce more often, and it is the less adapted soma that is removed by not being selected for further reproduction from the population.

Although Weismann’s findings were fundamental for establishing an understanding of evolutionary ideas, modern genetics as a whole can be said to have originated with the rediscovery of Gregor Mendel’s\textsuperscript{38} research. Mendel was interested in discovering the underlying inheritance pathways and factors. Probably Mendel’s most fundamental contribution was

"... not the conclusion he reached, but the way he reached it. The first critical step was to ask the right question. The obvious question to ask would be: why do mice give birth to mice and elephants to elephants? Although obvious, this would not have been a fruitful question to ask, because there was no way in which it could be answered."

(Maynard Smith, 1986)

In 1866, long before the relationship to the biochemical foundations of the theory of heredity was established, Mendel formulated the classical concepts of inheritance on the basis of breeding experiments in which the distribution of different phenotypic forms was analysed in the progeny of known matings. Breeding and progeny testing were (and to a great extent still are) fundamental tools of genetics. As an example, let us perform a similar simple experiment, using an imaginary breeding and progeny investigation.

Let us assume that a certain phenotypic trait of an entity can take two values: Red (denoted as R) and White (W). In order to accommodate the duality, the genotypes are such that both parents can ‘supply’ information to their offspring, but in such a way that one piece of the ‘information’ used for assembling the offspring’s genotype is inherited from one parent and the other piece of ‘information’ from the other parent.

or at all, the self-acting and necessary effects of Natural Selection, that I am led to conclude that the term itself, and your mode of illustrating it, however clear and beautiful to many of us, is yet not the best adapted to impress it on the general naturalistic public... I wish, therefore, to suggest to you the possibility of entirely avoiding this source of misconception in your great work (if not too late), and also in future editions of the ‘Origin’, and I think it may be done without difficulty and very effectually by adopting Spencer’s term (which he generally uses in preference to Natural Selection), viz. ‘Survival of the fittest’.

Wallace (1866)

\textsuperscript{38} Gregor Mendel was a head of a monastery in Moldavia while experimenting on inheritance factors in pea plants. His work was available long before Weismann’s findings but was only rediscovered some thirty years after his death.
<table>
<thead>
<tr>
<th>Parents (Mating Types)</th>
<th>Expected Portion of offspring</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR RR</td>
<td>All RR</td>
</tr>
<tr>
<td>RR WW</td>
<td>All RW</td>
</tr>
<tr>
<td>WW WW</td>
<td>All WW</td>
</tr>
<tr>
<td>RR RW</td>
<td>½ RR : ¼ RW</td>
</tr>
<tr>
<td>WW RW</td>
<td>½ RW : ½ WW</td>
</tr>
<tr>
<td>RW RW</td>
<td>¼ RR : ½ RW : ¼ WW</td>
</tr>
</tbody>
</table>

**Figure 35** This figure summarises the possible types of mating and progeny ratios for the single-gene colour trait. The relative proportions of genotypes and phenotypes represent the probability that an offspring will fit a given classification.

**Figure 36** An illustration of dominant and recessive inheritance. A cross between a true-breeding white-flowered plant and a true-breeding red-flowered plant results in only one white-coloured plant (middle). However, recessive (white) genes continue to express themselves in subsequent generations.

Obviously, our entity's genotype could be composed by two 'letters' defining the colour (RR, or WW, or RW). The offspring's phenotype would clearly be red in RR-cases, and white in WW-cases. The phenotypical consequences of RW or WR are, however, not obviously defined. Pink would correspond to a completely new quality of phenotype which is not explicitly coded in the genotype (P does not exist in our 'genetic' alphabet). However, in nature, certain genes are dominant over the others. This means that RW can, in our simple example, become red and we would say that R 'masks' W, preventing the latter from
expressing itself in the phenotype.\textsuperscript{39}

Mendel's inheritance factors are indeed understood as ancestors of what is today known as genes. Propelled by advances in cell research, and the discovery of chromosomes in particular, Mendel's work was re-discovered and appreciated only after his death. Mendel never 'saw' chromosomes, and probably never imagined that cytologists would soon witness under the microscope the process that leads to the segregation of alleles\textsuperscript{40}.

6.1.2.2. From Darwinism to neo-Darwinism

According to most traditional views, the environment in which organisms evolve and which they come to know and to which they come to adapt is given, fixed and unique. Here again we find the idea that organisms are basically 'parachuted' into a pre-given environment. This view undergoes refinement when we allow for changes in the environment, an allowance that was already empirically familiar to Darwin. Such a shifting environment provides the selective pressures that form the backbone of neo-Darwinian evolutionary theory. In this revised, kinematic view of evolution, however, a further step is introduced: these selective pressures are re-cast so that they now form a collection of constraints to be satisfied. By these means we come more to emphasize that \textit{the very notion of what an environment is cannot be separated from what organisms are and what they do}. This point has been made with particular eloquence by Lewontin (1989):

"... the organism and the environment are not actually separately determined. The environment is not a structure imposed on living beings from the outside but is in fact a creation of those beings. The environment is not an autonomous process but a reflection of the biology of the species. Just as there is no organism without an environment, so there is no environment without an organism."

The key point, then, is that the species is brought forth and itself participate in specifying its own domain of problems to be 'solved' by 'satisfying' these solutions; this domain does not exist 'out there' in an environment that acts as a 'landing pad' for organisms that somehow drop on parachutes into the world. Instead, living beings and their environments stand in relation to each other through \textit{mutual specification} or \textit{co-determination}. Thus what we describe as environmental regularities are not external features that have been internalised, as representationism and adaptationism both assume. Environmental regularities are the result of a conjoint history, a congruence which unfolds from a long history of co-determination. In Lewontin's own words, the organism is both the subject and the object of evolution.

\textsuperscript{39} But, one wonders at this stage, how this is possible. Nature is inhabited by large numbers of entities in which phenotypical traits are mixtures of their parents' properties. Thus 'pink' quite commonly occurs. 'Pink', however, arise only as a mixture of pigment in such a case and not as a consequence of a change in the underlying genetic coding.

\textsuperscript{40} The word 'allele' is a contraction of 'allelomorphs'. Each gene is able to occupy only a particular region of a chromosome, which is then referred to as its locus. At any given locus there may exist alternative forms of the gene in the population. These alternatives are said to be alleles of one another.
This classical Darwinism became a neo-Darwinism during the 1930s as a result of a so-called 'modern synthesis' between Darwinian ideas based on zoology, botany and systematics, on the one hand, and the (then) increasing knowledge about cellular and population genetics, on the other. This synthesis established the more fundamental perspective that modifications occur by small changes in organismic traits specified by heritable units, called genes. The genetic make-up responsible for an ensemble of phenotypes leads to differential reproduction rates, and thence to changes in the genetic make-up of a population over generations. Evolution is simply a consequence of the totality of these genetic changes in interbreeding populations. The pace and tempo of evolution can then be measured by changes in the fitness of genes; thus it becomes possible to give a quantitative basis for the visible adaptation of animals to the environment in which they live.

The main reason for the earlier lack of appreciation of Mendel’s work was an unwillingness of contemporary biologists to accept the 'abstract' nature of his theory. During the thirty years that elapsed between Mendel’s death and the re-discovery of his findings, great progress was made in the understanding of cell anatomy. For example, the existence of a nucleus had been recognised, and this was observed to contain certain threads, called chromosomes. Further, the chromosomes were observed to provide precisely the hereditary properties that Mendel had envisaged, in that each body cell was found to contain two sets of chromosomes (e.g. 23 pairs in human cells, see Figure X.), but the gametes (the egg in female and the sperm in male) received only half of each pair.

The chromosomes themselves could not be equated to inheritance factors, since humans were definitely known to possess more than 23 phenotypical traits! But, since the chromosomes resemble threads, it was reasonable to expect that the chromosomes were carriers of genes, and that these were arranged along the chromosomes, like beads on a string.

This expectation led Thomas Morgan and his research team to experiment with the fruit fly Drosophila melanogaster. They followed a number of phenotypical traits simultaneously, and then specifically the structure and behaviour of chromosomes over successive generations. The departure from 'Mendelian genetics' arose in the cases where traits were genetically ‘linked’ so that the traits did not segregate independently, as Mendel claimed they should. These findings led to the identification of such phenomena as pleiotropism and polygenic inheritance.

Figure 37 A metaphase human chromosome spread and resulting idiogram after chromosome images had been cut out and arranged in order
Pleiotropy is the effect that a single gene may simultaneously affect several phenotypic traits. Polygeny is the effect that a single phenotypic characteristic of an individual may be determined by the simultaneous interaction of many genes. As a result, 'most structures of an organism are meaningless without the rest of the organism' (Mayr 1982, p.54). The (one gene)-(one trait) model of evolutionary genetics was seen to be a gross oversimplification.

The next decisive step was the elucidation of the chemical nature of genes. It was only in 1953 that two Cambridge scientists, Watson and Crick, discovered the now famous double helix and described the DNA-replication mechanism. DNA, or desoxyribonucleic acid, is one particular type of macromolecule present in cells. Nucleic acids are strings of four kinds of nucleotide bases: adenine, thymine, guanine and cytosine. The DNA is known to be present in chromosomes, and hardly anywhere else in the cell. It was therefore rather natural to suppose that genes were made of DNA. The other type of macromolecules present in each cell are proteins. They are strings composed of 20 kinds of amino acids. Proteins usually fold to form large globular molecules and they form most of the components of the biological body (muscles, tendons, etc.).

The proteins, composed of 20 different kinds of amino acids, and the DNA, composed of four different kinds of nucleotide bases, are related by a 'genetic code' which transcribes the information contained in the DNA cryptogram into the specification of the amino acid sequence that builds up the proteins, i.e. the biological body. The transcription code itself is relatively simple. Groups of three nucleotide bases 'stand for' one amino acid. This basically means that DNA is capable of coding for $4^3=64$ ordered triplets, which provides a redundancy in the specification for the 20 amino acids. However, not all of the 44 triplets are redundant; some of them are interpreted as instructions, like 'end of protein'.

The revolutionary contribution of the Watson-Crick structure (Figure X.) was to explain how DNA can carry information and how it can replicate. For the purpose of this technical work it is not necessary to describe the chemistry of the various processes of cell division and the 'division of labour' that occurs between metabolites and macromolecules, proteins and nucleic acids and other such entities. It is enough to emphasise that during the replication process, yet other kinds of nucleic acid (messenger and transfer ribonucleic acids, mRNA and tRNA...
for short) utilise the information encoded in the original DNA in order to realise the transcription and generate its copy. This process is rather complex, and one of the major features of evolution is the evolution of such robust processes that work *nearly all* of the time. Yes, most of the time, a very few 'errors' (perhaps rather called 'deviations' or even 'explorations') do occur: it is these that are referred to as *mutations*. In the new cell, this DNA code is to used to create new-old proteins.

![Figure 6. Evolution in sexual and asexual populations, after a diagram first used by the American geneticist H. J. Muller in 1932.](image)

A, B, and C are different favourable mutations which, occurring in different individuals, can come together in a single descendant in a sexual population, but not in an asexual one.

*Figure 39 Propagation of information in sexual and asexual populations (adapted from Maynard-Smith, 1975)*

The term sex derives from the Latin *secare*, meaning to cut or divide something that was originally whole. The exchange of genetic information is fundamental to the cutting, dividing and recombination of DNA that takes place during sexual reproduction. The exchange takes many forms: bacterial conjugation, transduction via viral transport and the more familiar recombination that occurs in plants and animals. But sex requires finding a mate. The
behaviours that are associated with attracting mates often make individuals more vulnerable to predators. Yet, the benefits of sexuality must out-weight these costs. The great majority of existing kinds of species are fully sexual.

Diploid organisms have two copies of each chromosome in every cell other than in the haploid sex cells, with one copy being contributed from each parent. The total number of chromosomes in the cells of most eucaryotes varies between 10 and 50, but can exceed 1000 in some plants. During recombination, haploid gametes are fused to form new diploid cells. For infusion to generate a stable number of chromosomes in the sexually-reproducing lineage, a reduction division must take place.

This division is termed *meiosis*. In this process, each chromosome from both parents is duplicated. Each homologous pair lines up in parallel along the equator (metaphase plate) of the cell. A crossing over between homologous chromosomes may occur in which segments of the chromosomes are exchanged. The rate of crossing-over varies widely from species to species and between the sexes. Under meiosis, a reduction division takes place in which one
of the two homologous chromosomes is sorted into each cell. If crossing-over occurs, it appears to occur only in this first meiotic division. These chromatids undergo a second division such that each gamete receives a potentially hybrid combination of each pair of homologous chromosomes from each parent.

Sexual reproduction offers a significant ability for a species to generate genetic diversity and, as a consequence, phenotypic diversity. The overt advantage of sexual recombination to an evolving species is the significantly increased rate of exploration of the genotypic/phenotypic state space, as compared to clonal parthenogenesis, and this advantage becomes especially pronounced in changing environments. Recombination will ultimately tend to expose a wide variety of individual genotypes to the various phenotypic settings that the environment provides. If it is presumed that each offspring receives each chromosome from each parent in a random manner, the number of such combinations becomes virtually unlimited. In humans, \(2^{21} = 8,388,608\) distinct gametes can be produced simply by considering the set of possible combinations.

Sexual reproduction is clearly a complex process, involving more than simply the crossing-over of homologous chromosomes or the interchange of two strands of DNA. Genetic code also suffers from entropic decay and mechanisms are required to maintain its functionality. This randomisation of genetic information will quickly fail to produce competitive phenotypes. A lack of competitive phenotypes implies the imposition of severe costs to the species unless information maintenance mechanisms intervene (Atmar 1991).

For a well-optimised population residing in a stable environment, sexual recombination is expected to produce well-formed individuals, differing very little from any other individuals of the species. Variations should be purged (or at least allowed to fall below the selection pressure) from the phenotype as more accurate predictors of the environment are evolved. Any mechanism that aids in detecting and correcting errors in the underlying genetic code should provide an advantage to the evolving species. It appears likely that the observed complexity of evolutionary behaviour cannot occur without mechanisms enhancing the likelihood of the proper embryogenesis and maturation of individuals of the species. Sexual reproduction provides yet another form of error expurgation from the evolving phyletic line.

Errors are inevitable during this reshuffling of information. Maintaining the fidelity of the genetic code is critical. The enzyme complex that transcribes the chromosome may also check the nucleotide chain being replicated, adding the appropriate base to the complimentary strand as replication proceeds. Further error-checking (a kind of 'proofreading') is accomplished by checking for mismatches. But replication is not the only source of error. There are many environmental sources of mutational variation. These may result in deletions, insertions and inversions of sections of DNA. Mutations that affect both strands of DNA are difficult to repair because they are difficult to detect through error-checking.

In order to make this notion of adaptation measurable, a certain metric for expressing a degree of adaptedness of organisms needs to be introduced. This is where the notion of fitness comes into biology. This idea of fitness is often formulated as a measure of abundance. It is usually taken as a measure of individual abundance, i.e. as a measure of the surplus offspring
achieved, but it can also be constructed as a measure of population abundance\(^4\), i.e. as the effect of genes on the rate of growth in population. Dawkins (1982) discusses the notion of fitness at quite some length, and even opposes the use of the word, being more in favour of some other, more definitive and precise expressions (like selective value, selective potential or survivability), and offers five possible interpretations of the term fitness. Following tradition, the term *fitness* is accepted in this thesis. From the vantage point of adaptedness, the task of evolution consists of finding heritable strategies, i.e. *sets of interrelated genes that will be more or less capable of contributing to differential reproduction*. When a gene changes so as to improve its performance, it improves its fitness.

Fitness can also be taken as a measure of *persistence*. Here fitness measures the probability of a reproductive permanence over time. What is optimised is not the quantity of offspring but the probability of survival of the line. Clearly, this approach is more sensitive to long-term effects, and so it is an improvement over the more narrow view of fitness as abundance. By the same token, however, it poses formidable problems at the level of measurement.

It has been asserted that although fitness cannot be directly measured, *'its distribution in a population can be roughly estimated in any given environmental context on the basis of the ecology and the functional morphology of the organisms. Hence, it is an empirically testable proposition...'* (Hoffman 1989, p.29). Probably, the only singular measure of evolutionary fitness is the appropriateness of the organism's phenotypic behavioural prediction. Evolving phylogenies tend to increase their ability to predict the regularities of their environment. The quantitative ability to perform suitable predicting and elicit appropriate responses yields a measure of fitness. Thus in the words of Popper (1961, pp. 271-272):

"... survival, or success in the sense of an increase in numbers, may be due to either of two distinguishable circumstances. A species may succeed in prospering because it has managed, say, to improve its speed, or its teeth, or its skill, or its intelligence; or it may succeed or prosper merely because it has managed to increase its fecundity, depending fundamentally on genetical factors, or a shortening of the period of immaturity may have the same survival value as, or even a greater survival value than, say, an increase in skill or intelligence.

... But be this as it may, it should be possible, I think, to [measure] the *success in the adaptation of the individual organisms* of a species ...

Without some distinction such as this ... we are liable to lose sight of the original problems of Lamarck or Darwin, and especially of the explanatory power of Darwin's theory..."

\(^4\) The way of measuring fitness as abundance has a number of conceptual and empirical difficulties (and this is despite of Rubens, Jordaens and the other great depictors of abundance!). First of all, in most animal groups, reproductive success depends on sexual encounters with other individuals. Secondly, since the effects of any given gene are always intertwined with a multitude of other genes, it is not always possible to determine the effects of individual genes. Thirdly, the milieu in which the genes are supposed to express themselves is enormously varied and time-dependent. Finally, this milieu must be viewed in the context of the entire life cycle and ecology of an organism.
It is unreasonable to view and speak of the fitness of any particular gene in isolation from the entire cohesive genetic structure and its corresponding phenotype. While such a simplification is mathematically tractable and has been extensively analysed (e.g. Fisher 1930) it is not broadly useful in predicting or assessing the adaptation of behaviours. Such efforts have been labelled 'bean bag genetics' (Mayr, 1959, 1963, 1988).

Functional behaviour is the sole quality that is ‘optimised’ through selection. Genotypic and environmental variation result in phenotypic variation. Sub-optimal phenotypes are selected against (such as by culling). The result of a more-or-less continuous selection is the rather constant optimisation of the phenotype by means of a re-organising of its behaviour within the context of current environmental demands.

The most intuitive sense of adaptation is that it is some form of design or construction that matches optimally (or at least very well) some physical situation. For example, fishes are well suited for an aquatic environment, whereas zebras are well suited for running in savannah areas. Adaptation refers specifically to the process that is linked to reproduction and survival, i.e. to adapting. This process is - or so one supposes - what accounts for the apparent degree of adaptive design observed in nature. Thus, (Popper, 1961, pp.258-259):

"I assert that every animal is born with expectations or anticipations, which could be framed as hypotheses; a kind of hypothetical knowledge. And I assert that we have, in this sense, some degree of inborn knowledge from which we may begin, even though it might be quite unreliable. This inborn knowledge, these inborn expectations,
will, if disappointed, create our first problems; and the ensuing (somatic time) growth of our knowledge may therefore be described as consisting throughout of corrections and modifications of previous knowledge."

6.1.2.3. The Central Dogma

Consider the generative process as schematised in Figure 42.

![Diagram of DNA, RNA, and Protein]

*Figure 42 Division of labour between DNA and proteins (adapted from Maynard Smith, 1975)*

It appears that the flow of information in this replication chain can only take one direction. It seems that DNA can only provide a code for generating proteins, and that it is impossible
Emergence, Evolution, Intelligence; Hydroinformatics

to use proteins to synthesise the DNA. This is what Crick called the central dogma of molecular biology. The central dogma corresponds closely to Weismann’s claim. In Maynard Smith’s words, the importance of the central dogma is that it provides a molecular explanation for the theory of the independence of germ line and soma line. The central dogma is a theory, but there is very little evidence to suggest that it might be wrong. At the more biochemical level, the distinction between genotype and phenotype may be said to reflect a ‘division of labour’ between nucleic acids and proteins (Maynard Smith, 1975; see Figure 42).

6.1.2.4. Ying and Yang

We come to a kind of ‘chicken and egg’ problem that is a very special kind of recursive problem. The biological body, build from proteins, is needed to execute the DNA-coded information, and this body cannot be created without the DNA. This code needs the results of its execution in order itself to be executed!

Figure 43 Chicken or Egg?

The intrinsic characteristics of life itself are major, if not central, driving forces in evolution, defining what is to count as fit, or otherwise. Thus, what in the orthodoxy of genetics appears as the Other is seen in this light not even as the Origin, but as the Result. We meet here the Derridian logic of différance in its full stature. For this re-reading of evolution, modern biologists bring into the foreground a

42 "So much has now been written about Derrida’s neologistic innovation of différance that we scarcely know where to begin. We can perhaps best introduce it here following Allison, who, in his translation of Derrida (1973, p.82) explains this most briefly as follows:

“From the French différencé, [Derrida] derives the term différance. As in Latin differre, the French différencé bears two quite distinct significations. One has a reference to spatiality, as the English “to differ” - to be at variance, to be unlike, apart, dissimilar, distinct in nature or quality from something. This is even more apparent in its cognate form, “to differentiate”. The other signification has a reference to temporality, as in English to “defer” - to put off action to a future time, to delay, or postpone.”

The word différencé, which might as well be rendered into English as “differance”, brings these two references together, or rather it is that which brings them together. Thus, again from Allison, "différance [is] what makes it possible for sounds to be intelligible as voix ["speech"] and for marks to be intelligible as écriture [or "writing"]).” Thus in the ‘revised’ view of the genetic processes, the différencé is what makes it possible for the germ to be ‘realised’ as the soma, there being both a difference between these ‘forms’ and a deferring of the intervention of the one relative to the other in the world.

The real point from our present perspective is that we are not really dealing very much at all, even in this place, with biological processes as such, but, much more essentially we are dealing with the use and actions of signs, so that what we are really dealing with here is semiotics. For Derrida himself, (1973, p.138), ‘Signs represent the present in its absence; they take the place of the present ... The
number of factors, all relating to the richness of organisms as networks (of traits, of genes, of behaviours). A partial list of these factors is gene interdependence (pleiotropy), developmental constraints, genetic drift, evolutionary stasis, and the various levels for units of selection. As Richard Lewontin said in the recent critique of the orthodox-evolutionary position (1989): ‘It is not that these phenomena [i.e., developmental constraints, pleiotropy, etc.] are not mentioned, but they are clearly diversions from the big event, the ascent of Mount Fitness by Sir Donald Fisher and his faithful Sherpas.’ Increasingly, evolutionary biologists have become engaged in a movement away from Mount Fitness towards a larger and as yet incompletely formulated new theory. Alfred Russel Wallace was fond of saying that ‘Nothing in Nature is not useful.’ (Clemens, 1983). Our culture insists on, and many scientist use, without even so much as a second thought, arguments that appeal to the parsimony of nature and its optimal designs. The list of tangled issues that we have just covered, however, indicates that nature itself must be supposed to behave otherwise.

To conclude this section then, let us reiterate: the crux of the matter is that to explain an observed biological regularity as an optimal fit or optimal correspondence with pre-given dimensions of the environment appears less and less tenable even on its own logical and empirical grounds. Part of the difficulty in moving beyond the neo-Darwinian framework is to determine what to do after we abandon the idea of natural selection as the principal means

sign [is] a deferred present.’ Thus, again (1973, p.146):

“The use of language, or the employment of any code which implies a play of forms - with no determined or invariable substratum - also presupposes a retention and pretention of differences, a spacing and temporalising, a play of traces. This play must be a sort of inscription prior to writing, a proto-writing without present origin, without an arché.”

But now, for the purposes of the present work, this concept of différence of Derrida, this difference between differences and deferring of differences that we can only express by signs, itself undergoes a metamorphosis. For increasingly and soon overwhelmingly, signs and with these their differences and deferings, are no longer mediated exclusively by the human mind, but ‘by themselves’, so to say, in our computers and other equipment.” (M.B. Abbott, unpublished note)
of explanation, when we arrive in a situation in which not every structure, mechanism, trait or disposition can be accounted for by its contribution to some survival value. Are then the things without reason at all? The effect of post-Darwinian evolutionary biology has been to change the debate by studying tangled, circular relationships of congruence among the items that need to be explained. This however, lands us squarely back with the same kind of logic as we evoked before (Maynard Smith 1986, p.40):

"... I do not think that Darwinian natural selection is the only thing we need to understand to understand evolution. However, I do think that Darwin's theory is correct, and that it is the only adequate explanation for me of the most characteristic feature of living organisms. This feature is the way in which their structure and behaviour adapts them to survive and reproduce in a specific environment."

6.1.2.5. The Lamarckian Paradigm

The Lamarckian theory seems to have a great emotional appeal for a certain type of intellectual as well as for laymen. I was once approached by a colleague, a celebrated Marxist historian and most cultivated and well-read man. He understood, he said, that the facts all seemed to be against the Lamarckian theory, but was there really no hope that it might be true? I told him that in my opinion there was none, and he accepted this with certain sincere regret, saying that for ideological reasons he had wanted Lamarckism to be true. It seemed to offer such positive hopes for the betterment of humanity.

Dawkins (1986, p.291)

The recent resurrection and appreciation of Lamarck's thought does not need much additional documentation. Lamarck's *Philosophie Zoologique* (1809), after being cleansed of various misunderstandings and misinterpretations, is now revealing its elegant beauty of truthfulness. Lamarck's intuitive respect for the environment is now being vindicated.

Lamarck recognised that in nature there are no such things as classes or orders of families, but only individuals. This is in fact a profound observation, as outlined elsewhere in this text (Section 6.1), and as emphasised also by Foucault (1973), and this should become very clear later in this work, where this feature comes to take a central position.

Lamarck also rebelled against the then-supposed immutability of species. His explanation is a beautiful expression of system-environment structural coupling and of the idea of structural

---

43 According to Wills (1991, pp.65-66), the title Chevalier de Lamarck was a somewhat dubious affectation (Lamarck's 'real' name being Jean-Baptiste Antoin de Monet). However, convention now decrees that he is known simply as Lamarck.
adaptations as compensations for environmental perturbations.\footnote{It now appears, in support of Lamarck’s position, that certain species not only lack detectable sex chromosomes (amphibians), but in many ‘cold-blooded’ species (alligators, turtles and lizards) environmental temperature is a determinant of sex differentiation. (See Bull and Vogt 1979) Constant incubation temperatures of 31 °C and above produces females, cooler temperatures, of 24-27 °C, produce males, and, in Chelydra, even cooler temperatures (20 °C) again produce females. No peculiar assortment of genes at conception or even during cleavage can account for these sex ratio biases. Chromosomes seem to be capable of degrading and rebuilding their own components in response to environmental perturbations.}

"... when an individual of a given species changes its locality, it is subjected to a number of influences which little by little alter, not only the consistency and proportions of its parts, but also its form, its faculty, and even its organisation; so that in time every part will participate in the mutations which it has undergone."

Such slogans as ‘function creates the organ’ or ‘acquired characteristics are inheritable’ have little to do with Lamarck’s theory. He summarises his doctrine in six points (Leduc 1911; see also Zeleny, 1992):

- All organised bodies of our globe are veritable productions of nature, which she has successively formed during the lapse of ages
- Nature began, and still recommences day-by-day, with the production of the simplest organic forms. These so-called spontaneous generations are her direct work, the first sketches of organisation
- The first signs of an animal or a plant growth being begun under favourable conditions, the faculties of commencing life and all organic movements thus established have gradually developed little by little the various parts and organs, which in process of time have become diversified
- The faculty of growth is inherent in every part of an organised body; it is a primary effect of life. This faculty of growth has given rise to the various modes of multiplication and regeneration of the individual, and by its means any progress that may have been acquired in the composition and forms of the organism has been preserved
- All living things that exist at the present day have been successfully formed by this means, aided by a long lapse of time, by favourable conditions, and by changes on the surface of the globe - in a word, by the power that new situations and new habits have of modifying the organs of a body endowed with life
- Since all living things have undergone more or less change in their organisation, the species that have been thus insensibly and successfully produced can have a relative constancy, and can be of no very great antiquity
There is indeed a *grandeur* in this view of life:

"... the Lamarckian theory can explain adaptive improvements in evolution by, as it were, riding on the back of the Darwinian theory. Given that Darwinian selection is there in the background, to ensure that some acquired characteristics are advantageous, and to provide a mechanism for discriminating the advantageous from the disadvantageous, the inheritance of acquired characteristics might, conceivably, lead to some evolutionary improvement. But the *improvement*, such as it is, is all due to the Darwinian underpinning. We are forced back to Darwinism to explain the adaptive aspect of evolution."

Dawkins, 1986, p.300

Even further, following rather closely Gould (1980, pp. 70-71):

"In fact, it can be argued that a form of Lamarckian evolution is not only possible, but well known, and has been crucially significant in the evolution of at least one species. This is so-called *cultural evolution*, particularly in the human species. Humans acquire cultural characteristics ('knowledge') selected by their parents (among others) from their own accumulated wisdom. Humans, in turn, generate new characteristics ('knowledge') not possessed or transmitted from their parents. This is Lamarckian evolution *par excellence*, and is undoubtedly a significant mechanism for the evolution of the unique characteristics of this extraordinary animal.

*Homo sapiens* arose at least 50,000 years ago, and we have not a shred of evidence for any genetic improvement since then. I suspect that the average Cro-Magnon, properly trained, could have handled computers with the best of us (for what it is worth, they had slightly larger brains than we do). And what we have accomplished, for better or for worse, is a result of cultural evolution. And we have done it at rates unmatched by orders of magnitude in all the previous history of life. Geologists cannot measure a few hundred of few thousand years in the context of our planet's history. Yet, in this millisecond, we have transformed the surface of our planet through the influence of one unaltered biological invention - self-consciousness. From perhaps one hundred thousand people with axes to more than four billion with bombs, rockets, ships, cities, television, and computers - and all without substantial genetic change.

*Cultural evolution* has progressed at rates that Darwinian processes cannot begin to approach. Darwinian evolution continues in *Homo sapiens*, but at rates so slow that it no longer has much impact on our history. *This crux in the earth's history has been reached because Lamarckian processes have finally been unleashed upon it. Human cultural evolution*, in strong opposition to our biological history, is Lamarckian in character. What we learn in one generation, we transmit directly by teaching and writing. Acquired characteristics are inherited in technology and culture. *Lamarckian evolution is rapid and accumulative. It explains* the cardinal difference between our past, purely biological mode of change, and our current, maddening acceleration toward something new and liberating - or toward the abyss."

(emphasis added)
6.1.3. Main Evolutionary Principles

After such a quick tour d'horizon of hereditary processes as perceived through a quasi-biological vocabulary, we may try to generalise the mechanisms and to define the basic evolutionary process that we wish to emulate. In a principle, there are three main criteria for an evolutionary process to occur (Maynard-Smith, 1975):

- **Criterion of Heredity**: Offspring are similar to their parents: the copying process maintains a high fidelity;
- **Criterion of Variability**: Offspring are not exactly the same as their parents: the copying process is not perfect;
- **Criterion of Fecundity**: Variants leave a different number of offspring; specific variations have an effect on behaviour and behaviour has an effect on reproductive success.

These criteria provide the necessary conditions that a realistic evolutionary model must fulfil. First, a way of transferring the information from parents to offspring so that the offspring are alike to their ancestors must be defined. Second, errors must be introduced into this copying process, so that some variability in material that is being inherited must be instituted. In nature, these errors are referred to as mutations. And, third, there must be a mechanism to explain how these modifications occur. This mechanism operates by selecting the designs (phenotypes) which cope with the current environment more efficiently. The better adapted entities will have better chances of reproducing and will consequently produce more offspring, as corresponds to the actions of natural selection. It is essential to appreciate these requirements, otherwise our search for the best adapted germ lines and phenotypes would be random, without any accumulation of knowledge occurring through the process. Margalef (1963) eloquently depicts this universe of matter and information:

"Only a universe of energy could have no past. If there is matter, structures grow and differentiate and a past can be recognised and partially reconstructed. It is the problem of durationless non-matter versus enduring matter (Fokker, 1966). At one end of the spectrum is biblical chaos, a past without a past, because no matter exists to convey information. At the other end there is only information and no decisions - static information forever. We and the entire universe are caught in between, with the wonderful opportunity to enjoy creation. Some sophisticated people would rather speak of existential anxiety, but biologists in general and field naturalists in particular are really childlike and enjoy nature like a child playing in a mud puddle after the rain has given way to sunny skies."
6.1.4. Formalisation of natural processes

\emph{Natura non facit saltum}

Linnaeus

Living organisms can be viewed as the duality of their genotypes and phenotypes, with the \emph{différence} of their duality, providing the third, relating element. As stated previously, the genotype corresponds to an underlying genetic coding, and the phenotype to the manner of response and the morphology of organism corresponding. Lewontin (1974; see also Fogel, 1992) made this distinction even more transparent by delineating a populational genotypic (information-containing) space $\mathcal{G}$, and a populational phenotypic (behavioural) space $\mathcal{P}$. Lewontin (1974) established four mappings taking elements in $\mathcal{G}$ and $\mathcal{P}$ into one another. These mappings are:

- $\rho$: $I \times \mathcal{G} \rightarrow \mathcal{P}$,
- $\varsigma$: $\mathcal{P} \rightarrow \mathcal{P}$,
- $\gamma$: $\mathcal{P} \rightarrow \mathcal{G}$,
- $\mu$: $\mathcal{G} \rightarrow \mathcal{G}$.

The mapping $\rho$, of \emph{epigenesis (ontogeny, development)}, maps the elements $g_i \in \mathcal{G}, i=1, 2, ..., n$ into the phenotype space $\mathcal{P}$ as a particular phenotype $p_i \in \mathcal{P}$, whose development is influenced by its environment $I$. The environment $I$ can be represented as the set of all relevant environmental influences $\{i_1, i_2, ..., i_k\} \in I$.

The mapping $\varsigma$, of \emph{selection}, maps phenotypes $p_i$ into $p_j$. Since natural selection operates only on phenotypes, the underlying genotypes $g_i, i=1, 2, ..., n$, are not operated upon by the mapping $\varsigma$. The function $\gamma$, of \emph{genotypic survival}, describes the effect of selection and migration processes on $\mathcal{G}$. The function $\mu$, of \emph{mutation}, maps the representative coding $g_i \in \mathcal{G}$ to the point $g_i^* \in \mathcal{G}$. This mapping represents a transformation of genotypes under the actions of mutation and recombination. Evolutionary adaptation occurs over successive iterations of these mappings.

The neo-Darwinian argument asserts that natural selection is the singular evolutionary force that prevails in shaping phenotypic characters of organisms in the vast majority of situations encountered in nature. Evolution occurs primarily through a statistical culling of inappropriate individuals. It is strictly an \emph{a posteriori} process which rewards each current success (Mayr, 1988, p.43). Natural selection leads to populations of increasing degree of adaptation. As has been highlighted previously in this text, the evolutionary forces of natural selection do not 'look ahead'; they only 'award' or 'penalise' \emph{current} performance and certain performances from the past. So far as this point of view is concerned, such forces really do act like a \emph{blind watchmaker}. 
6.1.5. The Evolution of Complexity: Minimising Surprise

At the outset we are faced with a difficulty: we have no way of measuring the degree of complexity of a structure. Thus although most of us would readily agree that the organs of a man are more complex than those of an amoeba, and those of an amoeba more complex than those of a bacterium, we have no agreed criteria on which to base this decision, and no way of deciding by how much one organism is more complex than another.

It may therefore seem odd to start formulating a theory of evolution by introducing a term which cannot be fully defined. However, I see no escape from doing so. *If organisms were not both complicated and adapted, living matter would not differ from dead matter, and evolution theory would have nothing to explain*.

Maynard Smith, 1969, pp. 82-83 emphasis added
As evolution proceeds, organisms of increasing complexity are generated. Mayr (1982, p.53) emphasises that:

"... each organic system is so rich in feedback, homeostatic devices, and potential multiple pathways that a complete description is quite impossible. Hence, a prediction of its productions is also impossible. Furthermore, the analysis of such a system would require its destruction and thus preclude the completion of the analysis."

For a wide class of systems, as the complexity of the entities under selection increases, the local optima that are attainable fall progressively closer to the mean properties of the underlying space of entities. This may imply that complex biological systems, such as genetic regulatory systems, are ‘close’ to the mean properties of the ensemble of genomic regulatory systems explored by evolution.

Observing an ant finding its way across the beach Herbert Simon (1963) exclaimed: "But the complexity is in the sand not the ant!"). And only few lines later Simon offers a more definitive and direct proposition:

"An ant, viewed as a behaving system, is quite simple. The apparent complexity of its behaviour over time is largely a reflection of the complexity of the environment in which it finds itself."

Continuing the similar line of reasoning, almost on the same page, Simon generalises and replaces the ant with a Homo sapiens, a thinking man, to conclude:

"... a human [or any organism in this respect] can store away a great furniture of information that can be evoked by appropriate stimuli. Hence, I would like to view this information-packed memory less as a part of the organism than as part of the environment to which it adapts.

... A thinking human being [an organism] is an adaptive system; man’s goals define the interface between his inner and outer environments, including in the latter his memory store. To the extent that he is effectively adaptive, his behaviour will reflect characteristics largely of the outer environment (in the lights of his goals) and will reveal only a few limiting properties of the inner environment - of the physiological machinery that enables a person to think.

... Specifically, I should like to point to evidence that there are only a few ‘intrinsic’ characteristics of the inner environment of thinking man [organism] that limit the adaptation of thought to the shape of the problem environment. All else is thinking and problem solving is artificial - is learned and is subject to improvement through the invention of improved designs and their storage in memory.

(emphasis added)

There is an enormous complexity in living systems - in a cell, an organ, an individual, a population, a species, an ecosystem, or a society. ‘Living systems are characterised by elaborate feedback mechanisms unknown in their precision and complexity in any inanimate
An interesting subject of speculation is to estimate how the total information transmitted to the present time in the biosphere has been distributed in the three channels. Otherwise, the separation of the three channels is devoid of interest. The current methodology (Margalef, 1968) comprehensively evaluates the genetic, ecological, and cultural channels of information transmission.

When comparing the genetic, ecological, and cultural channels of information transmission, Margalef (1968) notes that the genetic channel is the most significant in terms of information distribution. The ecological channel is also significant, but the cultural channel is the least significant of the three.

![Graph showing the distribution of information across genetic, ecological, and cultural channels.](image-url)
feedback circuits formed by interaction between species are a very expensive memory with a relatively limited capacity for storage. At the ecological level, contemporary forests do not convey much more information than mesozoic forests. Climatic fluctuations and other changes in the conditions of life have forced succession to start again and again in different places, and old information has not been saved in the form of the feedback circuits in ecosystems; rather, species have been replaced by others and the ecological memory seems always to have played the role of an auxiliary memory of rather limited capacity. The opposite is true of genetic information. Although sometimes latent for long periods of time, it has preserved increasing amounts of structure able to influence the future. The genetic channel has undoubtedly enlarged more rapidly than the ecological channel. Surely the cultural or ethological channel has become tremendously enlarged with the development of higher vertebrates, and its increase reaches the proportions of an explosion with the advent of the man. Now, if we project the relative size of the channels back in time (Figure 45), the total change is like a fan divided into three unequal parts: an ecological channel enlarging negligibly, a genetic channel enlarging considerably, and a cultural channel appearing later but enlarging explosively."

6.1.5.1. Benefits of Evolutionary Search

Adaptive evolution is, to a large extent, a process of complex combinatorial exploration. It is relatively facile to view such a process as a process of adaptive 'walks' realised through the generation of ever fitter variants in such a search-optimisation process. Adaptive walks proceed from an initial entity, through fitter neighbours and offsprings, to locally or globally optimal entities that are fitter than their neighbours.

Adaptation is partially determined by past history and by the nature of the evolutionary process itself. Adaptation generally progresses through small changes involving a sort of local search procedure in the space of all possible genetic rearrangements. In order to understand the evolutionary process, the constraints such a local search procedure places on any optimisation scheme must be understood. Moreover, by comparing local to more global search algorithms, some insight can be gained into how the nature of evolutionary change depends upon the genetic distances that exist between parents and offsprings.

"The main task of any theory of evolution (sic) is to explain adaptive complexity, i.e. to explain the same set of facts which Paley used as evidence of a Creator. Thus if we look at an organism, we find that it is composed of organs which are at the same time of great complexity and of a kind which ensures the survival and/or reproduction of their possessor. Evolution theory must explain the origin of such adaptations."

(Maynard Smith, 1969, p.82)

Selection eliminates sub-optimal phenotypes 'statistically', so to speak. While many of the evolved behaviours are transparent in their purpose, the question of the underlying reason for, and the ultimate beneficiary of these behaviours has remained an open one that has undergone considerable debate. To misunderstand the beneficiary of the evolutionary process is to misunderstand the essence of the process. There are at least three contentions: the
Early on, genes were conceptualised as particles linearly arranged on chromosomes. Recombination was explained by a crossing-over mechanism which broke the chromosomes apart at a point between these ordered particles and reassembled chromosome segments. Thus, the entire process was performed by an operator that was syntactically geared, with semantics being downplayed radically.

Currently, a gene is defined in biochemistry as the code necessary to promote the synthesis of one polypeptide chain. But it is also known that in eukaryotes a gene and any given segment of DNA are rarely linearly related. 'Any "single" gene, in the sense of a single read passage of DNA text is not stored in one place' (Dawkins 1986, p. 174). Because of complex and interwoven informational mappings, Lewin (1983) suggests that a gene should be defined through strong relation between one polypeptide and one gene, reversing the previous definition but allowing the functional product to define the gene. Because selection acts only on the phenotype, it is far more prudent to emphasise the functionality of the underlying genotype rather than its coding structure.

A second argument might assert that benefit accrues to the individual or its genome. But if the individual is the beneficiary of evolutionary optimisation, then the benefit does not long persist, as an individual's life span is minute when compared with any evolutionary time scale. Further, if the individual acts for its own benefit then many observed traits of altruistic behaviour still appear, which in turn appears paradoxical. Thus, it is indeed the flow of information that matters, not the flow of energy.

If evolution acts to optimise individual genotypes, then a much greater rate of clonal parthogenesis must be expected than is observed. Once the perfect genotype was found, the optimum strategy would be to copy it endlessly. But the overwhelming majority of species are sexual.

If evolution is to be viewed as a process in which advantage accrues to individuals, it is reasonable to ask 'Is it ... evolutionarily to each parent's advantage to form a monogamous pair bond and make a large investment in the young?' (Keeton 1988, p. 544). The question implicitly assumes that some benefit for creating offspring accrues to the parents. But that benefit must be identified. The argument presupposes that each parent has an interest in leaving offspring, typically the greatest possible number of offspring. But individuals must compete for survival with other members of the population and the most intense competition comes from the members of the same species who are, essentially, behaviourally identical. An overabundance of the species is a threat to all members of the species. The fundamental reproductive drive becomes an explanation after the fact if individuals are to be viewed as the ultimate beneficiaries of evolutionary invention. "Animals sometimes behave in a way that makes no sense from the point of view of maximising individual fitness" (Hartl and Clark 1989, p. 560).

\[45\] Clonal parthogenesis: reproduction by a female gamete without fertilisation, especially as a normal process in invertebrates and lower plants
Evolution is a learning process. Phylogenetic learning (i.e., learning arising from within the species) is the most commonly exhibited and most presumably the ancient form of intelligence. Following Weiner (1961), phylogenetic, ontogenetic and sociogenetic learning can be equalised, each containing an elementary unit of mutability and a storage of a learned behaviour. In phylogenetic learning, the elementary unit of mutability is a single nucleotide base pair. The repository of learned behaviour is the species’ genome. From such a perspective, the unit of selection is the individual, but the ultimate beneficiary of evolutionary invention is the species.

What has been previously characterised as paradoxical or even anomalous behaviours are more easily explained when the advantages of the one or the other optimisation are viewed as accruing solely to the reproducing populations. Behaviour that would appear altruistic can be more easily explained by assessing its impact on the reproductive success of the population. It is not unreasonable to expect that individuals of certain populations will become sacrificial, never mate and only serve to protect those individuals that do reproduce. Just as the individual’s fitness is a function of its entire cohesive genetic composition - and no genetic structure can be measured in isolation - the population’s fitness is a function of the entire cohesive set of behaviours of its individuals. When the appropriateness of the individuals’ behaviour is viewed outside the context of the species, behaviours often appear unexplainable or even irrational; these observations can generally only be reconciled when attention is focused on the species as a whole, functioning within its environment.

6.2. Naive Evolution

The idea of using a model of evolution for what are at least perceived as other purposes is not new and it is certainly in no way original to this work. The first ideas have been traced back as far as 1932, to the work of Cannon. Turing (1950), in his seminal paper on Computing Machinery and Intelligence, recognised already ‘an obvious connection between [machine learning] and evolution’, and continuing further (p.456):

"We cannot expect to find a good child machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications

Structure of the child machine = hereditary material

Changes of the child machine = mutations

Natural selection = judgement of the experimenter

One may hope, however, that this process will be more expeditious than evolution. The survival of the fittest is a slow method for measuring advantages. The experimenter, by the exercise of intelligence, should be able to speed it up. Equally important is the fact that he is not restricted to random mutations. If he can trace a
cause for some weakness he can probably think of the kind of mutation which will improve it."

It was some years later, in 1957, that Box presented the first working algorithm, Evolutionary Operation (EVOP). This statistical process control technique was applied to a manufacturing plant management process, relying basically on random variation (mutation) and selection. EVOP has been successfully applied to a variety of processes.

Friedberg (1958, Friedberg et al. 1959) experimented with a set-up of three computer programs: Teacher, Learner and initially Herman (1958), and later Ramsy (Friedberg et al., 1959) worked along similar, evolutionary lines. The initial efforts with Herman showed "limited success", but the achievements of Ramsy were more than encouraging. However, as with many other great accomplishments, Friedberg's results did not receive much attention from the contemporary scientific community.

Bremermann (1958) remarked that the 'principle of evolution ... is most useful as a key to the understanding of creative thinking and learning' and identified evolution with an optimisation process. In his initial experiments (1958) Bremermann used a binary string as a coding alphabet for evolving entities, the goal of the search being a certain target bit string, and mutation (bit flipping) was employed as the only transformation operator. The initial results none the less were promising. Later (1962), Bremermann considered the more realistic and interesting problem of function minimisation, and concluded that mutation only is not adequate to attain a global optimum. He conjectured that sexual mating might overcome the situations where asexual alteration of solutions is stagnant, but 'none of the schemes gave any spectacular results' (Bremermann, 1962). A subsequent investigation (Bremermann and Rogson, 1964) indicated that mutating in random directions resulted in better outcomes for linear programming problems. Bremermann et al. (1965, 1966; Bremermann, 1967, 1968, 1973) tested other kinds of mutations such as majority mating and crossing-over pairs of independently discovered solutions. No definitive benefit was obtained. Bremermann (1967) simply concluded that many biological species are at a genetic stagnation point, rather than at an optimum.

The inability to discover suitable computer programs rapidly, as experienced by Friedberg, itself became the target of considerable criticism. Minsky (1961) wrote an unfavourable review, calling this research direction a 'comparable failure'. Further, Minsky wrote that 'the machine did learn to solve some extremely simple problems. But it took on the order of 1000 times longer than pure chance would expect'. This, Minsky-characteristic exaggeration was played down in the closing sections of his report, where he admits that 'in the later stages of [Friedberg et al, 1959] we see some real success obtained by breaking the problem into parts and solving them sequentially'. Minsky termed the key problem of the contemporary attempts a 'Mesa Phenomena'46 (Minsky and Selfridge, 1961) claiming that 'In changing

---

46 The term Mesa Phenomenon identifies a certain class of problems where extrema in an evaluation function are separated by an un-traversable flatland, thus providing no information for a search algorithm about which direction to continue its explorations in order to recognise an improvement in performance. Stated differently, there are very few configurations that perform
just one instruction at a time the machine had not taken large enough steps in its search through program space' (Minsky, 1961).

In both, Friedberg’s and Bremermann’s experiments, when seen from today’s perspective, one can identify a number of causes for the poor performance of the evolutionary algorithms that were investigated. Maybe the most important of these was that the population size was limited to unity, thus failing to explore the power of parallel search. However, all of these efforts created a strong intellectual background generating research avenues that have resulted in modern, useful models of evolution.

6.2.1. Numerical Exploration of Naive Evolution

In the world of crafted design that surrounds us, it indeed requires a leap of imagination to prefer what must at least appear at first glance as blind chance to the masterminded solutions of engineers and designers. In order to demonstrate the principles of our subsequent, more extended experiments, let us first perform a particularly simple numerical experiment, trying to utilise the elements of biological knowledge presented earlier.

Initially, a population of individuals (potential solutions) is created in their genotypical form. The simplest possible way of creating these entities (or proto-organisms) in such an introductory example is to create them randomly. The population size can be fixed, say to successfully, and there are no intermediate configurations between these.
The population consists of fixed-length strings, and the strings are composed over a binary alphabet, i.e. $I_c = \{0,1\}^l$, such as, with $l = 12$, for example:

One can view the strings from Figure 47 as chromosomes carrying the genes which are turned on (1) or off (0). In order to relate genotype and phenotype, one has to employ a mapping function of a kind that will correspond to the action of Lewontin's epigenesis, $\rho$ (sub-section 6.1.4). This process is referred to as an ontological development and it is dependent on the environmental conditions in which the organisms are located and develop. However, for the current purposes of testing a simplified model of evolution, and since the model of environment currently used is a steady one (i.e. no changes in the environment are supposed to occur during the evolutionary process), we can assume that the influence of the environment can be neglected altogether.

For pseudo-boolean objective functions, the representation can be used directly, whereas for real-valued problems, the classic example of such a mapping is the binary decoding:

$$\Gamma(a_{i1}, a_{i2}, ..., a_{in}) = \sum_{j=1}^{l} a_j 2^{i-1}$$

The bit string is in this case divided into $n$ segments of (usually) equal length (i.e. $l = m \cdot I_n$), and each segment is interpreted as the binary code of the corresponding object variable $a_i$ on a real-valued segment $[u, v]$. A segment decoding function then has typically the appearance of above equation.

There is obviously a close analogy between this binary-string coding and the writing of 'text' in the discrete alphabet of nucleotide bases which form DNA (Figure 40). Therefore, it could be argued that it is very close to what we currently perceive as the 'natural' paradigm. Moreover, theoretical considerations of the evolutionary algorithm performance (corresponding to the schemata theorem, see sub-section 6.5.5.) suggests the advantage of using alphabets of low cardinality. In this case, the smallest cardinal number of a meaningful alphabet corresponds to $k = 2$, since there are only two semantically correct symbols $\{0,1\}$.

Imagine now that each of the organisms, currently exemplified by binary strings, is a phenotypical individual, with only two traits - real valued variables denoted by $x$ and $y$. For instance, the first six binary digits might code for $x$ and the next six for $y$, so that each of these variables could take any real value between -4 and +4, giving $x \in [-4, +4]$ and $y \in [-4, +4]$. One has to apply a mapping function (the epigenesis, $\rho$) in order to provide a genetic, binary code with a real-valued, phenotypical utility, or to map the individual's genotypical form onto its phenotypical form. Let us take the first binary string (chromosome) from the population of genotypes (Figure 47) and try to decode it using introduced mapping, as depicted in Figure 48.
Figure 48 Binary string representing genotype (a) is interpreted into real-valued utilities representing corresponding phenotype (b).

We could now say that the genotype of Figure 48.a is mapped, or decoded, into the phenotype of Figure 48.b. We can regard this semiotically as a transformation from one set of sign vehicles to another set of sign vehicles, to the one of which we can affix the label of 'genome' and to the other the label of 'phenotype'.

Now we can progress one step further again. At this stage a 'measure of strength' of each individual entity, exemplifying its performance in the environment, must be established. Therefore, at this stage, the values of the phenotypic traits ($x$ and $y$) must be used to establish the fitness of a particular individual.

Figure 49 Rosenbrock function: Contour plot and perspective view

Let us assume, in this introductory example, that the fitness of the individual entities from
Figure 47. can be expressed by the much-used Rosenbrock function$^{47}$:

$$F(x,y) = 100 \cdot (x^2 - y)^2 + (1 - x)^2$$

Then the fitness of our sample genome is clearly equal to:

$$F(-4.00, -1.079) = 29195.47$$

The same procedure can be applied to the entire population. Therefore, Figure 47, which depicts a genotypical view on the population only, can be re-drawn to demonstrate the phenotypical traits exclusively (Figure 48.a), and thence to determine the corresponding fitnesses of every individual (Figure 48.b).

We observe that the criteria of fitness in nature commonly change in time, so that a best fitness at one time does not correspond to the best-fitness at another time, as, for example, the ‘environment’ may have changed or the creature may be hibernating, or may have become more mature, or whatever. In the case of the change in environment, which plays such a prominent part in Darwin’s work, the genetic algorithm provides a methodology for tracking the highest levels of fitness through the environmental change. We then speak of a genetic drift, with the corresponding development of a new species and all that follows from this.

What we have just described corresponds to one cycle in the development of genotypes to phenotypes. There is still no mention of reproduction and population growth, and therefore there is no competition and no natural selection.

Through the process of reproduction, similarly to the natural conditions (although very much simplified), the more strong (fit) individuals are those that get more and better chances to reproduce. Analogously in this simple experiment, a procedure must be invented that will ‘select’ individual organisms which will participate in reproduction, while promoting the more fit to participate more often in the reproductive processes. As in nature, this selection algorithm need not be a deterministic one. Reproduction depends on such factors as the number of encounters among individuals, their spatial distribution, their past experiences, and so on, most of which are much too complicated to concern us in such an introductory experiment. In order to capture some part of this process, however, the selection algorithm can be provided with one or the other kind of stochastic priority according to which the individuals will form couples in order to reproduce.

$^{47}$ The Rosenbrock function provides a difficult surface for gradient optimisation. It has a parabolic valley along the curve $y=x^2$ with a unique minimum of zero at the point $(1,1)$. In order to 'convert' the problem of function minimisation to fitness maximisation, one can simply invert the values calculated through application of the Rosenbrock function and assign it to individual fitness; thus:

$$\text{Fitness}(x,y) = \frac{1}{\text{Rosenbrock}(x,y)}$$
### Table

<table>
<thead>
<tr>
<th>ID</th>
<th>x</th>
<th>y</th>
<th>Rosenbrock</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-4.000</td>
<td>-0.108</td>
<td>25971.44</td>
<td>3.85E-05</td>
</tr>
<tr>
<td>2</td>
<td>0.444</td>
<td>0.064</td>
<td>2.08</td>
<td>0.480387</td>
</tr>
<tr>
<td>3</td>
<td>1.333</td>
<td>0.444</td>
<td>177.77</td>
<td>0.005625</td>
</tr>
<tr>
<td>4</td>
<td>1.206</td>
<td>-2.730</td>
<td>1750.99</td>
<td>0.000571</td>
</tr>
<tr>
<td>5</td>
<td>3.238</td>
<td>0.190</td>
<td>10602.98</td>
<td>9.43E-05</td>
</tr>
<tr>
<td>6</td>
<td>0.444</td>
<td>0.064</td>
<td>2.08</td>
<td>0.480387</td>
</tr>
<tr>
<td>7</td>
<td>1.333</td>
<td>0.444</td>
<td>177.77</td>
<td>0.005625</td>
</tr>
<tr>
<td>8</td>
<td>-3.238</td>
<td>-1.206</td>
<td>13685.08</td>
<td>7.31E-05</td>
</tr>
<tr>
<td>9</td>
<td>-3.238</td>
<td>-1.206</td>
<td>13685.08</td>
<td>7.31E-05</td>
</tr>
<tr>
<td>10</td>
<td>2.730</td>
<td>-2.984</td>
<td>10895.88</td>
<td>9.18E-05</td>
</tr>
<tr>
<td>11</td>
<td>-0.190</td>
<td>-3.619</td>
<td>1337.39</td>
<td>0.000748</td>
</tr>
<tr>
<td>12</td>
<td>-3.238</td>
<td>-1.206</td>
<td>13685.08</td>
<td>7.31E-05</td>
</tr>
<tr>
<td>13</td>
<td>1.333</td>
<td>0.444</td>
<td>177.77</td>
<td>0.005625</td>
</tr>
<tr>
<td>14</td>
<td>2.603</td>
<td>3.873</td>
<td>845.08</td>
<td>0.001183</td>
</tr>
<tr>
<td>15</td>
<td>3.619</td>
<td>3.873</td>
<td>8515.37</td>
<td>0.000117</td>
</tr>
<tr>
<td>16</td>
<td>1.206</td>
<td>-2.730</td>
<td>1750.99</td>
<td>0.000571</td>
</tr>
<tr>
<td>17</td>
<td>0.444</td>
<td>0.063</td>
<td>2.11</td>
<td>0.474297</td>
</tr>
<tr>
<td>18</td>
<td>3.619</td>
<td>2.984</td>
<td>10234.46</td>
<td>9.77E-05</td>
</tr>
<tr>
<td>19</td>
<td>3.238</td>
<td>0.190</td>
<td>10602.98</td>
<td>9.43E-05</td>
</tr>
<tr>
<td>20</td>
<td>3.619</td>
<td>3.873</td>
<td>8515.37</td>
<td>0.000117</td>
</tr>
</tbody>
</table>

### Figure 50
Overview of the phenotypical traits of an initial population, together with fitness values. The average fitness of the population amounts to 0.051, whereas the fitness of the 'strongest' individual amounts to 0.4804

### Figures 51 and 52
Distribution of Rosenbrock function and fitness values within the evolving population - a biased roulette wheel.
Figures 51 and 52 illustrate graphically distributions of fitness and function values within the population introduced in Figure 50. This information is used to form a basis for a ‘natural selection’ model to be used in our elementary experiment. The illustration of fitness distribution in particular suggests a comparison of the selection mechanism to the playing of a game with a biased roulette wheel (Goldberg, 1989). As has been already suggested, the selection algorithm should be non-deterministic. Thus, in order to select an individual that will survive to the next generation and subsequently be manipulated by processes simulating asexual and sexual reproduction, the selection algorithm should generate a number ‘randomly’, or it should, analogously, roll a dice on a biased roulette wheel.

Since the more fit individuals come to occupy larger ‘slices’ on the ‘wheel’, it is more likely that they will be selected to participate in subsequent processes of genetic modification. Through such a repetitive process, the more fit individuals acquire, on the average, greater chances to be selected to participate in the reproduction processes that will follow; thus the information that they convey as code is more likely to be propelled to the next generation in such an evolutionary process. It is quite obvious from Figure 55 that entities 2, 6 and 17 in the present example have excellent expectations in such a selection procedure. However, it must be observed that the possibilities of the remainder of the population persist as well: they could just be ‘lucky’.

The selected entities will have to undergo a sequence of unary (e.g. mutation-type) and higher-order (e.g. crossover-type, or sexual-reproduction-type) transformations. In our simplest explanatory experiment, the genotypes are built from only one strand of genes. Therefore, the simplest procedure for creating new entities in a way that resembles sexual reproduction is that depicted in the Figure 53. Two selected chromosomes are aligned, a crossover point is randomly selected, the chromosomes cut into two and then interchanged. The performance of the mutation operator is illustrated in Figure 54. A randomly selected bit is picked and its value is altered (from 1 to 0, or from 0 to 1).

Application of these operators guarantees two major prerequisites for an evolutionary process, namely those of heredity and variability, whereas the third (and last) prerequisite - fecundity - is guaranteed by the application of the already described selection mechanism through which more fit individuals are assigned a higher probability to reproduce.
<table>
<thead>
<tr>
<th>ID</th>
<th>x</th>
<th>y</th>
<th>Rosenbrock</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.190</td>
<td>0.063</td>
<td>0.73</td>
<td>1.371042</td>
</tr>
<tr>
<td>2</td>
<td>0.444</td>
<td>0.063</td>
<td>2.11</td>
<td>0.474996</td>
</tr>
<tr>
<td>3</td>
<td>0.952</td>
<td>0.063</td>
<td>71.16</td>
<td>0.014053</td>
</tr>
<tr>
<td>4</td>
<td>0.444</td>
<td>0.063</td>
<td>2.11</td>
<td>0.474996</td>
</tr>
<tr>
<td>5</td>
<td>0.444</td>
<td>-3.619</td>
<td>1456.94</td>
<td>0.000686</td>
</tr>
<tr>
<td>6</td>
<td>-3.238</td>
<td>-1.079</td>
<td>13392.02</td>
<td>7.47E-05</td>
</tr>
<tr>
<td>7</td>
<td>-0.317</td>
<td>-2.730</td>
<td>803.16</td>
<td>0.001245</td>
</tr>
<tr>
<td>8</td>
<td>1.333</td>
<td>-3.746</td>
<td>3051.36</td>
<td>0.00328</td>
</tr>
<tr>
<td>9</td>
<td>0.571</td>
<td>0.825</td>
<td>25.07</td>
<td>0.039888</td>
</tr>
<tr>
<td>10</td>
<td>0.190</td>
<td>0.063</td>
<td>0.73</td>
<td>1.371042</td>
</tr>
<tr>
<td>11</td>
<td>0.190</td>
<td>0.063</td>
<td>0.73</td>
<td>1.371042</td>
</tr>
<tr>
<td>12</td>
<td>2.730</td>
<td>-2.857</td>
<td>10634.48</td>
<td>9.4E-05</td>
</tr>
<tr>
<td>13</td>
<td>3.746</td>
<td>3.619</td>
<td>10852.07</td>
<td>9.21E-05</td>
</tr>
<tr>
<td>14</td>
<td>1.333</td>
<td>-3.746</td>
<td>3051.36</td>
<td>0.00328</td>
</tr>
<tr>
<td>15</td>
<td>1.206</td>
<td>1.333</td>
<td>1.53</td>
<td>0.653749</td>
</tr>
<tr>
<td>16</td>
<td>1.206</td>
<td>1.333</td>
<td>1.53</td>
<td>0.653749</td>
</tr>
<tr>
<td>17</td>
<td>0.190</td>
<td>0.063</td>
<td>0.73</td>
<td>1.371042</td>
</tr>
<tr>
<td>18</td>
<td>3.746</td>
<td>3.619</td>
<td>10852.07</td>
<td>9.21E-05</td>
</tr>
<tr>
<td>19</td>
<td>3.746</td>
<td>3.619</td>
<td>10852.07</td>
<td>9.21E-05</td>
</tr>
<tr>
<td>20</td>
<td>0.444</td>
<td>-3.619</td>
<td>1456.94</td>
<td>0.000686</td>
</tr>
</tbody>
</table>

**Figure 55** Overview of the phenotypical traits of the population in the second generation. The average fitness of the population amounts to 0.39, whereas the fitness of the strongest individual has increased to 1.371.

Thus, repetitively applying these operations - *select entities from initial population, mutate selected entities, crossover selected entities* - the computation continues until a new 'population' of the same size, namely 20 in this case, is created. One iteration over these operators is referred to as one generation. The figure above illustrates the results of applying these mechanism for the same sample population as was introduced in Figure 55.

After a few iterations, the sampling of the search space and the exchange of information among individual structures (through application of crossover) produces convergence to (hopefully) a global optimum. Note that the environment is, otherwise than in natural evolution, kept constant. This is, of course, due to the fact that in such, steady, conditions, the search for an 'optimal' structure is easier and more likely to be successful. The following figures, Figure 56, and Figure 57, summarise the evolutionary process of the most successful entities through an application of such a simple procedure.

Stated concisely, the evolutionary algorithm introduced in this way maintains a population of structures that evolve according to rules of selection and other operators that are referred to as 'search operators', such as those of recombination and mutation. Each individual in the population receives a measure of its performance in the environment: its fitness. This process can be summarised in a form of a pseudo-code for an evolutionary algorithms that can be readily implemented on a computer (see Figure 58).
There are already a few lessons to be learned from the numerical experiment just performed. Firstly, our evolutionary algorithm does not reach the unique minimum of the Rosenbrock function, of 0, but only the value of 0.208. This is due to the rather coarse representation adopted for the genotype coding. For example, the domain variables $x \in [-4, +4]$ and $y \in [-4, +4]$ are each discretised in $2^6 = 64$ segments. Such a discretisation does not allow for coding of, for example, $x=1$ and $y=1$. The nearest values are $x=0.953$ and $y=0.953$, and these are indeed the coordinates that our algorithm detects in the generation 16; thus the search task is successfully performed given the limitations of the underlying genetic coding. This clearly demonstrates the weakness introduced through the representation of the infinite nature of the real-world as a finite, in this case genetic, structure. This problem represents one of the most fundamental difficulties of the computer sciences, namely that the mapping of a set of features with infinite cardinality onto a similar set of features with only a finite cardinality introduces a sometimes indispensable truncation error which 'contaminates' the mapping to such a degree that it quite alters the nature of the problem that is being solved. Thus, the choice of the computer-representation of the domain that is being solved is crucial for successful problem-solving. This will become even more apparent in the sections that follow when the issues of knowledge representation within knowledge induction and machine learning paradigms using evolutionary principles are addressed.

Secondly, the evolutionary algorithm, as introduced through the performed experiment, is depicted as a randomised search procedure. The selection mechanism contains certain random components; both crossover and mutation have probabilistic aspects and utilise random processes. One consequence of such a randomised nature of evolutionary algorithms is that two consecutive evolutionary experiments could show quite different behaviours, such as might be exemplified through different convergence rates, for example. While this ‘unpredictability’ and ‘un-reliability’ in performance are certainly not the most welcome features from a purely engineering point of view, they do provide certain benefits as well. Due to such a unique, randomised sampling of a problem domain, it is almost impossible for such an algorithm to get trapped in local optima in the solution space. Evolutionary algorithm will not have difficulties even with discontinuous response surfaces. At the same time, and again from a purely
engineering point of view, evolutionary algorithms produce intermediate results during their running, exemplified as the most fit entity in every generation. Thus, unlike the more traditional techniques, which have to be worked out completely in order to produce a result, the evolutionary algorithm evolves the solution continuously. In certain domains, like real-time control, where the CPU available to perform certain task is not always guaranteed, such an access to intermediate results can be indispensable. Thus, evolutionary algorithms have the virtue, from a purely engineering perspective, that they are, so to say, ‘reliable’.

**Procedure Evolutionary Algorithm**

begin

\[ t = 0; \]
\[ \text{initialise Population}(t); \]
\[ \text{evaluate Population}(t); \]
while termination condition not satisfied

begin

\[ t = t + 1; \]
\[ \text{select Population}(t); \]
\[ \text{mutate structures in Population}(t); \]
\[ \text{crossover structures in Population}(t); \]
\[ \text{evaluate structures in Population}(t); \]

end.

**Figure 58 A Pseudo-code for an Evolutionary Algorithm**

It is interesting to observe how effective even this sampling of a search space can be. The underlying genotype coding discretises the universe of discourse in \[ 2^{12} = 4096 \] distinctive positions or configurations. In the previous experiment, the most profitable has been detected after 16 generations, and thus after \[ 16 \times 20 = 320 \] evaluations of the fitness function. Thus some 8% of the search space has been examined in order to ‘evolve’ the most fit entity. This is a rather impressive performance for such a simple, randomised search procedure. As has been demonstrated; the initialisation of the algorithm and the selection procedure and the application of search operators (crossover and mutation) are algorithmically extremely simple. The largest ‘expense’ of the entire process is in the calculation of the fitness function. Accordingly, the trade-off between base level (evaluation of fitness function) - and the meta level (selection and search operators procedures) seems already to be very acceptable. This statement will be quantified in the subsequent sections.

It is quite obvious that evolutionary algorithms belong to the weak-method class of the search techniques in that no assumptions are introduced about the problem-domain that might influence the problem-solving process. The only conjecture introduced is in the ‘representational phase’ which presumes the mapping function that can represent objects in our outer world as binary strings. This has a number of consequences, one of which has already been described.

Moreover, the evolutionary algorithm performs its search on the fitness landscape itself, rather than using auxiliary pieces of information. Other search methods, typically those of navigating through the search space by the generation and application of supplementary knowledge (like calculating derivatives of the fitness function in the case of so-called hill-climbing procedure, for example) lack this primitiveness. As has been mentioned earlier, the application of the stronger methods might prove
to be useful and result in the acceleration of the convergence rate (provided the meta-level activity remains relatively effortless). However, the assumptions should hold (such as, again in the case of hill-climbing, the assumptions of the continuity and differentiability of the function that is being optimised). Such demands are often rather exigent in real-world application, thus justifying the use of weaker, but more robust, and still relatively successful search methodologies.

The last lesson to be learned from our first numerical experiment regarding simulated evolution is that it is only the fitness evaluation function (inverted Rosenbrock function in our case) that makes the link to the problem-domain. The researcher with imagination can devise many and various ways to evaluate fitness, and can search for solutions in different domains correspondingly. In the sections that follow we shall demonstrate the evolutionary principles operating on domains as diverse as knowledge base induction, the creation of various numerical models, the calibration of performance of hydrodynamic and some other models, the generation of connectionist networks and some others besides, thus demonstrating the wide applicability of such a weak method that performs well over many problem domains.

In the continuation, we take a more conventional approach to defining evolutionary algorithms and develop a formal methodology for a broad class of evolutionary algorithms: genetic algorithms (GA), evolution strategies (ES), evolutionary programming (EP), learning classifier systems (LCS) and genetic programming (GP). Thus, firstly, a framework for defining evolving entities is introduced together with some critical notions regarding so-called fitness landscapes with a purpose of establishing a link with the search techniques that have already been introduced. In the later sections of this chapter we describe the most commonly used genetic operators more formally, emphasising the most distinctive features of the Evolutionary Algorithm (EA) variants.

6.3. Genetic Representations

We now develop a formal framework for describing entities that are evolving. Following Radcliffe and Surry (1994), a genotypic representation of an individual chromosome is a string of n genes. Each gene can take on values from some (typically but not necessarily finite) set $A_i$. Thus, the genotypic representation space will take a form:

$$\Omega = A_1 \times A_2 \times A_3 \times \ldots \times A_n$$

so that a chromosome $\omega \in \Omega$ is formally a vector of gene values.

In classical Genetic Algorithms (GA), as introduced by Holland (1975), the elements or members of sets $A_i$ typically take binary values, thus:

$$\forall a_{ki} \in A_i, \ a_{ki} = \{0, 1\}$$

In many other cases, however, $A_i$ members can be any real-valued number (see, for example, Babovic et al., 1994).

In Evolution Strategies (Schwefel, 1981), the members of sets $A_i$ are typically real-valued utilities:
\[ \forall a_k \in A_i, \quad a_{ki} = \{ \mathbb{R} \} \]

The \( A_i \) components in *Evolutionary Programming* (Fogel et al., 1966) are indeed very broadly defined and can be adapted to any problem at hand, from numerical optimisation (Fogel, 1992) to evolution of finite state automatons (Fogel, 1993) and connectionist networks (Fogel and Porto, 1993).

In *Genetic Programming* (GP) (Koza, 1992), \( A_i \) members are algebraic operators, like:

\[ A_i = \{ \text{signum; sin; \sqrt{}; exp; ln; \text{-}; \text{+}; \text{*}; \text{/}; \text{^}; \text{not}; >; <; =; \text{T}^\text{^}; \text{if-then-else} \} \]

Following Weismann’s deductions (Section 6.1.2.1), such a chromosome, which corresponds to a genotypic form of an individual entity, must be transcribed into the phenotypic form of the individual. In the case of Genetic Algorithms, this typically corresponds to a mapping of a binary string into a number of parameters (Babovic, 1993) applying some sort of interpretation:

\[ \rho : \Omega \rightarrow \Xi \]

where \( \Xi \) denotes a search space of phenotypes, representing a set of potential solutions to the problem and \( \xi \in \Xi \) is a vector in a solution space.

Let \( \psi \) be a *fitness function*, which associates a real valued measure of an entity’s ‘adaptedness’ to the environment as a mapping:

\[ \psi : \Omega \rightarrow \mathbb{R}^+ \]

where \( \mathbb{R}^+ \) is a set of non-negative real numbers.

The problem of evolutionary adaptation then becomes one of identifying extrema of the mapping above, and associated structures \( \omega \in \Omega \). We shall identify these global optima as a set \( \Omega^* \subset \Omega \). It should be carefully noted that although our search is carried out on a fitness landscape, and that indeed the sampling of the search space is performed on the basis of phenotypes \( \xi \in \Xi \) in order to identify all global optima \( \Xi^* \subset \Xi \), there could be more than one chromosome satisfying the conditions of global optimality, so that the evolutionary process should generate genotypical structures \( \omega \in \Omega \). This indeed closely resembles the situation observed in nature: the forces of natural selection in reality act on the phenotypes, but it is the (genetically) *coded information-flow of genotypes*, that really matters when propelling advantages on evolutionary time-scales.

### 6.4. Fitness Landscape

The *fitness landscape*, the term introduced by Sewell Wright (1932), is now considered to be one of the most useful analogies in evolutionary genetics. It provides a visualisation of the mapping between the genotypes of the population and their fitnesses, together with the
application of the search operators that influences the ruggedness of this mapping.

The properties of fitness landscapes are extremely important in that they reflect the difficulties with which the search process is faced in a way that is immediately intuitive. The more rugged the landscape, the more difficult the problem. Many problems are indeed locally or even globally discontinuous, forcing the individual to get trapped in local optima. This clearly introduces difficulties that traditional techniques might encounter, but which evolutionary algorithms may avoid.

Fitness is a notion broadly used within the evolutionary algorithms. However, it can easily be equated with some other concepts that are used in operations research or artificial intelligence, such as utility or credit assignment. At this place we shall simply equate fitness with some reasonable function which is being optimised by means of various methods. Thus, the exploration of the fitness landscape can be understood in the most narrow sense as one of solving an optimisation problem but in the more general sense as one of exploring a fitness landscape. It should be understood as a general purpose search mechanism which is the key process in the modelling intelligent behaviour.

In the continuation we shall develop a landscape model that is simply a labelled graph. Most of the ideas and terminology presented here are due to Pearl (1984) and Jones (1994, 1995). In the most general form, a landscape, \( \mathcal{L} \), is a simply a directed graph \( \mathcal{G}(\mathcal{V}, \mathcal{A}) \) whose vertices \( \mathcal{V} \) are labelled with real numbers and whose arcs (edges, links) \( \mathcal{A} \) are labelled with certain weights. The weights are normalised, in the sense that the sum of weights on the edges leaving any vertex is 1. Any such graph shall be called simply a landscape.

If one performs any form of computation on the raw object space, \( \mathcal{G} \), one constructs (or more correctly, the computational method constructs) a representation of the objects in their space. Following the nomenclature relating phenotype and genotype spaces (sub-section 6.1.4), this representation determines a representation (behavioural) space \( \mathcal{P} \).

\[ \xi \xrightarrow{\rho} \psi \xrightarrow{\mathcal{P} \circ \mathcal{G}} F(\xi) \]

*Figure 59* Schematisation of a composite mapping relating genotypes, phenotypes and associated fitness values
A short notational explanation should be inserted here. \( \mathcal{G} \) and \( \mathcal{P} \) denote object spaces and introduce broader notions than do the sets \( \Omega \) and \( \Xi \). In fact, the set of genotypes, \( \Omega \), and the set of phenotypes, \( \Xi \), are only subject to the conditions that \( \mathcal{G} \subset \Omega \) and \( \mathcal{P} \subset \Xi \), since the spaces \( \mathcal{G} \) and \( \mathcal{P} \) include the operators modelling the interactions among the objects in these spaces.

The objects in \( \mathcal{G} \) are usually of interest for some reason or reasons, and we will suppose that the degree to which an object is interesting or desirable can be expressed as a single real value. Within the EA framework, this is commonly called the *fitness* of the object in question. Since the mapping of epigenesis has 'converted' genotypes, \( \omega \), to phenotypes, \( \xi \), and the fitness calculation is based on the behaviour of the phenotype rather than on that of the genotype, it is admissible to denote the fitness of an object \( \xi \in \Xi \) as \( F(\xi) \) despite the fact that the fitness value is based on the underlying genetic coding.

It must be emphasised that the visualisation of the composite mapping \( \rho \circ \Psi \), differs from the visualisation of the fitness landscape. As will become more apparent later, the formation of a fitness landscape depends on the *search operators* applied, so that one may say that the landscape 'undulates' under the actions these operators, or that it alters during the search process depending on the operators that are applied. Such a model proves to be extremely useful in building up an understanding of the endeavours of search methods that are non-deterministic.

The value of fitness may be difficult to calculate and can involve the utilisation of rather expensive (in CPU terms) processes. Classical AI refers to this process of fitness calculation as being similar to *credit assignment* (Minsky, 1963). In order to generalise the notion in this way, one has to expand it so that it highlights the issues that arise when transforming the feedback from the problem domain and so that it manipulates a knowledge structure that is internal to the problem solver. Credit assignment typically refers to the rating of a structure's component as being beneficial or harmful to solving the problem. The term is most popular within the machine learning paradigm, but since search algorithms use a form of credit assignment in making decisions about which element of the search space is to be visited next, the notion is used here as well.

At this point, a representation space and a function that assigns values to points in that space are specified. The notion of the topology of the landscape evokes sensations that are already conceptually close to many of us. Accordingly, it is often easier to view and solve the problem if it is understood as a 'walking' from point to point. In order to define the landscape more fully, one must, however, also introduce the notion of a *neighbourhood*. This is necessary because it is only the relation between neighbouring points that evokes the powerful imagery of a landscape. The peak, an optimum in such a vocabulary is then understood as a vertex which is 'higher' than all of its neighbours. Thus, the *conditio sine qua non* in the definition of landscapes is a definition of a neighbourhood.

The neighbourhood of an object is principally defined by the positions (locations of other objects) to which the object is connected and to which it can progress by a single application of an operator \( \phi \). According to the definition of the landscape adopted by Jones (1995), connecting edges are labelled with a certain weight. These weights in turn are defined by the
operator $\phi$. This operator is best thought of as a process which, when given the vertex $v \in \mathcal{V}$ as input, produces the set of possible outcomes, each of which is a vertex in $\mathcal{B}_\phi$, and each of which has a certain weight. These weights can be understood, albeit in a somewhat different vein, as awards or penalties representing either costs or rewards associated with their inclusion in the final solution. The action of $\phi$ is best thought of as one of a node expansion. Once all the neighbours of the node $v$ are defined, the node $v$ is said to be ‘expanded’.

By far the most critical element of the graph exploration process is that of vertex generation. This corresponds to computing the representation of a node from that of its ancestor, and in evolutionary circumstances this corresponds to the actions of the search operators of mutation and crossover. The (possibly infinite) set of outcomes (vertices) that the operator $\phi$ may produce on an input $v$ will be referred to as the neighbourhood of $v$ under $\phi$, and will be denoted by $\mathcal{N}_\phi(v)$ (Jones, 1995). If $\phi$ produces $w$ from input $v$, we write $\phi(v)=w$. If $w \in \mathcal{N}_\phi(v)$ then $\mathcal{B}_\phi$ will contain a directed edge $(v, w)$ labelled with the probability that $\phi(v)=w$. Thus, the edges of the graph $\mathcal{B}_\phi$ are induced by the operator $\phi$, as indicated by the subscript. The number of successors (number of neighbours) emanating from a given vertex $v \in \mathcal{V}$ is called the branching degree of the vertex and is normally denoted as $B^\phi(v)$. The landscapes where $\mathcal{S}^\phi(v) < \infty, \forall v (v \in \mathcal{V})$ are referred to as a class of a locally finite graphs.

As Jones (1995) eloquently describes in his doctoral thesis, a correct and complete model of a fitness landscape implies that a particular operator will form a particular landscape. For example, a simple bit-flipping mutational operator transforming a binary string $A \in \{0,1\}$ may induce a landscape as presented in Figure 60. Probabilities along the edges are equal to $1/l$, where $l$ equals the length of the binary string.

![Figure 60 Fitness landscape induced by a bit-flipping operator](image)

Application of a more general mutational operator that may affect more than a single bit at
a time may result in a more rich landscape formation (Figure 61). In this case, for a binary alphabet, there is $2^i$ possible neighbours.

This definition of a landscape allows for the existence of operators $\phi$ which take more than one vertex as input and produce more than one vertex as output. Crossover operators then form even more diverse landscapes that then allow for distant 'jumps' on the search space.

![Figure 61 A part of fitness landscape induced by more general mutational operator](image)

EA-based search, then, is carried out on all of these landscapes simultaneously, making use of different connectivity structures that are formed under the applications of different operators. Other constitutive elements with EA algorithms, like population size, probabilities of applications of various genetic operators (and consequently, even if only implicitly, of various generations of corresponding landscapes), generation mechanisms of initial population, collectively provide a meta control that is directly responsible for the success of the navigation on process over the fitness landscape.

It follows that the fitness landscape cannot be simply equated to a particular visualisation of the fitness function. The fitness landscape is a more complex structure which takes into account inter-connectivity among phenotypes provided by operators which examine the search space.

Using a more biological vocabulary, each peak in such a fitness landscape corresponds to an optimised phenotype, and thus one or more optimised genotype(s). The topography changes
its shape over time as a function of organism-environment interactions. Wright (1932) describes evolution as a 'shifting balance'. As the topography shifts, the allelic balance in the population changes as a result of a selection process performed on the corresponding phenotypes. Evolution proceeds probabilistically up the slopes of the topography toward the peaks as selection sorts-out inappropriate phenotypic variants.

Many authors (Wright, 1960, Templeton 1982, Raven and Johnson, 1986) suggest that it is more appropriate to view an adaptive landscape from, so to say, an inverted position. The peaks become 'minimised prediction error entropy wells' (Atmar, 1986, see also Fogel, 1992). Such a viewpoint is intuitively appealing. Searching for peaks depicts evolution as a slowly advancing, tedious and uncertain process. Moreover, there appears to be a certain fragility to an evolving phyletic line; an optimised population might be expected to fall quickly from the peak under slight perturbations. The inverted topography leaves an altogether different impression. Population advances rapidly, falling down the error wells until completely optimised, at which point stagnation occurs. If the topography is generally static, rapid descents will be followed by long periods of stasis. If, however, the topography is in continual flux, stagnation may never set in at all.

The inverted adaptive landscape stresses the total system behaviour (in terms of functional error). Equivalent phenotypes may have different underlying genotypes and yet have the same predictive error. Indistinguishable complex behaviours can be encoded by very different genetic programs. The structure of the genetic code itself is generally irrelevant.

Selection drives phenotypes as close to the optimum as possible, given the initial and boundary conditions. Note that the environment is constantly changing. Species lag behind, continually evolving to a new optimum. No organism should be regarded as being perfectly adapted to its environment. The sub-optimality of behaviour should be expected in any dynamic environment. Selection never ceases to operate regardless of the population's position on the topography.

Some phenotypic characteristics are deeply ingrained in the developmental sequence of the organism. They are historical with an own historicity. Other adaptations are maintained by a constant pressure of natural selection, without which they would degenerate due to random genetic drift (Hecht and Hoffman, 1986).

All populations undergo selection continuously for adaptations to the current environment. But each also carries an evolutionary history of adaptation to past environments. 'Evolution is opportunistic, and natural selection makes use of whatever variation it encounters' (Mayr 1988., p.153). 'Natural selection does not work like an engineer. It works like a tinkerer' (Jacob, 1977). Evolution can be expected to lead both to 'organs of extreme perfection' (Darwin, 1859) and to global sub-optimality. At the same time, the organs and organisms that appear sub-optimal in one environment may contain the seeds of successors that are optimal under another environment again. Functionally critical behaviours may be expected to be extremely well-optimised. Behaviours that are less critical may be expected to vary to a greater degree.

The adaptive landscape undulates in response to environmental variation. Adaptation is in
response to past variations, and it is only in this sense that it can be said to anticipate the future. Furthermore, the fitness of a genotype may depend upon its frequency in the population, or upon co-evolving properties of other species or habitat characteristics. Thus ‘fitness’ is not a property of a genotype alone, but depends upon its environmental context.

These caveats having being made, it remains the case that the powerful imagery created by Wright’s adaptive landscape and Fisher’s Fundamental Theorem of Natural Selection are amongst the most important concepts in evolutionary theory. When the genetic system is simple, involving a single locus or a weak epistasis, and when fitnesses are constant in time, the mean fitness of individuals in the population will increase monotonically and asymptotically towards a maximum, or peak, in an adaptive landscape. In the more realistic circumstances of a strong epistasis in a complex genomic system, with intermediate levels of linkage and recombination, the consequent genetical constraints, mirrored in the vigorous lability of phenotypic properties with the genomic variations, guarantee that the fitness landscape will be very rugged, with many peaks, ridges and valleys.

Nevertheless, this discussion indicates that combinations of various techniques can be very useful in the context of search space exploration. And in fact, as will be demonstrated later, only proper combination of hill-climbing, simulated annealing and evolutionary search can yield adequate results. In the methodology we are using, all of these search techniques are incorporated in unified platform, supporting each other in order to provide a best possible (in a certain sense) exploration of a search space.

6.5. Genetic Algorithms (GA)

This particular description of the genetic algorithm (GA) initially restricts itself to the strict interpretation of a GA variant investigated by Holland (1975) and his students (e.g. De Jong, 1975). It is still the case that most of the existing theory for genetic algorithms applies solely, or at least primarily, to the model introduced by Holland, or to variations of this model which we will call generically the canonical genetic algorithm. Recent advances in theoretical analysis (Vose, 1993; Radcliffe, 1994) apply mainly to the canonical GA. In the later sections of this chapter, the basic notions relevant to pure canonic GA will be expanded to encompass some of the most recent developments in the GA field.

This algorithm, in a manner similar to that demonstrated in the numerical experiment introducing evolution in so-called naive way (Section 6.2.), receives as input an array of search space objects that form a population. The size of this population, K, similarly to that of scatter-taboo and beam-search, is limited in the sense that the number of objects that is retained to be kept is constant from one iteration or evaluation to another. In principle, the initial population of objects is generated randomly in order to obtain a uniform distribution over the search space.

A key point of this model is that adaptation proceeds not by making incremental changes to a single structure (as frequently advocated in classical AI and machine learning, e.g. Fisher 1987; Winston, 1975), but by maintaining a database of solutions which are continuously recombined and altered in a process that is similar to the process of natural evolution.
Each iteration again consists of an evaluation of a *fitness* of each object in a population during the search process. The fitness here implies again a degree of adaptedness of the evolving object to its environment. The most fit objects are selected in a randomised procedure to participate in reproduction for the generation of a subsequent generation.

A very good characterisation of what genetic algorithms can accomplish is given by Vose (1993, p.64):

"A real GA will with large probability [proceed] asymptotically [to] that local minimum having the largest basin of attraction. As the population grows, the probability of a GA [proceeding] asymptotically anywhere else converges to zero."

Thus, a GA successfully optimises only functions that have global optima associated with the largest basin of attraction. This property is not particularly well suited to many optimisation problems. However, it can be an effective 'satisficing' mechanism, and thus rather useful for a certain class of practical applications.

Hopefully, after a few generations, the population will converge towards a global optimum. As stated by Davis (1987):

"... the metaphor underlying genetic algorithms is that of natural evolution. In evolution, the problem each species faces is one of searching for beneficial adaptations to a complicated and changing environment. The 'knowledge' that each species has gained is embodied in the make-up of the chromosomes of its members."

```
procedure Genetic_Algorithm
begin
  t=0;
  initialise population P(t);
  evaluate fitnesses for each member in population P(t);
  while termination condition not satisfied
  begin
    select parent structures from P(t);
    recombine structures in P(t);
    mutate structures in P(t);
    evaluate structures in P(t);
    t=t+1;
  end
end.
```

*Figure 62 Pseudo-code for genetic algorithm*

Through the application of a crossover operator, which roughly corresponds to sexual reproduction in our natural world, an offspring entity is generated which comes-to-presence as a combination of the phenotypic traits of its ancestors. Evidently, this sort of search strategy is (almost) unaffected by possible local discontinuities in the search space. It is the
correlation between parents and offsprings that can be most easily understood as a facility to view and overcome the ruggedness of a landscape. Therefore the ability of evolutionary algorithms to perform walks on a fitness landscape proceeds through their exchanges of information concerning distinctive solution locations.

A similar discussion has been presented earlier with respect to a comparative analysis of scatter search and beam search. In this respect, GA can indeed be understood as an amalgamation of these two techniques. It maintains a population of structures which are then recombined in a randomised fashion to generate new search points. The number of offspring produced can be normalised with the length of the beam, and the randomised nature of their alterations can be perceived as a degree of 'scatterness' in a search procedure.

6.5.1. Crossover

In canonical GAs the recombination operator plays a central role. Formally speaking, crossover $\theta : \Omega \times \Omega \rightarrow \Omega$ is an operator that acts entirely on the genetic encoding of the individuals. The application of recombination is not entirely deterministically controlled. The crossover rate is controlled by an exogenous parameter, $p_c$, which indicates a probability that an individual will undergo such a modification. As has been demonstrated earlier (Section 6.3), the crossover requires two parent chromosomes:

$$\omega = o_1 \times o_2 \times o_3 \times \ldots \times o_{k-1} \times o_k \times o_{k+1} \times \ldots \times o_l$$

$$\zeta = z_1 \times z_2 \times z_3 \times \ldots \times z_{k-1} \times z_k \times z_{k+1} \times \ldots \times z_l$$

whose content is interchanged around a point which is modelled as a uniform, random variable $\chi \in [0, 1]$, so that the reciprocated sub-strings will appear with the following chromosomes:

$$\bar{\omega} = o_1 \times o_2 \times o_3 \times \ldots \times o_{k-1} \times z_k \times z_{k+1} \times \ldots \times z_l$$

$$\bar{\zeta} = z_1 \times z_2 \times z_3 \times \ldots \times z_{k-1} \times o_k \times o_{k+1} \times \ldots \times o_l$$

This, one-point crossover can naturally be generalised to allow a general $m$-point crossover (De Jong, 1975). Clearly, the uniform crossover (Syswerda, 1989) represents an extreme recombination between two parents: it consists of deciding randomly whether a bit will contribute to one or the other offspring.

Theoretically speaking, it is not clear which version of crossover is the most appropriate for a certain class of problems. Some theoretical work has been done (Caruna, et al. 1989; Eshelman, et al. 1989; Schaffer, et al. 1989), but in practice more empirical studies need to
be performed to assess the feasibility of a particular operator (Babović et al, 1994; Wu, 1994).

6.5.2. Mutation

Mutation in canonical GAs is defined as ‘bit-flipping’ and is an operator of only minor relative importance (Holland, 1975). It works by, once in a while, flipping an occasional bit in an individual chromosome. The probability of occurrence of mutation in canonical GAs is rather low, typically $p_m=10^{-3}$. Formally speaking, a mutation $\mu$ alters the individual $\omega \in \Omega$ such that $\mu: \Omega \rightarrow \Omega$. On a single individual $\omega=(o_1 \times o_2 \times o_3 \times \ldots \times o_l)$, it performs as follows:

\[
o_i' = \begin{cases} 
  o_i \cdot x_i > p_m \\
  1-o_i \cdot x_i \leq p_m
\end{cases}
\]

6.5.3. Selection Mechanism

Selection in canonical GA emphasises a probabilistic survival rule mixed with a fitness dependent chance of having (different) partners for producing more or less offspring (Bäck and Schwefel, 1993). From the perspective of computational cost, the search operators as introduced in the previous sections are rather inexpensive processes which keep the efficiency of genetic algorithm at $o(lN)$, where $l$ denotes the length of binary string and $N$ representing the population size.

The selection mechanism is, clearly, another critical component of the genetic algorithm. The application of the selection mechanism results in entities that will participate in the crossover or be altered by the mutation operator. Thus, the selection mechanism is a procedure that designates entities that will survive. The selection phase is composed of a calculation of expected survival rates for each individual configuration and the conversion of these real-valued estimates into a discrete number of sampled entities that actually survive. The selection algorithm should, therefore, assure a correct sampling of a population (so as not to annihilate relatively good realisation of search operators), while keeping the computational costs as low as possible.

In a now classical contribution, Baker (1987) discussed a number of different selection mechanisms in terms of spread and bias. Baker defines bias as the absolute difference between an individual’s sampling probability and its expected value. The optimal, zero bias is achieved when the sampling probability equals the expected value. Spread is defined as the possible range for the actual number of offspring. Moreover, the minimum spread is defined as the smallest possible spread which permits a zero bias. It is quite obvious that the bias represents a measure of accuracy, whereas the spread selection represents the algorithm’s precision and consistency.
6.5.3.1. Stochastic sampling

This technique is based on a so-called 'biased roulette wheel' method and is similar to the one described previously. The technique is typically implemented in $O(K^2)$ time. Stochastic sampling with replacement is actually the technique used earlier here, in the chapter on naive evolution. The most basic model of selection mechanism uses information of individual performance, thus forming the basis for a fitness proportionate algorithm. Thus, more fit individuals will be more frequently selected. However, as will be clarified in the sequel, such a mechanism must obtain a minimum bias and a low spread with minimal possible computational expense. Formally speaking, the fitness proportional selection $\xi: \Omega \to \Omega$ designates individuals $\omega_i \in \Omega, \forall i \in \{1, 2, \ldots, K\}$ to participate in the application of genetic operators, using the individual relative probabilities calculated as:

$$p_i(\omega_i) = \frac{F(\omega_i)}{\sum_{k=1}^{n} F(\omega_k)}$$

Since the expected values do not change during the selection phase, they are equal to the sampling probability, so that the resulting bias is zero. Spread however, is unlimited: an individual might not be selected at all, but on the other hand it might come to occupy the entire population.

Stochastic sampling with partial replacement, however, alters the individual's expected chances during the selection phase so that each time an individual is selected, its expected value is decreased by one. This reduces spread at the cost of bias.

6.5.3.2. Remainder sampling

This technique involves two distinctive phases. In the so-called integral phase, samples are allotted on the basis of the integer portions of expected values. The second, so-called fractional phase, samples on the basis of the expected values' fractional portions.

Remainder Stochastic Sampling with Replacement is a combination of a deterministic integral phase with a biased roulette wheel applied in a fractional stage. It has zero bias and spread is kept well under control. Remainder Stochastic Sampling without Replacement combines the deterministic integral phase with the fractional phase in which the individual's expected value is set to zero once it has been selected. Thus, multiple selections in the fractional stage are disabled. The result is a procedure providing a medium bias and minimum spread. Deterministic Sampling is a technique in which even the fractional phase is deterministic and individuals with the largest fractions are selected until the population is filled. It provides a minimum sampling error, but unacceptably high bias.

In Remainder Stochastic Independent Sampling, the fractional phase is based on a uniform probability without taking into account the fractional portion magnitudes. This algorithm is obviously easier and cheaper to run. It provides a low bias and a minimum possible spread.
6.5.3.3. *Stochastic Universal Sampling*

This technique is a zero-biased, minimum spread procedure which achieves all $K$ samples in a single traverse through the population, and so in $o(K)$ steps. On a standard biased roulette wheel, there is a single pointer which indicates the 'winner'. *Standard Universal Sampling* is analogous to a spinning of the wheel with $K$ equally-spaced pointers which then, in a single spin, result in $K$ winners. In randomly ordered populations (to avoid positional bias), an individual's selection depends exclusively on its fitness and the result of the roulette wheel's (single) spin. Thus the bias equals zero.

Empirical observations confirm the results of this theoretical view. In his, original paper on this subject, Baker (1987) discusses some other properties, like bias severity, bias direction and bias propagation, and as a conclusion offers the view that stochastic universal sampling and remainder stochastic independent sampling are two techniques that meet most of the requirement set for an effective selection mechanism.

6.5.3.4. *Tournament selection*

Tournament selection operates by randomly taking $k$ members of the population and forming in such a way a *tournament of $k$ individuals*, and then selecting the best individual from the tournament to participate in crossover or mutation. One of the advantages of tournament selection is its possible 'parallelisation', thus avoiding selection bottleneck in canonic genetic algorithms that is serial in its nature.

It can be shown (Goldberg and Deb, 1990) that tournament selection is equivalent to the ranking scheme (Whitley, 1989) in which the highest rank on average contributes 2 members to the genetic pool, the middle rank 1 member and the bottom rank none.

6.5.3.5. *Crowding Model*

The crowding model (De Jong, 1975) is inspired by observations of ecological phenomena occurring in natural populations whereby individuals of the same species are often competing for limited resources. Individuals that are dissimilar typically occupy different niches and do not compete. The consequence is that, for populations of a fixed size, new members of a particular species replace older members of that species. Ideally, for the situation at equilibrium, the overall number of a particular species does not change.

Crowding models allow only a fraction of the evolving population to reproduce and die in each generation. A portion of the population specified through the application of a *generation gap* ($G$), is selected through a fitness-proportionate selection mechanism to be altered by genetic operators (crossover and mutation). Consequently, $G \times K$ individuals from the population must be replaced (that is, must die). The method through which this replacement takes place is of the essence of crowding. Firstly, a random sample of size of $CF < K$ individuals (where $CF$ denotes a *crowding factor*) is taken and, secondly, the most (genotypically, bit-wise) similar chromosome gets replaced. Note that fitness does not play any role in the replacement. The function of fitness is reduced to a fitness-proportionate selection mechanism.
Crowding strives to maintain the diversity of evolving entities and it does partially succeed in this. However, it preserves the existing diversity and does not evolve to provide a useful diversity. Mahfoud (1992), introduced certain modifications to the most rudimentary algorithm that resulted in some considerable improvements to this technique.

6.5.3.6. Preselection

Preselection schemes were originally used by Cavicchio (1970). Basically, Cavicchio observed that it is numerically rather expensive to calculate the 'distance' among evolving entities and that some different method of replacement should be employed. The technique originally proposed simply compared the fitnesses of the resulting offspring to the fitnesses of parents. Should a offspring's fitness be higher than the fitness of the parents, these offspring simply replace their parents. Otherwise, the parents remain in the population.

6.5.4. Fitness calculation

6.5.4.1. Genotype-to-Phenotype Mapping

One of the longest-running and sometimes most heated arguments in the field of genetic algorithms (and to a lesser degree within the wider evolutionary-computing community) concerns the representations of evolving solutions. This is an open debate that deals with issues like the applicability of problem-specific and representation-specific operators, hybridisation with other search techniques, the handling of constraints, the meaning of genes and the efficacy of newer variants of the basic algorithms, such as genetic programming.

A gene is the unit of analysis in determining the features of the phenotype, and hence its fitness, from the genotype; it is coded by a small sub-section of the genotype. The term epistasis refers to the linkage between the genes on the genotype, such that the expression of one gene modifies or even over-rules the expression of another gene.

If there is no epistasis, and consequently the fitness contribution of each element on the genotype is unaffected by the values of any of the others, then the search can be successfully carried out independently on each element. Simple, 'greedy' hill climbing is the most adequate. At the other end of the epistatic scale, when there are many dependencies between the elements, the only useful building blocks are long, and these are relatively easily disrupted by genetic operators. In the case of extreme epistasis, only random search is possible. The appropriate region on the epistatic scale suitable for applying a GA is somewhere between these two extremes, and GA representations must be chose with this in mind (Davidor, 1990).

As indicated previously, the canonic genetic algorithm operates on binary strings of a fixed length, \( l \); thus the individual chromosome is, formally, a \( \omega \in \Omega \), a vector of gene values:

\[
\Omega = A_1 \times A_2 \times A_3 \times ... \times A_l
\]

As has been outlined earlier (section 6.3), in genetic algorithms the members of the sets \( A_i \) are typically represented by the integers zero or one:
\[ A_i = \{0, 1\} \]

Such an individual member of the GA population \( \omega = a_1 \times a_2 \times a_3 \times \ldots \times a_n, \omega \in \Omega \) is used as a basis for the establishing of phenotypical individual using some form of mapping. In some now classic experiments (De Jong, 1975), the bits were not interpreted but passed to the objective function being optimised in the primitive form instead. Interpretation mappings are typically very simple, usually converting a bit string to a real-valued number (De Jong, 1975). The typical mapping (e.g. like the one above) uses a positional encoding (Angeline and Pollack, 1993a) where the semantics of interpretation are tied to the position of the bit in the genotype rather than its content. As a result, the interpretation of a bit string is likely to be a simple combination of the various positions, similarly to the representation of a union of independent features. Positional encoding also provides a means to organise the representation so that bit positions with related semantics are reasonably adjacent and have a lower probability of being separated by a crossover operation. One point must be made here rather strongly, and that is that genetic algorithms manipulate the bit strings without any regard to their eventual interpretation.

Similarly to what has been indicated earlier, one form of this mapping relating phenotypical and genotypical individuals can be:

\[
\rho_i(a_1, a_2, \ldots, a_n) = u_i + \frac{v_i - u_i}{2^{l_i} - 1} (\sum_{j=1}^{l_i} a_j 2^{j-1})
\]

In this case, the entire binary string has been divided into \( m \) segments (usually of equal length, \( l_i \) so that \( l = m \times l_i \). The phenotypical individual \( \xi \in \Xi \) thus becomes:

\[
\xi = \rho^1 \times \rho^2 \times \rho^3 \times \ldots \times \rho^m
\]

David Goldberg is usually identified as the leading supporter of binary representations. He has, however, developed a theory of *virtual alphabets* for what he calls real-coded genetic algorithms (Goldberg, 1990). Goldberg considers the case in which parameters are treated as individual genes and search operators are defined as being of a generalised k-point crossover, and even a uniform crossover type. In these circumstances, Goldberg argues, the algorithm ‘chooses’ useful building blocks and processes the search space rather in close accordance with the schemata theorem.

The other stream within the GA community uses direct real-valued codings of parameters in the search space, thus avoiding mappings from binary strings to the real-valued numbers. These codings open up a space for a variety of recombination parameters, as represented by averaging crossovers (Davis, 1991), random respectful recombinations (Radcliffe, 1991a) and ‘blend crossovers’ (BLX-\( \alpha \); Eshelman and Schaffer, 1992). These are combined with *creep* (Davis, 1991) and *end-point mutations* (Radcliffe, 1991a). Similarly, the Evolution Strategies Community has always used real-valued codings. This work, taken together with
the contribution of Michalewicz (1992), provides compelling evidence that this approach outperform binary codings, whether of traditional or Gray-coded flavour. Davis (1991) in particular provides a potent example of the improvement that can be achieved by moving from binary-coded to real-value-coded genetic algorithms.

6.5.4.2. Gray Coding

The mapping as just presented, although straight-forward to implement, has some negative features that discourage its use. One of these has to do with the occurrence of so-called Hamming cliffs. Consider the case in which 3-bit long binary strings are decoded into a natural numbers on a left-hand side of the Figure 63. It is obvious that the binary expression for the Arabic number 3 (011) and the number with the Arabic numeral 4 (100) differ in values of all three positions in a string. Thus, for a change from 3 to 4 in binary representation, a ‘triple-mutation’ is necessary. It is these occurrences that are referred to as Hamming cliffs; they occur when two entities, although similar at one representational level, differ considerably at another representational level.

<table>
<thead>
<tr>
<th>Genotype</th>
<th>Standard Mapping</th>
<th>Genotype</th>
<th>Gray Coding</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
<td>011</td>
<td>2</td>
</tr>
<tr>
<td>011</td>
<td>3</td>
<td>010</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>110</td>
<td>4</td>
</tr>
<tr>
<td>101</td>
<td>5</td>
<td>111</td>
<td>5</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
<td>101</td>
<td>6</td>
</tr>
<tr>
<td>111</td>
<td>7</td>
<td>100</td>
<td>7</td>
</tr>
</tbody>
</table>

**Figure 63 A comparison between ‘standard mapping’ and Gray code mapping**

In order to overcome this difficulty, a so-called Gray\(^{48}\) coding (Caruana and Schaffer, 1988) has been proposed. A Gray code represents each number in the sequence of integers \(0...2^l\) as a binary string of length \(l\) in such an order that adjacent integers have Gray code representations that differ in only one bit position. Progressing through the integer sequence consequently requires one-bit flipping at each step. This attribute has been referred to as an adjacency property. It should be emphasised that, although studied by many authors (Hollstein, 1971, being one of the pioneering works) the advantages of Gray coding has not been demonstrated unconditionally, whether theoretically or in applications.

\(^{48}\)Gray codes are named after Frank Gray who patented their use for shaft encoders in 1953 (Gray, 1953). Gray codes actually have a much longer history and the inquisitive reader may refer to Scientific American, of August 1972. which contains two articles on the origin of binary codes (Heath, 1972) and some entertaining aspects of Gray codes (Gardner, 1972).
6.5.4.3. *Fitness scaling*

It seems that for the successful performance of genetic algorithms, two requirements should be satisfied: diverse population and a continuous selective pressure. However, these two requirements can just as well work in a counter productive way. Stronger selective pressure eliminates from the population all the entities that do not meet the selection criteria, consequently decreasing the population's diversity, and a low (genetic) diversity within population does not provide the variability that is necessary for a successful search. As Whitley points (1992):

"It can be argued that there are only two primary factors (and perhaps only two factors) in genetic search: population diversity and selective pressure... In some sense this is just another variation on the idea of exploration versus exploitation that has been discussed by Holland and others. Many of the various parameters that are used to 'tune' genetic search are really indirect means of affecting selective pressure and population diversity. As selective pressure is increased, the search focuses on the top individuals in the population, but because of this 'exploitation' genetic diversity is lost. Reducing the selective pressure (or using larger population) increases 'exploration' because more genotypes and thus schemata are involved in the search."

One direction taken in this work is to attempt to alter the fitness function by introducing a so-called *fitness scaling* mechanism. Following rather closely Goldberg (1989), these mechanisms are divided into the following categories:

A. **Linear Scaling.** This method alters the chromosome's fitness as:

\[
\Psi'(\xi) = a \Psi(\xi) + b
\]

where the parameters \(a\) and \(b\) are usually selected in such a way as to map the average fitness into itself and increases the best fitness several times. Thus, the values of coefficients \(a\) and \(b\) are, quite obviously, domain dependent and should be carefully accommodated for every case study.

B. **Sigma Truncation.** This method can be conceived as one of providing an improvement of linear scaling in the direction of promoting domain independence of the tuning parameters:

\[
\Psi'(\xi) = \Psi(\xi) + (\Psi - c \sigma)
\]

where \(c\) is chosen to be a small integer and \(\sigma\) is the population's standard deviation. Possible negative values of scaled fitness \(\Psi'(\xi)\) are set to zero.

C. **Power Law Scaling.** In this method, the fitness is scaled as follows:
where \( k \) is chosen to be close to unity (say \( k=1.005 \)).

Michalewicz (1992) enriches the power scaling by introducing yet another parameter, the *age of population*. In such a way scaling, is *non-uniform* and it changes with the population's statistical properties and its age.

The statistical properties of the population are captured through its *span*, defined as:

\[
S_i = \sqrt{\frac{1}{n-1} \sum_{j=1}^{n} (\Psi(x_j^i) - \Psi(x_j^f))^2}
\]

The value of \( k \) is then calculated as:

\[
k = \left( \frac{S^*}{S_0} \right) \tan^{p_1} \left( \frac{t}{T+1} \right) \frac{\pi}{2}
\]

where \( p_1 \) is a system parameter determining the influence of the span \( s \) on the magnitude of \( k \), while \( p_2 \) and \( \alpha \) are system parameters determining the speed of the changes for \( k \). \( s^* \) is an experimentally-determined, desired value for the span, where \( S_0 \) represents the span's magnitude at the given generation\(^{49} \). Correspondingly, \( t \) denotes the current generation and \( T \) the total number of generations to be taken in a particular run. Michalewicz (1992) used \( \alpha=0.1; p_1=0.05; p_2=0.1 \) and \( s^*=0.1 \).

The consequence of such a non-uniform fitness scaling is such that in the early stages of evolution the emphasis is placed upon exploitation of the search space, and consequently collection of information, whereas in the closing stages of the evolutionary search the exploration process starts to dominate, thus minimising the wasting of computational effort by exploring only the most promising solutions.

---

\(^{49}\) Michalewicz (1992) introduces such a nomenclature (\( s_0 \) instead of \( s^* \)) to indicate that in order to 'save' some of the computational effort, it is not necessary to calculate the value of the span for every generation \( t \), but at some other intervals (say every 10 generations) and apply this value for the calculation of power law scaling.
6.5.4.4. Fitness determination based on competition

Competitive learning is long been a standard topic in machine learning (Samuel, 1959; Tesauro 1992) and has been in use within the evolutionary algorithm community since the first of Bremermann's experiments. The reasoning behind such an approach derives from the fact that 'a competitive learning process encourages an incremental development so that as a new strategies are developed by one learner, its opponent adjusts its abilities and discovers new strategies on its own' (Angeline, 1993). This enhancement of competitiveness increases the abilities of the opponents to learn and adjust to both the opponent and the environment, until the optimal configurations are reached. The interest of this approach derives from the aspiration that the program should itself discover clues directly, from first principles. Thus, the ultimate goal is the maturation of problem solvers solely on the physics of the problem domain.

In most of the competitive learning studies conducted so far, a range of games has been adopted as an experimental set-up. Samuel (1959) experimented with checkers and Tesauro (1990) with backgammon. The essence of competitive learning resides in the fact that the learner plays against itself, practising and trying to detect imperfections in its own strategy. Once these imperfection have been detected, the learner can utilise them in the creation of its own 'personal' range of strategies. Quite obviously, there is no a priori defined concept to be learned. This concept to be learned, as well as the learner's ability to comprehend this concept, are both emergent properties. Through such a repetitive process, opponents incrementally create more and more difficult concepts to be learned, thus supporting discovery of more and more sound strategies.

However, Angeline (1993, p.127) point out that:

"In any learning system, the implicit 'goal' is to locate a strategy such that no feedback received forces the training algorithm to change it. In other words, the concept is stable with respect to the teacher and method of credit assignment. Thus the ratcheting of performance to a good strategy is only one possible limit situation for a competitive learner. Another training sequence could lead to limit performance that is optimal for the player playing itself, but fail drastically when forced to play other players. For instance, in a deterministic game such as Tic-Tac-Toe, a learner could draw itself consistently without being even remotely proficient at that game, in effect 'cooperating' with itself to maximise its score and minimise its own modification. Such a situation, where the learner identifies a sub-optimal strategy that is stable with respect to the task environment and the method of credit assignment is termed a cooperative stable state."

In order to make some use of a competitive model and to promote its evolution towards an effective general method, the bias towards cooperative stable states must be inhibited. Axelrod (1984) performed one of the pioneering works in the study of the cooperation and inferred that the environment in which competitive learning takes place should itself provide good chances for opponents to meet repeatedly. Accordingly, the learner has a large probability of playing against itself and through this self-competition achieves that state in which only the drive towards change is minimised.
As has been reported elsewhere (Tesauro, 1992, Epstein 1994), the success of competitive learning in the creation of artificial backgammon players is due to a certain stochastic feature in the training process. For every move, dice should be rolled again, thus compelling the learner to enter into a variety of strategic situations that would not be possible within a purely deterministic procedure. The power of non-determinism reveals itself again!

The population of evolving solutions in evolutionary algorithms represents a natural source for competition. However, this feature has not been fully utilised. Probably the main reason for this neglect is the belief that such an approach is too vague and unstructured and that it would most probably not lead to successful solutions at the rates exemplified in standard experiments with evolutionary algorithms. However, more recent research performed within the so-called Artificial Life paradigm, indicates the opposite (e.g., Ray, 1992; Lindgren, 1992).

In standard GA applications, the fitness value of an individual does not depend upon the fitness values of other individuals in the population. Such an independence of fitness allows the maintenance of faithful, reliable and invariable measures of fitness throughout the process of simulated evolution.

What is assumed in these process is the objectivity of the fitness measure. For relatively simple problems of function optimisation, this approach is by no means sufficient but still probably represents the most appropriate choice, since, quite obviously, an unprejudiced fitness function would need to incorporate a large amount of knowledge about the problem domain in its formulation. The very corner stones of GAs as weak, domain independent, search methods, would be shaken accordingly. The evaluation of fitness based on competition provides a GA with a competitive edge in highly complex problem domains without sacrificing its domain independence.

To a certain degree competitiveness in EAs is provided through tournament selection (see page 143 for tournament selection within Genetic Algorithms and page 174 for tournament selection in Evolutionary Programming). However, this formulation of competitive fitness is still based on certain objective fitness evaluation.

Competitive fitness models are based on the state of the current population. In competitive fitness model, compares each strategy against each other strategy in the population must be compared. Thus, having the fixed population size \( N \), there must be \( N^2 \) competitions performed in order to establish competitive fitness in this way.

Hillis (1992) used a competitive fitness evaluation in a different fashion. The problem Hillis addressed is the evolution of an algorithm that performs sorting of a number (in his case 16) with as few element exchanges as possible. The number of all possible examples to be sorted was then in his case 16! (=2.09 • 10^13) so that grounding the model induction on a complete set was not practical. At the same time, evolution towards a fixed-size training of some smaller size (say 1000) was observed to result in sub-optimal solutions.

In order to maintain evolutionary pressure intensity, Hillis introduced another population which evolved in interaction with the original population. Each member of this, additional
population then encoded for an example to be sorted. The fitness of these (as Hillis called them) *parasites* is inversely proportional to the fitness of the evolving solutions. Using such an approach, Hillis's algorithm evolved towards a solution which required only 61 position exchange, which is only a single exchange worse than the best known sorting network for 16 numbers.

![Diagram](image)

**Figure 64** Three types of competitive fitness function. (a) full competition described by Axelrod (1989); (b) bipartite competition used by Hills (1992); and (c) tournament fitness proposed by Angeline (1993) (from Angeline, 1993)

If the size of the populations equals to $K$, then only $K/2$ competitions need to be performed in each generation. This is considerably less than the $K!$ required in Axelrod's (1987) model. However, since the performance of the evolving solution is compared against a single 'parasite' there is no basis of comparing the solutions. 'The score received by [an evolving solution] is relative to the difficulty of permutations it attempted and each [evolving solution]
sees a distinct set of permutations' (Angeline, 1993, p. 133). Angeline continues:

"Pitting evolving members of a population against each other to determine fitness creates an interesting tension in evolutionary algorithms. For instance, while the population of the sorting networks in Hillis (1992) is adapting to the specific permutations it is being tested against, the population of permutations [parasites] is searching for the set that forces the sorting network to perform as badly as possible. In order for the sorting networks to reproduce from generation to generation consistently, they must generalise their sorting ability rather than encode for a specific subset of permutations. The need to compensate for the continuing diversity in the permutations directs generalisation in the sorting networks."

The tournament fitness model was originally introduced by Angeline (1993). It uses a binary tournament to determine a relative fitness ranking within an evolving population. The entire population participates in the tournament. Initially, evolving solutions are randomly paired and tested against each other. The better-performing entities progress towards the next round in the tournament. The process is repeatedly iterated until an overall winner is determined. The hierarchical nature of the ranking is strictly enforced, ties being broken by random selection. The total number of competitions for a population size $n$ is (Angeline, 1993):

$$\sum_{i=1}^{\lfloor \log_2(n) \rfloor} \frac{n}{2^i} = n - 1$$

which for comparison is one less than the number required to play each member of the population against a single strategy in a comparable independent fitness function determination.

Angeline (1993) summarised the performance of this algorithm as follows:

"Quantification of performance on the task is unimportant when using tournament fitness; all that is required is a concept of 'better' to compare two strategies. This removes all need for determining exactly how much better one [solution] is than another - the resulting tournament hierarchy is sufficient information for reproduction. Unless the competition, that is the measure of the better, is noisy, an optimal player always wins the tournament."

Once the tournament has been run, any standard selection technique can be used to designate parents for the next generation. To conclude, a fitness evaluation based on competition creates similar tension as that encountered in the model initially described by Hillis (1992), but in more comprehensive manner. Tournament-based fitness reduces the risk of falling into cooperative stable states, since the chances of repetitive playing between two and the same players are very low. For more detailed description of the technique reference must be made to Angeline (1993).
6.5.5. The Schemata theorem

Although the notion of the evolution of a solution is intuitively appealing to many, and the basic belief of evolutionary progress towards better adapted, more competent entities seems in some way 'natural', a more mathematically sound 'proof' of the success of such a process is more appreciated in technical circles. The now-classical work introducing genetic algorithms (Holland, 1975) offers an explanation of the evolutionary process in terms of an implicit sampling of hyperplane partitions of a search space.

Holland argues that a genetic algorithm assigns credit not to a string in the population, but to the schemata of a population, i.e. to its sets of features. He observes that, by assuming that schema are assigned credit, the feedback to a single individual in a population provides information about the various schemata of which this individual is a member (Holland, 1975, p.69):

"Given f features, a single structure A ... is an instance of 2^f distinct schemata, as can be easily affirmed by noting that A is an instance of any schema \( \xi \) defined by substituting #'s for one or more of the f attribute values in [the structure's] representation."

Thus, Holland states that while fitness is not a measure of the individual's performance, it none the less represents a source of information on the 2^f schema of which it is member.

Perhaps the best way to understand Holland's explanation is by means of an example. Consider a simple 3-dimensional space (Figure 65), and assume that the problem is encoded with bit strings with a length of only 3. This can be visualised as a cube with the 000 configuration at its origin. When traversed in a suitable order, the nodes of the cube differ by exactly one bit. The front plane of the cube is defined by strings that start with zero. Thus, one can formally define this plane as 0##, where # denotes 'do not care' symbol. Thus, more formally, these strings, or schemata, are composed of \( A_j \in \{0, 1, \#\} \). Here 1 and 0 are defined bits, whereas # denotes the indifference (don't care) symbol.

We introduce \( o(H) \) to denote the order of schema \( H \), i.e. the number of fixed positions, and \( \delta(H) \) to denote the defining length of a schema \( H \). Let \( m(H) \) denote the number of instances of a schema \( H \) in a population \( P \).

Figure 66 illustrates the so-called 4-dimensional hypercube. In this case, schema 1### corresponds to the outer cube, whereas, 0### corresponds to the inner cube. Both of these schemata are of order \( o(1###)=1 \) and \( \delta(1###)=1 \) and \( o(0###)=1 \) and \( \delta(1###)=1 \). Following this principle, 10## denotes the front plane of the outer cube and 00## the front plane of the inner cube, and so on.
A bit string matches a particular schema if that bit string can be constructed from the schema by replacing the asterisk # with an appropriate value. In this case it is said that the corresponding bit-strings are instances of a particular schema. For example, H=(01#1#) has four instances: 01111; 01110; 01011; 01010. More generally, all bit strings of a length l that match a particular schemata are contained in the hyperplane partition represented by that particular schemata. Every genotypical individual represented as a binary encoding is thus a vertex on a hypercube and is a member of $2^l-1$ hyperplanes. Since the schemata are defined over the alphabet consisting of 3 symbols, {1, 0, #}, it is also easy to establish that the entire search space is composed of $3^l-1$ hyperplanes in total.

A population of individual entities provides information on numerous hyperplanes. Moreover, low ordered hyperplanes should be sampled by numerous points in the population. A key notion, of a genetic algorithm's intrinsic parallelism or implicit parallelism, is derived from the fact that many hyperplanes are sampled when a population of strings is evaluated (Holland, 1975). In fact it can be argued that the number of hyperplanes that are sampled is far more than the number of strings in the population. Many different hyperplanes are sampled.

50 The string composed of * symbols exclusively corresponds to the space itself and is not counted as a partition of the space. For a more specific discussion, see Holland (1975, p.72).

51 Holland initially used the term intrinsic parallelism and only later decided to use the term implicit parallelism. Unfortunately, the term implicit parallelism since has entered into the quite other world of parallel-computing terminology to refer to a parallelism which is extracted from a code written in a functional language that has no explicit parallel constructs. Implicit parallelism, then, does not in the present case refer at all to the potential for running genetic algorithms on parallel hardware and associated system software, although, just to complicate things further again, genetic algorithm are quite generally viewed as highly parallelisable procedures.
evaluated in an implicitly parallel fashion each time a single string is evaluated (Holland, 1975, p.74). The exploitation of the cumulative effects of evaluating a population of points that provides statistical information about any particular subset of hyperplanes is fundamental to GA-based problem-solving.

Implicit parallelism implies that many hyperplane configurations are solved in parallel. The theory indicates that, through the process of reproduction and recombination, the schemata of competing hyperplanes increase or decrease their representation in the population according to the relative fitnesses of the strings that lie in these hyperplanes.

Using the average schema fitness:

\[ F(H) = \frac{1}{m(H)} \sum_{\omega \in H \subset P(t)} \Psi(\omega) \]

within the population \( P \) at generation \( t \), and the average fitness of the entire population \( \Psi(t) \), the schema growth can be calculated as:

\[ m(H^{t+1}) = m(H^{t}) \frac{\Psi(H^{t})}{\Psi(t)} \]

This formula clearly implies the use of a fitness-proportionate selection. If, for the sake of simplicity, one assumes the situation in which the fitness of a certain schema is above average, \( \Psi(H) = c \Psi(t) \), where \( c \) takes a constant value over the previous \( t \) generations, it is relatively easy to obtain:

\[ m(H) = m(H^{0})(1+c)^t \]

which is to say that, the fitness proportional selection allocates an exponentially increasing sampling of the above-average-fit schemata.

\[ m(H^{t+1}) \geq m(H^{t}) \frac{\Psi(H^{t})}{\Psi(t)} (1-p_c \frac{\delta(H^{t})}{l-1})(1-p_m)^{\delta(H^{t})} \]

Taking into account the disruptive factor of the destruction of 'useful' above-average-fit schemata under the action of crossover, \( 1-p_c \frac{\delta(H)}{(l-1)} \) and mutation, \( (1-p_m)^{\delta(H)} \), the resulting Schemata Theorem of canonical genetic algorithms can be stated in the form:
The theorem then states that short, low-order, above-average schemata receive an exponentially increasing number of trials under the fitness-proportionate selection. These, short, low-order, above-average schemata are often referred to as building blocks (Goldberg,
1989). The same author (1989, p.42) sees the power of GA as residing in its ability to detect, locate, generate or isolate these building blocks. The amalgamation of building blocks into more useful (more fit) pieces of chromosome provides the essential characteristic of the canonic genetic algorithms.

The detection of building blocks and their successful combination in longer, more 'competent' or 'information-enriched', sub-strings depends heavily on the coding scheme used. For difficult, multimodal functions where interaction among the genes (epistasis) is strong, a successful encoding should ensure that: (1) related genes are close together on the chromosome, while (2) there is little interaction between genes. These conditions are not always easy to meet. Genes may be related in ways which do not allow all closely related genes to be placed close together in a one-dimensional chromosome (Beasley et al., 1993a). In many cases, the exact nature of interaction may not be known to the programmer or his construct. The condition of non-interaction among genes is related to the problem at hand. If the contribution of each gene to the overall fitness were not to be related to any of the other genes, the easiest way to solve the problem would be through using 'greedy' hill-climbing, which would optimise genes - one by one. However, in most problems, the contributions of the individual genes to overall fitness are strongly interrelated, thus placing constraints on the organisation of chromosome.

The choice of a binary alphabet assures the sampling of the maximum number of hyperplanes through an evolutionary process, thus supporting the argument for the use of binary strings to a large degree. This characterisation, inherent to more classical GAs has, however been sharply questioned in more recent works. Davis (1991), Michalewicz (1992), and indeed many others, have been promoting direct, real-valued encoding and have concluded that (Michalewicz, 1992):

"... The floating point representation is faster, more consistent from run to run, and provides a higher precision (especially with large domains where binary coding would require prohibitively long representation). At the same time its performance can be enhanced by special operators to achieve high (even higher than that of the binary representation) accuracy. In addition, the floating point representation, as intuitively closer to the problem space (for optimisation purposes), is easier for designing other operators incorporating problem-specific knowledge...

... These conclusions are in accordance with the reasons of users of genetic-evolutionary techniques who prefer the floating point representation given in (Goldberg, 1990): (1) comfort with one-gene-one-variable correspondence, (2) avoidance of Hamming cliffs and other artifacts of mutation operating on bit strings treated as unsigned binary integers, (3) fewer generations to population conformity."

(emphasis added)

52 Multimodal fitness functions are of the greatest interest in GA research. Indeed, unimodal functions can usually be solved more easily and more effectively using other, simpler methods.
Even, the schema theorem itself has been questioned by several authors (e.g. Antonisse 1989, Vignaux and Michalewicz 1992, Fogel 1992). However, it is still the most generally accepted characterisation of processes occurring within the genetic algorithm paradigm.

Holland uses the concept of implicit parallelism to argue in favour of alphabets of low cardinality. This has been further elaborated by Reeves (1993). The main advocates of low-cardinality alphabets argue that the degree of intrinsic parallelism can be maximised by maximising the number of schemata to which each solution belongs. This is clearly achieved by maximising the string length, which in turn requires minimising the cardinality of the genetic alphabet used. This simple argument has in its turn been described as seriously misleading by a number of researchers, including Antonisse (1989), Radcliffe (1990, 1994a) and Vose (1991). For Mason (1993), 'there is now no justification for the continuance of the bias towards binary encodings'.

It is both a strength and a weakness of the schema theorem that it applies equally, given a representation space $\Omega$ for a search space $\Xi$, no matter which mapping is chosen to relate genotypes and phenotypes. Assuming that $\Omega$ and $\Xi$ are sets of same cardinality $n$, then there are $n^!$ such mappings and the schemata theorem applies equally to all of them. The only link between the representation and the theorem is the term $q(\xi)$. The theorem states that the expected number of instances of any schema at the next time-step is directly proportional to its observed fitness (in the current population) relative to everything else in that population. Thus the ability of the schema theorem, which governs the behaviour of simple genetic algorithms, to lead the search to interesting areas of the space is limited by the quality of the information it collects about the space through its observation of schema fitness averages in the population (Radcliffe and Surry, 1994). These authors continue even further:

"It can be seen that if schemata tend to collect together solutions with related performance, then the fitness-variance of schemata will be relatively low, and the information that the schemata theorem utilises will have predictive power for previously untested instances of schemata that the algorithm may generate. Conversely, if schemata do not tend to collect together solutions with related performance, while the predictions the theorem makes about schema membership of the next population will continue to be accurate, the performance of the solutions that it generates cannot be assumed to bear any relation to the fitnesses of the parents. This clearly shows that it is essential that domain-specific knowledge be used in constructing a genetic algorithm, through the choice of representation and operators, whether this be implicit or explicit. If no domain-specific knowledge is used in selecting an appropriate representation, the algorithm will have no opportunity to exceed the performance of random search."

In order to overcome these obvious deficiencies of the schemata theorem for complex applications and implementations of genetic algorithms, Radcliffe (1994a) introduced certain generalised schemata formae and set out to develop a formalism to allow operators and representations to be developed while taking due regard of stated assumptions about performance correlations in the search space. The aim was to maximise the predictive power of the schema theorem and thus the ability of the GA to guide the search effectively by allowing for a coding of knowledge about the search space by specifying families of formae.
that might reasonably be assumed to group solutions with related performance. The reader
may refer to (Radcliffe and Surry, 1994) for descriptions of formae for real parameter
optimisation, for solving travelling-salesman-problem-like and set-theory problems.

Angeline’s (1993) arguments are slightly different. Since the only feedback from the problem
domain is in the form of a fitness value for each population member, the components (i.e.
building blocks) of individual chromosomes cannot be explicitly rated. There is no sub-
component within the evolutionary algorithm that can perform credit assignment at the level
of components. Moreover, should two evolving solutions have the same fitness but a different
internal structure, there is nothing in EA to prefer the one configuration over the other. With
EAs, randomised procedures are applied to alter potential solutions, and these altered solutions
continue to be subject to changes. This ‘empirical approach leads to the selection of features
that are both beneficial for solving the problem and conducive to manipulation by the
operators being used’ (Angeline, 1993, p. 45). Angeline (1993) continue:

"Thus it cannot be the organisations of representational components that are selected
by evolutionary weak methods but more abstract phenotypic features, i.e., the qualities
the representation possesses when applied to a task. Interestingly enough, when a
positional encoding is used, such as is often the case in genetic algorithm, there is no
distinction between representational features and phenotypic features. But when the
interpretation of the representation is not tied to a particular position of the
representation, such as in genetic programming or evolutionary programs, the mapping
between structure and behaviour is more complex. Selection for representational
components in evolutionary algorithms using these more complex interpretation
functions cannot occur because the representational features, as well as organisations
of them such as schemas, are inaccessible to the credit assignment mechanism."

This viewpoint is in fact in close correspondence with much earlier biological views
(Dobzhansky, 1956, p.340):

"It cannot be stressed too often that natural selection does not operate with separate
‘traits’. Selection favours genotypes. The reproductive success of a genotype is
determined by the totality of the traits and qualities which it produces in a given
environment."

6.5.6. Generation gaps and steady-state-replacement

Standard genetic algorithms tend to evaluate the whole current population, select from these
and apply genetic operators to produce the next generation. A steady-state algorithm such as
GENITOR (Whitley, 1989) replaces one individual at a time rather than a whole generation
at a time. Since, however the worst individual in the population is replaced, the steady state
GA needs to have global statistics available about the entire population.

The steady-state genetic algorithm (Syswerda, 1989) is a variant of the canonical genetic
algorithm in which a generational replacement is replaced by a so-called ‘steady-state
replacement’. The difference between these algorithms is that, in steady-state replacement, the
worst performing entities from the evolving population are removed and crossover (and
eventually mutation) immediately after the creation of offspring are applied. Thus, contrary
to generational replacement, where the entire population of offspring is created, here the
'mixture' of parents and offspring is continuous. It is believed that there are several
advantages to such a replacement procedure. Firstly, it is naturally elitistic, thus preserving
the best individuals within the population. Secondly, it provides a moderately spontaneous
drift within the population, providing the better chromosomes with better reproductive chances
and allowing inferior members to die out.

6.5.7. Some Other Extensions of Canonical Genetic Algorithms

6.5.7.1. Elitism

Elitism is a simple extension of the canonical genetic algorithm which is directed towards the
preservation of the best-performing entities (the elite), from generation to generation,
unaffected. This assures a monotonically increasing fitness and removes the possibility of
‘destroying’ the most fit configuration by forcing it to participate in crossover and mutation
operations. At the same time, however, elitism supports premature convergence and promotes
the possibility of a fixation on sub-optimal genotypes. Moreover, it reduces the ability to
follow changes in environment, whereby the best-performing elite may well become sub-optimal, and even exceedingly sub-optimal, in time.

6.5.7.2. Fitness sharing

The idea of fitness sharing promotes a perspective on GA in which a number of stable sub-
populations, species, evolve simultaneously.

The idea of fitness sharing stems directly from nature. In natural ecosystems, a large number
of different species co-exist and each one of them fulfils a certain role within the ecosystem.
Each of these roles is referred to as an ecological niche. Organisms belonging to a certain
niche utilise the same (finite) resources and accomplish the same tasks. If the niche becomes
too ‘crowded’, the resources designated for this niche would not be sufficient to support all
the organisms within this niche. Thus, the size of the niche is related primarily to the amount
of basic resources available.

The rationale of speciation within search algorithms resides in the better identification of
niches as local optima. By degrading the fitness at each of these niches, the formation of
stable sub-populations is encouraged. Sharing is carried out on the basis that the fitness ‘pay-
out’ within each niche is finite and must be shared by all individuals in that niche. The total
pay-out for a niche is equal to the magnitude of the maximum, so larger maxima can support
proportionally larger populations (Beasley et al., 1993c).

The fundamental question with respect to niche formation is to determine whether two
individual solutions belong to the same niche or not. Thus, some boundaries to the niches
should be imposed. Goldberg and Richardson (1987) introduce so-called niche radius to
delineate a niche. Closeness between the two individuals is measured by the distance in a
single-dimension parameter space (Deb and Goldberg, 1989). One of the main problems with
this technique has been outlined in (Beasley et al., 1993c) and (Smith et al. 1992) and is the
demanding computational aspects. Quite obviously, in addition to the ‘panmictic calculations’, a new round of calculation must take place in order to ‘adjust’ the fitness while taking the sharing part into account. This additional complexity is $o(k^2)$, where $k$ again represents the population size. This disadvantage becomes debilitating for large populations.

Moreover populations are always large for multimodal problems, even if only in order to capture all extrema. Goldberg (1989b) suggests that for multimodal problems in which the GA attempts to locate $p$ extrema, the population size should be $p$ times larger than the population size for ordinary panmictic, unimodal search. The reason is that niched GA should in the first place allow for a sufficient number of individuals to explore each region of each local optima without much migration among these regions of the search space. In order to support such a stable situation of a co-existence of populations with only a small migration rate at each of many optima, the population must be appropriately sized.

The methodology presented so far does not take into account the relative proportions of fitnesses: the population is supposed to be more-or-less equally distributed among the various extrema. In certain cases, however, it might be of interest to distribute the population in unevenly sized niches, and then according to the magnitudes of the local extrema, thus allowing higher peaks to host larger niches while restricting smaller peaks to smaller niches. Goldberg, Deb and Horn (1992) propose to increase the population size $k$ by a peak ratio factor $\phi$ defined as:

$$
\phi = \frac{\text{average fitness of all peaks}}{p \Psi_{\text{min}}}
$$

where $p$ represents the proportion of peaks of interest and $\Psi_{\text{min}}$ is the fitness of the smallest peak of interest. The required population size then becomes $k = np\phi$, where $n$ denotes the population size necessary to locate one solution (Goldberg, 1989b).

The efforts made towards the reduction of computational cost from $o(k^2)$ to $o(k)$ are reported in Oei, Goldberg and Chang (1991), who point out that it is the population that should be sampled instead of all the possible distances among individuals being calculated. The same article introduces a niche-size threshold parameter, used to limit the number of individuals in each niche. A corresponding peak ratio $\phi$ can take values between 1 (from uninteresting peaks which have a low fitness relative to $\Psi_{\text{min}}$) and $1/p$ (for the cases in which uninterested peaks have values close to $\Psi_{\text{min}}$). Introducing these assumptions in the computational cost computation, the computational cost becomes (Oei et al., 1991):

$$
o(n\phi p^c (\alpha + A\beta p))
$$

where:

$\alpha$ the time for one function evaluation

$g_c$ number of generations to convergence
the time necessary to compute the distance from one individual to another
A the number of individuals that must be sampled in each niche.

A niche formation and evaluation technique that can be used for learning classifier systems has been described by Smith et al. (1992).

6.5.7.3. Population models

The original conception of genetic algorithms due to Holland (1975) contains no notion of the ‘position’ of an individual within the population. All individual chromosomes are kept within the unstructured population and the decision about which of them will participate in crossover is based solely on their fitness (and a little bit of luck in the biased roulette wheel selection). Recently, however, newer methodologies have appeared which abandon these, unstructured or panmictic, populations in favour of more structured populations. The population structure can be superimposed in many different ways.

A coarse-grained model (known as well as an island model) is a paradigm in which a number of panmictic populations is allowed to evolve in parallel without much interaction. The interaction between these various populations is allowed through migration. In certain cases the migration occurs stochastically and asynchronously (e.g. Norman, 1988), whereas in others it occurs in rotation (e.g. Whitley, 1989). In certain other cases the islands themselves have a structure, such as a ring, and migration only occurs between neighbouring islands (Coohn et al., 1987). This model is sometimes referred to as a stepping stone model. The largely independent course of evolution on each island encourages speciation while ultimately allowing genetic information to migrate anywhere in the structured population. This helps to reduce premature convergence and encourages covering if the algorithm is run with suitably low migration rates.

A fine-grained model is also referred to as a diffusion model (Mühlenbein et al., 1991) or cellular (Gordon and Whitley, 1993). In this class of models, it is most usual that every individual has a unique coordinate in a certain space - typically a grid of some dimension and a rectangular or circular boundary. Each individual only mates with its neighbours. The neighbourhoods can overlap, and the size of a neighbourhood can be linked with the fitness value of the individual associated with it. Replacement is also local. This model is well suited for implementation using so-called Single-Instruction Multiple-Data (SIMD) parallel computers (also called array processors, or, loosely speaking, ‘data-parallel’ machines).

The characteristic behaviour of fine-grained models is that in-breeding within demes tends to cause speciation as clusters of related solutions develop, leading to a natural nicheing behaviour (Davidor, 1991). Over time, the strong behaviour of a certain population spreads slowly across the population due to the overlapping nature of the neighbourhoods, and it is this behaviour that gives rise to the terminology of a diffusion model. As in real diffusive models, there is a tendency towards homogenous populations, but at much slower rates than in panmictic populations. As with coarse-grained models, the diffusion helps to avoid premature convergence to local optima. Moreover, if the search is halted at an appropriate stage, the niche formation and evolving process allows a larger degree of coverage of different optima to be obtained than is typically possible with panmictic populations. A
number of papers has been published recently describing slight variations of representations of diffusion and we shall mention here just a few of the most prominent of these, namely those of Manderick and Spiessens (1989), Mühlenbein (1989), Spiessens and Manderick (1991), Davidor (1991b), Baluja (1993) and Davidor et al. (1993).

Figure 68 A fine-grained panmictic model of population

6.5.7.4. Variable length chromosomes and syntax-preserving crossover

In canonical genetic algorithms, the syntax of genotypes is guaranteed under the application of the crossover operator. This is due to the fixed length of the genotypes (l=const). However, variable length genotypes have desirable properties in many cases.

Traditional genetic algorithms use genotypes of a certain, predefined fixed length of chromosome. Thus, for a given search problem, an organisation of a genotype is designed beforehand with strict definitions of gene loci. As has been mentioned earlier, such a semantics is strictly positional, and a simple recombination operator (Section 6.5.1., page 141) is in principle sufficient to provide a successful mixing of the search configurations.

However, the fixed length chromosomes are not the best suited for solving problems of high complexity. Moreover, for the purposes of evolving phenotypes of (in principle) unbounded complexity given a fixed mapping between genotypes and phenotypes, variable length chromosomes must be considered.

Variable length chromosomes are used in many flavours of evolutionary search: the messyGAs, classifier systems and genetic programming that are described later are all based on variable length chromosomes.
It is perhaps useful here to refer back again to the book of nature. We there find that, in addition to frequent changes introduced by mutation and crossover, chromosomes are occasionally also modified by other naturally occurring genetic operations, such as gene duplication and gene deletion. Genetic duplications are rare and unpredictable events in the evolution of genomic sequences. In gene duplication, an entire segment of the DNA sequence is duplicated in the living cell. Initially, there are no immediate changes. Cells continues to produce same proteins; only the underlying DNA code is redundant. Thus, gene duplication is semantics-preserving. However, this newly introduced, redundant portion of code is subject to other genetic operators like mutation and crossover which may change this repetitious gene into something different. Some evolutionary geneticists (Ohno, 1970) claim that gene duplication is the driving force of natural evolution. The reasoning behind such a claim is rather transparent. In a non-redundant code situation, natural selection has to maintain the gene in order to preserve the production of protein. In the redundant situation, this is not necessary and an organism feels no disadvantages should the redundant part of code be altered, since the original piece of the code still maintains its function. It is however precisely when the changes in the redundant portion accumulate and possibly introduce advantages for phenotypes that interesting changes occur. Ohno (1970, see also Koza 1994) emphasises that:

"... the true character of natural selection is not so much an advocator or mediator of heritable changes, but rather it is an extremely efficient policeman which conserves the vital base sequence of each gene contained in the genome. As long as one vital function is assigned to a single gene locus within the genome, natural selection effectively forbids the perpetuation of mutation affecting active sites of a molecule"

In the later sections, it will be seen that simple mutation and crossover suffice to explain major evolutionary changes that can result in a description of a possible increase in the complexity of phenotype's performance:

"... while allelic changes at already existing gene loci suffice for racial differentiation within species as well as for adaptive radiation from an immediate ancestor, they cannot account for large changes in evolution, because large changes are made possible by the acquisition of new gene loci with previous non-existent functions.

... Only by the accumulation of forbidden mutations at the active sites can the gene locus change its basic character and become a new gene locus. An escape from the
ruthless pressure of natural selection is provided by the mechanism of gene duplication. By duplication, a redundant copy of a locus is created. Natural selection often ignores such a redundant copy, and while being ignored, it accumulates formerly forbidden mutations and is reborn as a new gene locus with a hitherto non-existent function.

Thus, gene duplication emerges as the major force of evolution."

The effects of genotypes of variable length are explored particularly within studies of evolution of phenotypes that emulate behaviours. Lindgren (1991) reports a particularly fascinating study of the iterated prisoners dilemma, whereas Koza (1994) introduces an entirely new family of genetic operators based on gene duplication principles (branch duplication, argument duplication, branch creation, argument creation, branch deletion and argument deletion). Harvey (1992a, 1992b) describes the principles underlying this approach from a so-called evolutionary robotics perspective, and addresses the issues of syntax preserving crossover in genetic algorithms within this context.

6.5.8. Messy Genetic Algorithms

Messy genetic algorithms (mGA) constitute an approach to evolutionary computation that has been introduced with the intention of overcoming the standard genetic algorithm’s deficiencies as encountered in difficult, multimodal problems. As has been indicated earlier, the semantics of the traditional GA is positional. Recombination thus basically shuffles the positions in the bit strings rather than the content of the genotype. This results in a rather difficult obstacle for the standard GA, which is often referred to as the linkage problem. The core of the linkage problem lies in the inability of the standard GA to form useful building blocks. The necessary linkage associated with a problem is often not known, and some other theoretical approaches (Goldberg and Bridges, 1990) have shown that a reordering of the genome is too slow to be used in a search for a tighter linkage. As has been indicated earlier as well, GAs are particularly useful in difficult, multimodal problems so that there has been a considerable incentive to find a solution to this problem. Accordingly, in the so-called class of deceptive problems, which are famous for introducing very difficult linkage problems where standard GAs are of little use, mGAs have been developed with the purpose of overcoming the deficiencies.

mGAs work by searching for tight building blocks and then combining them together to form the optima in a way that respects a version of the schema theorem. The intricacies of messy genetic algorithms are described in full detail elsewhere (Deb, 1991; Goldberg, Deb and Korb, 1990, 1991; Goldberg, Horb and Deb, 1989). Here, only the most fundamental differences between ‘standard’ and messy variants of GA are summarised.

Messy genetic algorithms operate with genotypes of variable length. Moreover this, variable-length, string representation may be under- or over-specified with respect to the problem being solved. In an over-specified case a specific locus may be occupied by more than one gene value (that is, it may posses two or more allele values). The resolution of the over-specification is settled through the use of simple rules, like that of a first-in-first-out or left-to-right-scan, that inspects the genotype in a definitive fashion and selects the first gene that it
detects. Under-specification is resolved through the use of so-called competitive templates. The competitive template can be best understood as a default genotype to which allele values are applied in the cases where evolving genotypes are under-specified.

The search operators used in mGA are again those of crossover and mutation. Since genotypes are of variable length, the ordinary crossover operator cannot be applied directly. Instead, cut and splice operators are employed. Cut splits the string of length 1 with a certain probability:

\[ p_c = (l-1)p_s \]

that obviously grows with the increase in string length \( l \), while the splice combines the sub-strings into whole strings again with a certain fixed probability \( p_s \). An analysis of cut and splice (Deb, 1991) indicates that, for small values of \( p_c \) and large values of \( p_s \), short strings are recombined almost fully and large strings are processed in the same way as they are with a single-point crossover operator.

The selection mechanism used is that of binary tournament selection and consequently there are only two phenotypes in each of the tournaments.

mGAs divide the genetic search into two phases: a primordial and a juxtapositional phase. In the primordial phase, the population is first initialised to contain all possible building blocks of a certain, characteristic length. This length is chosen such as to encompass possibly deceptive building blocks. After this, the algorithm is run for a number of generations with the application of the reproduction operator only. In this stage, then, the population is ‘enriched’ with good building blocks. During this process, the population size is reduced by 50\% at specified intervals. Thus, the primordial phase prepares the population for the second stage, the juxtapositional phase, in which the population size is kept constant and all genetic operators (cut, splice and reproduction) are applied. In all empirical tests to date, mGAs have always found globally optimal strings (Deb and Goldberg, 1991).

In order to select salient building blocks, a level-wise genetic algorithm was proposed by (Goldberg, Deb and Korb, 1990; Deb, 1991). A level-wise mGA consists of a number of eras. Initially, the population is initialised with all individual genes with a unit length. Since these unit-length building blocks cannot be deceptive, a random competitive template is used to reinstate the missing positions in the genotype. The best solution found at the end of this, first era is then used as the template for the second era. The second era proceeds in a similar fashion, firstly initialising all possible strings with a length of two, and then using the best template from the first era to fill-in the missing bits, after which the procedure continues.

Since messy GA have rather unstructured genotypes and are geared towards the simultaneous solutions of sub-problems through explicit evolution (or, rather, breeding) of building blocks, this process sometimes diverges to provide a large number of dissimilar configurations of similar fitness. In order to ‘unify’ these efforts, two empirical mechanisms were developed: genic thresholding and length-based tie-braking (Deb and Goldberg, 1991).
Genic thresholding requires two chromosomes; both have a certain threshold value \( \pi \) of genes in common in order to participate in the same tournament during the selection process. The threshold value is established using a hyper-geometrical distribution and is calculated as:

\[
\pi = \frac{l_1 l_2}{n}
\]

for 'messy' strings of length \( l_1 \) and \( l_2 \).

Tie braking is used to prevent parasitic bits participating in strings. Parasitic bits are bits that conform to a competitive template but serve no purpose within the individual string. There is a danger of these parasitic bits becoming incorporated in low-level building blocks and preventing the expression of some other building block. An additional \( k-1 \) null bits (explicit place holders) may be added to functional genes, or all building blocks at current level and less may be generated, but the key point is to prefer strings with the least length when fitness ties occur between building blocks. In this way, strings will be preferred that have the least amount of stray genetic material, thereby reducing the threat of the expression becoming blocked from parasitic bits (Deb and Goldberg, 1991).

Goldberg et al. (1989) state that:

"A difficult test function has been designed, and in two sets of experiments the mGA is able to find its global optimum... In all runs on both sets of experiments, the mGA converges to the test function global optimum. By contrast, a simple GA using a random ordering of the string is able to get only 25% of the subfunctions correct."

and year later it was reported in Goldberg (1990):

"Because mGA can converge in these worst-case problems, it is believed that they will find global optima in all other problems with bounded deception. Moreover, mGAs are structured to converge in computational time that grows only as a polynomial function of the number of decision variables on a serial machine and as a logarithmic function of the number of decision variables on a parallel machine. Finally, mGAs are a practical tool that can be used to climb a function's ladder of deception, providing useful and relatively inexpensive intermediate results along the way."

### 6.6. Evolution Strategies

The first developments of Evolution Strategies (ES) occurred in Berlin, Germany, in 1964 (Schwefel, 1965; Hartmann, 1974; Höfler, 1976). The problems tackled at that time were mainly of an optimisational nature and had to do with hydrodynamic shape optimisation, like flushing-nozzle shape optimisation, and control problems like the optimisation of a PID controller within a highly non-linear system (Bäck et al., 1991). Early evolution strategies used a real-valued variable representation, whereas the only search operator employed was that of mutation. Such an algorithm is referred to as a two-membered ES and is often denoted as (1+1)-ES. It is based on a population consisting of one parent individual chromosome
which is formally a $\omega \in \Omega$, a vector of gene values:

$$\Omega = A_1 \times A_2 \times A_3 \times ... \times A_l$$

As has been outlined earlier (Section 6.3., page 63), the members of the sets $A_i$ are typically real-valued utilities:

$$A_i = \{R\}$$

Such an individual member of the ES population $\omega = a_1 \times a_2 \times a_3 \times ... \times a_n$, $\omega \in \Omega$ operates as a parent and generates a unique offspring individual. The better of the two is selected to generate an offspring for the following generation, and so on. The mutation is realised by introducing normally distributed random numbers:

$$a'_i = a_i + N_0(\sigma_i)$$

where $N_0$ denotes a vector of independent Gaussian probability distributions with zero mean and standard deviations $\sigma_i$. The values of the standard deviations $\sigma_i$ are controlled by external heuristics, using the so-called $1/5$ success rule (Rechenberg, 1973). This rule is confirmed theoretically and postulates that control of the standard deviation should be based on the measured success frequency. Thus, the ratio of successful mutations to all mutations should be $1/5$. If it is greater than $1/5$, this increase the standard deviation; while if it is smaller, it decreases the standard deviation (Bäck and Schwefel, 1993; more generally, see the statistical-analytic derivation of Khinchin, 1957, and hence Khinchin, 1949).

However, this $1/5$ success rule, did not always produce useful results within the two-membered ES context. There were certain topological situations that were extremely difficult to examine successfully using such a principle. Therefore, the multi-membered evolution strategies evolved further. Initially, it was Rechenberg (1973) who introduced a population principle, that corresponds to establishing multi-membered ES. The presence of more ancestors introduced the possibility of sexual reproduction among the parents. In this first multi-membered ES, all of the progenitors created just one offspring. Following a convention established by Schwefel, such an algorithmic structure can be denoted as a $(\mu+1)$-ES, where $\mu$ corresponds to the cardinality of the population set. The discrete recombination was developed in order to simulate sexual reproduction. In this case, two selected parent chromosomes $\omega_a$ and $\omega_b$ recombine their traits according to an algorithm which assigns the traits from one of the parents to one of the offspring according to the following principle (Bäck et al., 1991):
Thus an offspring 'receives' discrete value of a 'gene' from one of the parents according to the outcome of the uniform random variable $\chi$ on the interval $[0,1]$ and this is sampled anew for each component of the vectors $a_i'$ and $\sigma_i'$. The mutation operator is realised in a similar way to that described earlier.

The drive towards a multi-membered ES that produced more than one offspring ($\lambda > 1$) was initiated to accommodate the benefits of parallel search and to enable a self-adaptation of strategic parameters like the standard deviations of the mutations (Schwefel, 1981). These variants of the ES became known as $(\mu + \lambda)$-ES and $(\mu, \lambda)$-ES. Selection in the $(\mu + \lambda)$-ES and $(\mu, \lambda)$-ES is completely deterministic and its task is to select the $\mu$ best ($1 \leq \mu \leq \lambda$) individuals out of $\lambda$ offspring.

The fundamental idea behind ES is to allow control parameters to evolve together with the evolving individual rather than being controlled by some 'exogenous heuristics in a deterministic manner' (Bäck et al., 1991). It must be underlined that, although the standard deviation values are 'individualised' in so far as every individual entity's trait has its own value, still their magnitudes are controlled externally. So far, these were never considered to be a part of the genotype. However, with the evolution of multi-membered ES this situation changed dramatically, allowing for an evolution of $\sigma_i'$ as well. The nomenclature suggests that in a $(\mu + \lambda)$-ES, an initial population of $\mu$ generates $\lambda$ offspring. Thus, increasing the size of the population to $\mu + \lambda$ from which the $\mu$ most fit individuals are selected deterministically to form the subsequent population of ancestors, should allow the parents to survive until better offspring are created. Such a variant of an algorithm which preserves the best offspring from generation to generation and thus ensures a monotonically increasing fitness is referred to as the elitistic form of such an algorithm.

However, in a certain class of problems where the fitness landscape undulates with time under

---

53 Selection operates on $\mu$ parents in $(\mu, \lambda)$-ES, whereas in $(\mu + \lambda)$-ES, the parents are selected from the totality of $\mu + \lambda$ potential ancestors.
changes in environmental pressures, such an elitistic approach does not behave in the best possible way, particularly with respect to strategy parameters. Therefore, the \((\mu, \lambda)\)-ES has been recommended with a heuristic ratio \(\mu/\lambda = 1/7\) as being rather closer to optimal (Schwefel, 1987).

There are several possible kinds of recombination operators that can be used in evolution strategies and they are summarised below, from top to bottom (no recombination; discrete recombination; intermediate recombination; global, discrete recombination and global, intermediate recombination: (see Bäck and Schwefel, 1993):

\[
\begin{aligned}
\mathbf{a}_i' &= a_{a,i} + \chi(a_{b,i} - a_{a,i}) \\
&= a_{a,i} \text{ or } a_{b,i} \\
&= a_{a,i} + \chi a_{a,i} \\
&= a_{a,i} + \chi(a_{b,i} + a_{a,i})
\end{aligned}
\]

The mutation operator transforms both the variable itself and the control parameters as follows:

\[
\sigma' = \sigma \exp\mathbf{N}_0(\Delta\sigma)
\]

\[
a' = a + \mathbf{N}_0(\sigma')
\]

where \(\Delta\sigma\) denotes the step-size meta-control variable.

Mutation, in combination with selection, performs a kind of hill-climbing search. However, with assigned control parameters \(\sigma\), for every variable \(a_i\), the direction of search is always in the direction of the coordinate system, rather than in the direction of the slope of the problem surface. Schwefel (1981) introduced a version of correlated mutations, in which an additional control variable, a strategy vector \(\theta\), was introduced. The correlated mutation now operates as:

\[
p(C_0(\sigma,\theta) = \frac{\det\mathbf{A}}{(2\pi)^n} \exp(-\frac{1}{2} \mathbf{N}_0(\sigma)\mathbf{N}_0(\sigma)^T))
\]

where \(\mathbf{N}_0\) denotes a vector of independent Gaussian probability distributions with zero mean and standard deviations of \(\Delta\sigma\) and \(\Delta\theta\). \(C_0\) refers to a normally distributed random vector with
expectation zero and probability density \( p \), as exemplified in the equations below (Bäck et al., 1991):

\[
\sigma' = \sigma \exp(N_\sigma(\Delta \sigma))
\]

\[
\theta' = \theta + N_\theta(\Delta \theta)
\]

\[
a' = a + C_i(\sigma', \theta')
\]

The diagonal elements of the covariance matrix \( A^{-1} \) are the independent variances \( \sigma_i^2 \) (squares of the mutation step sizes) of the object variables \( a_i \), while the off-diagonal elements represent the covariances \( c_{ij} \) of the mutations. In a similar way to the \( \sigma \) control parameter, \( \theta \) is as well incorporated into the genetic representation. The comparison in performance between simple and correlated mutations is depicted in Figure 70.

The pseudo-code for an evolutionary strategy of this kind is presented in Figure 71.

The pseudo-code is almost identical to the pseudo-codes for GA and EA. The real differences between these methods occur, as has been already demonstrated, at the lower levels of the structure.

---

**Procedure** Evolutionary_Strategy  
**begin**

\( t=0; \)

initialise Population\((t)\);

evaluate Population\((t)\);

while termination condition not satisfied  

begin

\( t=t+1; \)

select Population\((t)\);

mutate structures in Population\((t)\);

crossover structures in Population\((t)\);

evaluate structures in Population\((t)\);

end

**end.**

**Figure 71** A pseudo-code for an evolutionary strategy (ES)
6.7. Evolutionary Programming

L.J. Fogel introduced the notion of using simulated evolution on a population of contending algorithms in order to develop artificial intelligence, and he explored this possibility in a series of studies (Fogel, 1962a, 1962b, 1962c, Fogel 1963, Fogel and Walsh 1964, Fogel et al., 1966). According to this view, intelligent behaviour requires the composite ability to: (1) predict one’s environment, coupled with (2) a translation of these predictions into a suitable response subject to the given goal. In order to provide maximum generality, evolutionary programming describes the environment as a sequence of symbols taken from a finite alphabet. The evolutionary problem was defined as one of evolving an algorithm that would operate on the sequence of symbols thus far observed in such a manner as to produce an output symbol that is likely to maximise the performance of the algorithm relative to the next symbol to appear in the environment and a well-defined pay-off function. Finite state automata provide a useful representation of the required behaviour.

A finite state automaton (FSA) - see Hopcroft and Ullman (1979) - is a transducer that can be simulated by a finite alphabet of input symbols, that can respond in a finite alphabet of output symbols and possess some finite number of different internal states.

In a graph-theoretic sense, an automaton is a labelled graph with an initial vertex and a set of final vertices. Thus:

\[ \Sigma \leftarrow A \Rightarrow \Psi \mid \leftrightarrow \Psi' \leftarrow I \]

where \( \Sigma \) defines an alphabet\(^{54} \), \( \Psi \) a set of final states and \( I \) a set of initial states.

An automaton is deterministic when the mapping \( A \rightarrow \Psi \times \Sigma \) is one to one. Thus it is enough to give:

\[ \Psi \times \Sigma \rightarrow \Psi \mid \leftrightarrow \Psi' \leftarrow I \]

The corresponding input-output symbol pairs and next state transitions for each input symbol, when taken over every state, specify the behaviour of any finite state automaton, for any starting state. Although finite, FSAs can generate quite a rich behavioural repertoire. If we denote with \( I \) the number of input signals that an automaton can ‘understand’, with \( S \) the number of internal states and with \( O \) the number of possible output actions, then the ‘range’ of an FSA’s repertoire would amount to:

\[ (S + O)2^{I-S} + S \]

L.J. Fogel proposed the use of evolutionary programming to evolve finite state automata using the following procedure. A population of parent-state automata are exposed to the environment, that is, the sequence of symbols which have been observed up to the current

\(^{54} \text{An alphabet is any set } \Sigma \)
time. For each parent machine, and as each symbol is offered to the machine, the output symbol is compared to the next input symbol. The quality of this prediction is then measured with respect to the given pay-off function. After the last prediction is made, a function of the pay-off for each symbol indicates the fitness of the automaton.

![Transition Diagram](image)

**Figure 72** A transition diagram associated with **Figure 73** Representation of a finite state automaton as a transition table

Offspring machines are created by mutating each machine randomly (so that the recombination operator is missing). Each parent produces a single offspring. There are five positive modes of random mutation that naturally result from the description of the automaton: change an output symbol, change a state transition, add a state, delete a state, or change the initial state. Mutations are chosen with respect to a probability distribution, which is typically uniform. The number of mutations per offspring is also chosen with respect to a probability distribution (e.g. Poisson) or it may be fixed a priori. These offsprings are then evaluated over the existing environment in the same manner as their parents. Other mutations, such as majority logic mating, were as well proposed, but apparently never really implemented in a running algorithm.

More formally, for the mutation the standard deviation is calculated for every individual as the square root of a linear transformation of its own fitness value, *i.e.* for mutation $\mu: \Omega \rightarrow \Omega$ which alters $\omega_i \in \Omega$, $\forall i \in \{1, 2, 3, ..., N\}$ as follows:

$$\omega_i' = \omega_i + \sigma \xi$$

$$\sigma_i = \sqrt{\beta_i \Psi(\omega) \gamma_i}$$
The random variable $z$ is modelled using the probability density function:

$$ p(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} $$

In this way, for each component $o_i$ of the vector $\omega=(o_1 \times o_2 \times o_3 \times \ldots \times o_l)$, a different scaling of the mutation rate can be achieved through tuning the parameters $\beta_i$ and $\gamma_i$. In practice, however, these are often set to one and zero, respectively.

Fitness values $\Psi(\xi_k)$ are obtained by exposing automata to the environment. Those automata which provide the greatest pay-off are retained to become parents for the next generation. Typically, half of the total number of the automata are saved so that the population remains the same size.

**procedure Evolutionary_Programming**

```plaintext
begin
  t=0;
  initialise Population(t);
  evaluate Population(t);
  while termination condition not satisfied
  begin
    mutate structures in Population(t)
    evaluate structures in Population(t);
    select structures in Population(t);
  end
end.
```

*Figure 74 A pseudo-code for evolutionary programming*

The selection mechanism $\zeta: \Omega \rightarrow \Omega$ designates individuals $\omega_i \in \Omega$, $\forall \ i \in \{1, 2, \ldots, k\}$ through an application of a kind of **tournament selection**. From a population set that is composed of parents and their offspring (thus giving a size of $2k$), $2k$ tournaments of size $q$ are performed, that is, one for each individual $\xi_k$. The members participating in the individual tournaments are randomly selected. Every tournament results in an (integer) score indicating the number of structures in tournament $\xi_k$ that increase their score. Thus: the score $\eta(\xi_k) \in (0, q)$ $\forall \ k \in (0, 2k)$. This information is then used to **rank** the individuals, and only the $k$ best performing are retained as parents for the following generation. It should be observed that the selection procedure is **elitist**, since it guarantees the 'survival' of the best structure (the best structure will always win the tournament). This process is iterated until it is necessary to make an actual prediction of the next symbol (as yet unexperienced) in the environment. The best machine generates this prediction, the new symbol is added to the experienced environment, and the process is repeated.

There is an inherent versatility in such an evolutionary form of programming. The pay-off
function can be arbitrarily complex and can possess temporal components. There is no requirement for the classical squared error criterion or any other smooth function. Furthermore it is not required that the predictions be made with a one-step look ahead. Forecasting can be accomplished using an arbitrary interval and extended over an arbitrary length of time into the future. Multivariate environments can be handled and the environmental processes need not be stationary as the simulated evolution changes will adapt to changes in the transitional statistics.

In general, the standard evolutionary programming algorithm imposes certain difficulties with respect to the tuning parameters $\beta_i$ and $\gamma_i$. To overcome the difficulties, D.B. Fogel (1992) developed by way of an extension, a meta evolutionary programming that introduces self-adaptation on $\mathbf{N}$ individual variances $c_i=\sigma_i^2$, so that the mutation in the meta-EP becomes:

$$\omega_i' = \omega_i + z_i \sqrt{c_i} = \omega_i + z_i \sigma_i$$

$$c_i' = c_i + z_i \sqrt{\zeta c_i} = c_i + z_i \sqrt{\zeta}$$

where $\zeta$ denotes an exogenous parameter. Fogel (1992) proposed to set the lower bound on the mutated $c_i$ equal $\epsilon_o$, so that no negative variances can occur.

In addition to his meta-EP, Fogel (1992) proposes what he calls a Rmeta evolutionary programming algorithm which incorporates correlation coefficients into the evolutionary process. Thus, the complete vector of $\mathbf{N} (\mathbf{N} - 1)/2$ correlations, $\forall i \in \{1, 2, 3, ..., \mathbf{N} - 1\}, j \in \{1, 2, 3, ..., \mathbf{N}\}$, is calculated as:

$$\tau_{ij} = \frac{c_{ij}}{\sqrt{\sigma_i \sigma_j}}$$

This forms a covariance matrix that self-adapts to the evolutionary process. Fogel (1992) has tested the performance of this algorithm for a population size $\mathbf{N}=2$. The extension of this notion to embrace larger populations is not obvious due to the requirements of symmetry and the need to maintain a positive definiteness of the correlation matrix.

### 6.8. Learning Classifier Systems

A classifier system is an attempt to apply evolutionary algorithms within a machine learning paradigm. The original idea was initially presented in Holland’s seminal paper (1986). One of the pioneers in the field, Stewart Wilson, defines the paradigm concisely as follows (1994):

“A classifier system is a learning system in which a set of condition-action rules, called classifiers, compete to control the system and gain credit based on the system’s
receipt of reinforcement from the environment. A classifier’s cumulative credit, termed strength, determines its influence in the control competition and in an evolutionary process using a genetic algorithm in which new, plausibly better, classifiers are generated from strong existing ones, and weak classifiers are discarded."

Classifier systems are thus particular kinds of production systems with general mechanisms for processing rules in parallel that provide for the adaptive generation of new rules and for testing the usefulness of existing rules. Typically, rules in classifier systems are encoded as binary strings.

![Diagram](image)

**Figure 75** Learning classifier system is embedded in the task environment in order to acquire credit in an empirical fashion

As De Jong (1988) explains:

"To anyone who has read Holland (1975), a natural way to proceed is to represent an entire rule set as a string (an individual), maintain a population of candidate rule sets, and use selection and genetic operators to produce new generations of rule sets. Historically, this was the approach taken by De Jong and his students while at the University of Pittsburgh (e.g., see Smith, 1980, 1983), which gave rise to the phrase ‘the Pitt approach’.

However, during the same time period, Holland developed a model of cognition (classifier systems) in which the members of the population are individual rules and a rule set is represented by the entire population (e.g., see Holland and Reitmann, 1983; Booker, 1982). This quickly became known as ‘the Michigan approach’ and initiated a friendly but provocative series of discussions concerning the strengths and
weaknesses of the two approaches."

6.8.1. Non-learning Classifier Systems

Classifier systems provide an excellent example of the principle of emergence. The underlying action of each individual classifier is extremely simple, yet the global behaviour of classifier system can be highly complex. This situation resembles fine-grained parallel systems in which the activities of each individual classifier can be assigned to and executed by an individual processor. The performance of each individual classifier is locally determined in that its precondition part decides when the classifier will be ‘fired’ and its action part establishes what will be ‘fired’.

Each classifier (basically a production rule with a somewhat limited syntax) consists of a precondition and an action part. An individual classifier may have more than one condition in its precondition part (when it is referred to as a multiple conditions classifier) and only one action part.

The alphabet of classifiers consists of 1, 0, and #. Thus, individual classifiers are perceived as (in principle) fixed-length strings composed over {0, 1, #}. 1 and 0 are referred to as literals, while # is referred to as an indifference (do not care) symbol. The precondition part of an individual classifier may contain the negation symbol ‘~’, which is used to denote the situation in which a condition is fulfilled when no individual condition matches it. A summary of classifier system syntax, following Forrest (1991a) is provided in Figure 76.

```
<classifier system> ::= <classifier>*
<classifier> ::= <condition>+ => <action>
<condition> ::= <alphabet>° | -<alphabet>°
<action> ::= <alphabet>°
<alphabet> ::= 1 | 0 | *
```

*Figure 76 Syntax of a classifier system (adapted from Forrest, 1991a)*

The input to a classifier program is the set of external messages (often referred to as detector messages) that are added to the message list during the program’s execution. The output is a set of messages (referred to as effector messages) read from the message list by an external agent. Just as many traditional programs can run interactively, a classifier program can be thought of as receiving intermittent inputs from the external environment and occasionally emitting output signals, which are interpreted as messages.

As indicated in Figure 77, a non-learning classifier system performs in a match-and-execute cycle. Each individual classifier can thus be regarded as an individual instruction that reads
messages as inputs and produces messages as outputs.

Consider the following, four-bit (l=4) classifier system:

*00* => 1101
*101
***1 => **1*
~1111 => 1111.

This classifier system has three classifiers. The second classifier illustrates multiple conditions, and the third contains a negative conditions.

If an initial message, 0000, is placed on the message list at time T0, the pattern of activity shown on the right, will be observed on the message list. The final two message lists (<empty> and 1111) would continue alternating until the system is turned off. In T1, one message (1101) matches the first (distinguished) condition and both messages match the second condition. Pass-through is performed on the first condition, producing one output message for time T2. If the conditions had been reversed (**1 distinguished), the message list at time T2 would have contained two identical messages (1111).

**Figure 77** Match-and-execute iterative cycle in a non-learning classifier system (adapted from Forrest, 1991a)

```
procedure CFS
begin
  t=0;
  initialise MessageList ML(t);
  initialise classifier Population(t);
  while termination criterion not satisfied
    t=t+1;
    ML(t) :=readDetectors(t);
    ML'(t) :=matchClassifiers ML(t), P(t);
    ML(t) :=sendEffectors ML'(t);
  end
end.
```

**Figure 78** A Pseudo Code for a Non-Learning Classifier System
Evolving Solutions

Effector messages form a so-called message list. This list has a duration of only one time step, since, after matching has been done, the output messages completely overwrite the old message list. The termination criterion of this match-and-execute iterative process can be either a fixed number of iterations or the situation in which the message list remains unchanged (quiescent) for two consecutive steps. Forrest (1991a) provides an illustrative example of this process (Figure 77).

6.8.2. Learning Classifier System

For systems of even moderate complexity, the issue of the design of classifier systems springs immediately to the central position. Designing 'by hand' of classifier systems is hampered by many problems - the knowledge acquisition bottleneck being probably the central one. This top-down process of designing is essentially one of establishing a mapping between high-level structures of the real-world and low-level structures of the classifier system.

The bottom-up approach, in turn, can be based on some kind of learning (such as genetic algorithms and Q-learning) in which low-level structures (individual classifiers) receive reinforcements from the environment in which the classifier system is operating. This process of credit assignment allows the system to emerge and self-organise using only information about the usefulness of its individual instructions.

There are two ways in which learning algorithms control classifier systems: (1) by controlling the write access to the message lists and (2) by controlling which classifiers are in the database of rules.

Consequently, some formal measures of usefulness of individual classifiers must be established. This is done by introducing the notion of the strength of a classifier. Its strength reflects the overall usefulness of a classifier to the performance of the entire system.

Once an external message is received, classifiers matching this message may potentially form a message list. In order to assure efficiency, the size of the message list is usually restricted to a small number. Here, the strength of the individual classifier and its specificity\textsuperscript{55} are used to decide which classifiers will enter the message list. Sometimes, another measure, of support\textsuperscript{56} is also used.

A learning algorithm updates the strengths of classifiers, 'rewarding' the ones that contribute to the formation of good solutions and 'punishing' the ones that fail. The system is embedded in a problem environment and its emergence depends solely on its performance within its environment. For a hydroinformatics orientated application of a learning classifier system and use of dynamic programming in establishing a strength of individual classifiers, a reference may be made to Wilson G. (1995).

\textsuperscript{55} Specificity of a classifier corresponds to the number of literals in its condition part.

\textsuperscript{56} Support roughly corresponds to the number of previously active classifiers that support the proposition that the bidding classifier should be active now.
In addition to this, so-called ‘bucket-brigade’ algorithm, the genetic algorithm can also be applied to alter existing classifiers by using the genetic operators already described producing new classifiers and enriching the knowledge-base. The application of the GA is only periodic, once the strengths (which are here equalised to the fitness of the individual classifiers) are accumulated. Through this process, weak classifiers are eliminated from classifier systems, strong ones are retained and new ones are generated applying genetic operators in the expectation of introducing even more successful recombinations. The entire process is schematised in Figure 79.

Classifier system architecture differ from traditional production systems architectures in several respects: the former are suited to a parallel activation of rules, they have no permanent memory and employ only simple syntactic matching or rule conditions. Most production system architectures assume that all of the rules are matched against the working memory in some way. Classifier systems make an additional assumption however: namely, that in any one time step, more than one activated classifier may add its message to the message list simultaneously. Thus, many classifiers may be active at the same time. Large sets of classifiers and a reasonably sized initial message list may yield a highly parallel behaviour. The parallelism of the message list allows competing hypotheses to be maintained, while the problem of conflict resolution (providing that the message list is large enough) among

![Figure 79 Illustration of a learning classifier system's performance (adapted from Wilson, 1994)](image-url)
potentially competing production rules (classifiers) is ameliorated in non-learning configurations because multiple classifiers can then be activated at one time. In learning classifier systems, a form of conflict resolution has to take place, however, because the message list is restricted and messages compete to become active. Even here there is a domain independent way to decide which messages are posted through the bidding mechanism. Conflict resolution plays a major role in production systems that allow arbitrarily complex pattern matching of the left-hand part of a rule against the state of the working memory.

```
procedure CFS
begin
  t=0;
  initialise MessageList ML(t);
  initialise classifier Population(t);
  while termination criterion not satisfied
    t=t+1;
    ML(t) :=readDetectors(t);
    ML'(t) :=matchClassifiers ML(t), P(t);
    ML'(t) :=selectMatchingClassifiers ML'(t), P(t)
    ML'(t) :=taxPostingClassifiers ML'(t), P(t)
    ML(t) :=sendEffectors ML'(t);
    C(t) :=receivePayoff(t);
    P'(t) :=distributeCredit C,P(t);
    P"(t) :=generateNewRules P'(t);
  end
end.
```

Figure 80 A Pseudo Code for a Learning Classifier System (LCS)

Another way in which classifier systems differ from standard production systems is that they have no permanent memory beyond the information encoded in the rules (classifiers). This is to say that messages on the message list disappear after one time step unless they are 'refreshed' by an active classifier. The absence of a permanent memory can be disadvantageous in situations where static information needs to be maintained on the message list over a large numbers of iterations. Since most information can be quickly regenerated when needed (that is, the classifiers are themselves a form of long-term memory), the need for this operation should not arise very often. However, in cases where it is a problem, some forms of secondary memory could be implemented by partitioning the message list into the regions in which messages persist for more than one time step (Forrest, 1991a).

Because the syntax of each individual classifier is so simple, the inner loop of the system (the matching algorithm) is fast by comparison with the complex pattern-matching algorithms used in OPS-like languages (Forgy, 1984). Although the classifier system does not have the ability to specify arbitrary patterns with pattern variables, it does have the # syntax, which allows some repetitive patterns to be represented. This design decision trades-off representational power for efficiency and learning capacity, and indeed it was originally based on cognitive
modelling considerations (Holland et al., 1986). As a result, each classifier typically represents a much smaller fragment of knowledge than does a standard production rule. Thus clusters of rule sets must be developed to represent complex patterns in the environment. This leads to a finer-grained representations and decreases the amount of complex computation (the size of the reasoning step) required over each cycle of the system.

Since connectionism has become such a dominant approach to modelling sub-symbolic cognitive processes, a brief comparison of classifier systems and connectionist algorithms is appropriate. The two models are similar in the following respects: (1) they are both, in principle, massively parallel, (2) the elementary units are limited computationally and denote 'sub-symbolic' activities, (3) learning is an essential component, and (4) knowledge is active rather than static. In fact, it is possible to view classifier systems as a kind of connectionist model by interpreting each classifier (rule) as a link and each message as a node. However, this analogy, which has been investigated recently by several authors (Farmer, 1991; Belew and Gheritty, 1989), also reveals some of the key ways in which these paradigms differ, namely with respect to: (1) the specification of patterns of connectivity, (2) the synthesis and evolution of connections, (3) their use of computing with recursive connections, and (4) the application of intermittent as opposed to continuous reward.
Classifier systems use a pattern language in which one rule can represent a multitude of connections. In this sense, each rule is on a somewhat higher level than is each synaptic connection in a connectionist network. Both models have learning algorithms that adjust the 'strengths' of individual connections (the bucket brigade in the case of the classifier systems and back propagation in the case of the connectionist models). However, classifier systems use an additional mechanism - the genetic algorithm - to synthesise connections dynamically. In the typical connectionist model, the initiating set of connections is complete (that is, all possible links between input and output nodes are represented). The learning algorithms then adjust weights such that links whose weights fall below some threshold are removed from the equivalent graph. In contrast, classifier systems are initialised with a random set of rules that represents a tiny fraction of the space of all possible rules, and the learning algorithm evolves the rule set, as well as adjusting the relative strengths of the different rules. Taken together, these two features of classifier systems - the use of a pattern language and the ability to evolve the rule set - results in the ability to restrict the manipulations to only those connections that are the most relevant. For large-scale systems, this will be a very small fraction of the total number of possible connections. Another distinction between classifier systems and connectionist networks is the way in which they handle recurrent connections. While both models can represent recurrent (looping) structures, the vast majority of connectionist applications uses feed-forward networks and there are major difficulties with using the common error-propagation algorithms on recurrent networks. Classifier systems, on the other hand, were designed to be recurrent. Finally, classifier systems can function with intermittent rewards, unlike connectionist models which provide an error correction at every iteration of the system. This feature of classifier systems can be an advantage in domains where instantaneous feedback is not available.

One apparent drawback of classifier systems is that each classifier reads (examines in its entirety) a potentially large message list, of which most of the messages may not be relevant. Having each classifier read the entire message list, of which most of the messages may not be relevant, introduces a time-consuming search. However, it is easy to imagine messages being routed directly to the classifiers that they will activate. Because the format for expressing the condition parts of a classifier is so restricted, it would be possible to sort the conditions of any given list of classifiers so that messages could be routed efficiently. In this case, each classifier would have to read only the relevant messages. Messages that were relevant to many classifiers would still be effectively global, but messages that were relevant to only one classifier would be read only by that classifier. This 'data-flow' approach would not change the overall behaviour of the system (although it would certainly affect the system's time complexity), and is therefore considered to be best implemented at the level of the hardware. We shall here assume that only its relevant messages reach each classifier.

6.9. Genetic Programming

Genetic Programming (GP) is an extension of the Darwinian computational model of learning into the space of programs. The first ideas with respect to the evolution of sequential programs can be traced back to Cramer (1985). The basic notion of GP in its present form was first introduced by Koza (1990, 1992). Whereas Genetic Algorithms typically operate by combining binary strings which encode real-valued independent variables (see, for example, Babović, 1993), in the case of GP the symbolic expressions are subject to genetic operators (recombination and mutation).
6.9.1. System Identification Problems

The main intention is to create an artificial environment populated with artificial entities capable of reproducing and equipped with hereditary capabilities. In an evolutionary situation, these entities should try to adapt themselves to the environment and achieve the best possible performance in that environment.

The problem of creating an artificial environment can be equated to a system identification problem, since modelling in the sense of 'systems identification' can be defined as the process of generating closed-form mathematical explanations of observed data. Indeed, from this point of view it has been claimed that (Ljung, 1987):

"...inferring models from observations and studying their properties is really what science is about."

So long as we understand the word 'model' in the broadest sense, as 'a collection of signs that serves as a sign' (Abbott 1992, 1993), this can be accepted in the present context. Rather more narrowly, Caines (1988) regards system identification as the invention and evaluation of scientific theories, that is, system identifications performed using a scientific method such as that defined by Pierce (1880-1900/1956; see also Amdisen 1994). This method involves abduction, deduction and induction or inductive inference, followed by independent verification. It is an iterative process that facilitates gaining new knowledge about the nature of an observable environment using a purely logical grammar in the sense of Husserl (1900/1931/1970; see also Derrida, 1967/1993). System identification techniques are applied in many fields in order to predict the behaviour of unknown (or not-so-well-known) systems given input-output data.

Following (Iba et al, 1993) we may define the problem in the following way. Imagine that a system produces an output value, $y$, and that this $y$ is dependent on $m$ input values, thus:

$$y=f(x_1, x_2, x_3, \ldots, x_m)$$

Given a set of $N$ observation of input-output tuples, i.e.

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{i1}$</td>
<td>$y_1$</td>
</tr>
<tr>
<td>$x_{i2}$</td>
<td>$y_2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x_{in}$</td>
<td>$y_N$</td>
</tr>
</tbody>
</table>

*Figure 82 An input-output example set*
the system identification task is to approximate the true function \( f \) with an approximate function \( \tilde{f} \). Once this approximate function \( \tilde{f} \) has been estimated, a predicted output \( y \) can be found for any input vector \( (x_1, x_2, \ldots, x_m) \), i.e.

\[
\tilde{y} = \tilde{f} (x_1, x_2, x_3, \ldots, x_m)
\]

The \( \tilde{f} \) is called the 'complete form' of \( f \). This is to say, we are searching, using evolutionary algorithms, for a complete form of \( f \).

In a similar fashion to what occurs in natural evolution, evolutionary algorithms can be best understood as general-purpose search algorithms\(^{57}\). They use computational models of evolutionary processes as key elements in the design and implementation of computer-based problem solving systems. All evolutionary algorithms share a common conceptual base of simulating the evolutionary processes of individual structures through sub-processes of selection, mutation and recombination.

### 6.9.2. Reverse Polish Notation

The power of evolutionary algorithms resides in their domain independence. The connection between recombination operators and the domain is represented only by the mapping, and this mapping is left entirely to the freedom and responsibility of the modeller. These mappings range from mappings to real-valued parameter values (Babovic, 1993), through the derivation of town-visiting ordering in the well-known travelling salesman problem (Manderick, 1992), to high-level knowledge representations (Antonisse and Keller, 1987). The only requirement that the mapping must fulfil is that of alphabetical closure, meaning that, through the application of the mapping, every genotypical individual should produce a legal phenotypical counterpart (i.e. the phenotype should be a well formed formula in a grammar of a particular language of phenotypes).

This requirement, however, can be, rather expensive in computational terms. In such cases, otherwise rather straight-forward crossover and mutation operators should be appended with more elaborate parsing techniques and even facilities for patching ‘incorrect’ phenotypes should be introduced.

Instead of using the more conservative notations of genotype-to-phenotype mapping, a convenient way to represent algebraic expressions for high-level representation problems is that of Reverse Polish Notation (RPN) or prefix notation (Walker et al, 1993). Using RPN, a formula like \((x + y) * 3\) becomes \(* + x y 3\). Note that brackets are no longer necessary. There is a one-to-one correspondence between the standard infix and the prefix notations for algebraic formulae.

---

\(^{57}\) One might argue that natural evolution cannot be expressed as an algorithm. It is, therefore, more correct to state that natural evolution is a means by which an organic entity (its genotypical lineage) adapts itself to its environment and, correspondingly, searches for a configuration by which it most profits.
Figure 83 An algebraic expression as a tree and in prefix (RPN) and standard (infix) notation

A RPN formula can be represented as a tree. Nodes in the tree can be non-terminal and associated with operators like + or *; or they can be terminal and represent operands like x, y or 3. The terminal nodes are termed the leaves of the RPN tree. Non-terminal operator nodes have as many ‘children’ as they have operands. The number of children is called the arity of the operator.

RPN trees have some very useful properties. We shall define them here, and make use of them later in the text. Thus following (Walker et al, 1993):

- A node associated with a constant is a well-defined RPN tree;
- A node associated with a variable is a well-defined RPN tree;
- If $\xi_1, \xi_2, \xi_3, ..., \xi_n$ are all well defined RPN trees, then $\xi_{n+1}$ is a well defined RPN tree iff:
  - the root of tree $\xi_{n+1}$ is associated with an operator of arity n;
  - $\xi_1, \xi_2, \xi_3, ..., \xi_n$ are all children of $\xi_{n+1}$.
This has a very useful consequence, namely that replacing a child node of a well-defined RPN tree by an arbitrary well-defined RPN tree yields a well-defined RPN tree, as is clearly depicted in Figure 86.

6.9.3. Search Operators
6.9.3.1. Crossover or Sexual Reproduction

The crossover operator is a higher-order transformation of parental algebraic expressions. Due to the advantageous features of the prefix notation, it is relatively easy to implement the crossover operator and indeed it is not necessary to employ a more complex parsing algorithm. The formula syntax remains correct (well-defined), regardless the crossover site. The principle underlying the crossover operator is depicted in the Figure 86.

The crossover $\Theta: \Omega \times \Omega \rightarrow \Omega$ between two RPN trees $\xi_1, \xi_2 \in \Omega$ can be defined through the following procedure:

- randomly select a node, $n_1$, in the tree $\xi_1$
- randomly select a node, $n_2$, in the tree $\xi_2$
- swap the two sub-trees

In general, the term crossover is used to describe a trading of information between genomes, similar to diploid sexual manipulation. The procedure above describes one-point crossover, since only one crossover site per RPN tree is selected. In general, multi-ploidal reproduction with multiple (and even uniform) crossover can be realised in the virtual world and certainly in particular cases this could be advantageous, but it has found little practical application to date.

---

58 It is often said that evolutionary algorithms are randomised search procedures. The concept of 'random' is indeed a difficult one. What we imply under random is perhaps best described by a biologist, as follows:

It is sometimes said - usually by critics of Darwinism - that mutation is random. Now 'random' is a notoriously difficult word to define. I think that most scientists, when they speak of an event being random, mean that it would not be efficient to enquire into its causes, either because they think that the cause is in principle unknowable, as in quantum theory, or because it would be too much trouble to discover. In this sense, mutation is certainly not random. A lot is known about the causes of mutation - that is, of changes in DNA. However, most geneticists do hold two things to be true of mutation. First, there is no restriction on the kinds of changes in sequence of DNA molecules that can arise by mutation, any more than there is any restriction on the sequence of letters that can be produced by a typewriter. Second, if a mutation is caused by a particular agent - for example X-rays - it is not generally true that the effect of the mutation will be to make the organism more resistant to the causative agent: in brief, mutations are not adaptive.

Maynard Smith (1986, p.40)
Figure 85 A node, which is a legal RPN tree indicated under (a), is replaced by a another legal RPN tree (b) which yields a completely new RPN tree (c)
6.9.3.2. Mutation

Mutation, \( \mu \), is a unary-type transformation that alters individual \( \omega \in \Omega \) so that \( \mu: \Omega \rightarrow \Omega \). Each evolutionary-algorithmic technique defines mutation in a sense that best suits its own purpose. Biologically speaking, mutation denotes the haploid, asexual manipulation of the genome. Traditionally, in genetic algorithms, mutation is referred to as a bit-flipping or a segment-inverting operator (Babovic, 1993; see also section 6.5.2). In Evolutionary Programming, mutation is understood as any such manipulation of a structure (Fogel, 1992; see also section 6.7). Researchers in Evolutionary Strategies use the term to describe an even broader variety of different operators.

In Genetic Programming, however, the action of the mutation operator has frequently been described as a random substitution of a sub-tree with another sub-tree. There are several kinds of (computational) mutations possible. In the computer system used here, several forms are implemented:

- *branch-mutation*, where a whole sub-tree is replaced with another, in-principle
arbitrary sub-tree;
- *node-mutation* (constant, operator, parameter), which applies a ‘random’ change to a single node, replacing its original value by another, and again in-principle, arbitrary value;
- *inversion-mutation*, which randomly inverts the order in which operands are ordered in an expression: thus \( f(x_1, x_2, x_3) \) might become, for example, \( f(x_3, x_1, x_2) \).

### 6.9.4. Selection

Natural selection is the blind watchmaker, blind because it does not see ahead, does not plan consequences, has no purpose in view. Yet the living results of natural selection overwhelmingly impress us with the appearance of the design as if by a master watchmaker, impress us with the illusion of design and planning.

(Dawkins, 1986, p.21)

The purpose of parent selection is to provide more reproductive opportunities, on the whole, to those population members that are the more fit. It is the selection criterion which boosts the more fit individuals to participate in reproduction and thus propagate the information they carry to their offsprings. Obviously, it is the selection criterion which makes the fundamental difference between evolutionary algorithms and random search. Random search implies a uniform random distribution of trials in the search space and corresponds to a process extending over the total number of evaluations in one GP run without any memory participating in the process.

It must be emphasised that selection should not be modelled as a deterministic process. This is because, firstly, selection in nature is not deterministic, and, secondly, it has been observed that deterministically modelled selection introduces more drawbacks than advantages.

Therefore, selection must be understood as a process that indicates, rather, some sort of *stochastic priority*. There is a number of selection mechanism. We use the most fundamental model of *fitness proportional selection*. Some authors (*e.g.* Goldberg, 1989) again compare this selection mechanism to playing a biased roulette wheel. On a biased roulette wheel, every chromosome is, once again, assigned an area proportional to its fitness. However, through the even simpler process of ‘rolling the dice’, there is still some likelihood that the weakest candidates could survive. The main idea is to keep the sampling process probabilistic, while favouring stronger individuals.

### 6.9.5. Calculation of Fitness

The success of a formula for fitness depends on the perceived performance as defined by its environment. The fitness of an expression thus depends on many factors, the most important of which is the degree of error it produces over a set of samples as defined in Figure 82. More formally, an error of a complete form \( f \) may be calculated as the sum of the absolute values of errors:
Another possibility may be to use the maximum error, the sum of squared errors, a coefficient of determination, a coefficient of efficiency, or indeed whatever might be felt appropriate.

\[ \varepsilon(\mathcal{F}(x)) = \sum_{i=1}^{N} |\mathcal{F}(x) - y| \]

6.9.5.1. Parsimony bonus

Any data set can be approximated by an expansion of a series of basis function. But, this is not the main goal when using this technique. Consequently, a method must be devised which will promotes a parsimony of algebraic expression. In this case, through the application of search operators, the most successful parts of different formulae are recombined into even more successful offspring formulae which, in return, mirror realistic, physical relationships in the problem domain. As an additional, stimulating incentive for propelling codes towards simpler, more compact expressions, a fitness evaluation may be introduced into them that contains an additional bonus for simplicity of expression. For example, following (Walker et al., 1993):

\[ \text{Lh = concave2(vex_cave(expr(0.02 ^ (K ^ vex_cave(Fr(vex_cave(exp(concave2(1.22 ^ (Ea ^ hollow(vex_cave(Ea, s2(vex.} \]

**Figure 87** An illustration of a tree structure of a complex expression

---

59 As the generalised schemata theorem states that the most successful parts of a chromosome receives an exponentially increasing number of evaluations. In other words, the application of search operators identifies the most successful 'pieces' of (in this case) algebraic expression and allocates an exponentially increasing number of search operators applications as evolution progresses. This takes place for all successful (pieces of) formulae and therefore a recombination of the most successful contributions may produce an even more successful overall solution.
\[ \Gamma(l) = Ke^{-\frac{(l-\delta)^2}{2\sigma}} \]

where:
- \( \delta \) - estimated optimal formula size (length)
- \( \sigma \) - measure of tolerated deviation from \( \delta \)
- \( l \) - the actual formula's size (length)
- \( K \) - a parameter determining the influence of the formula size on the total fitness

Note that expression above denotes a bonus and that it should be subtracted from the error calculation.

6.9.5.2. Minimum descriptional complexity

The problem of GP can now be understand in terms of hypothesis formation. GP evolves an RPN tree that 'explains' a set of data (or satisfies a set of constraints expressed through data) as evaluated through its fitness function. More complex description of a hypothesis the volume of missclassified data is minimised. If hypothesis describes all the data, then it does not allo for measuring errors. A simpler description of hypothesis may be penalised by increasing the misclassified data. If hypothesis is trivial and it contains nothing, all data are described literally and there is no generalisation. The rationale of the method is that a ballance of the two extremes is required.

The best theory to explain a set of data is the one which minises the sum of the length (in bits) of the description of the theory; and, the length (in bits) of data when encoded with the help of the theory. This is so-called minimum description length (MDL) principle (Rissanen, 1978). Stated slightly differently, the MDL principle can be re-stated as that given hypothesis space \( \mathcal{H} \), a hypothesis \( H \) should be selected such that the length of the shortest encoding of observed (training) data \( D \) with hypothesis \( H \) is minimal. Similar approach will be further explored in Section 6.12., however, with another objective. Here, we introduce various measures of complexity of RPN trees that should be minised in a process of hypothesis formation.

Imagine an individual RPN tree that is composed of a number of functions (like \( \sin, \ln \), etc.) and a number of terminal nodes (constants or variables). Following Rosca and Ballard (1994) the structural complexity of an expression \( F_{0} \) may be defined as:

\[ SC(F_{0}) = \sum_{1 \leq i \leq m} \text{Size}(F_{i}) \]

where \( \text{Size}(F_{i}) \) represents a number of nodes in a tree \( F_{i} \). The evaluational complexity \( EC(F_{i}) \) (i.e. the measure of the program's execution difficulty) can be addressed by introducing:
\[ EC(F_i) = \text{Size}(F_i) \sum_{i<j \leq m \text{ or } \text{called} \in F_i} EC(F_j)\text{\#Calls}(F_i, F_j) \]

where \(\text{\#Calls}(F_i, F_j)\) is the number of times \(F_i\) calls \(F_j\). Assuming that all generic functions can be executed in the same time, the evaluational complexity shows how many time units it takes to execute an individual program.

6.9.6. Generic and custom operators

The success of GP largely depends on the function set upon which it operates. As a basic (generic) function set, the GP computer system used here was composed of the following arithmetic and control operators:

- unary operators \{signum, sin, \sqrt{}, \exp{}, \ln{}, not\}
- binary operators \{-, +, *, \^, >, <, =, !=\}
- ternary operators \{if-then-else\}

In addition to these, it is possible to define a set of so-called custom operators. Thus, for example, since there is no cosine defined, but it is always possible to define \(cos=f(sin)\), and hence \(tan=f(sin, cos)\), and so on.

Obviously, the definition of a custom operator set considerably facilitates the search process. The definition of custom operator set may indeed be understood as an incorporation of domain-specific knowledge into the search procedure. However, in order to keep the present experiments as transparent as possible, no custom-made operators were used.

![Custom operator WAVES: the algebraic expression and tree structure of the operator](image.png)
6.10. Problems

More or less every computational technique has its problems and its deficiencies. Evolutionary algorithms are no exception. In order to tackle these problems, their source must be identified at its origin. Clearly, if the evolutionary model is to be successful, it must provide facilities that keep the influences of these problems under control. At the origin of most of the problem with Evolutionary Algorithms are the sub-problems of premature convergence and lack of population diversity.

A classical problem with EAs is that the genes from a few comparatively fit individuals may rapidly dominate the population, causing it to converge to a local optimum. This can be readily seen from the Schemata Theorem which states that, under fitness proportionate selection, reproductive trials are allocated in proportion to the relative fitness of an individual. This can sometimes result in premature convergence. Since the population is finite, the Schemata Theorem states that inevitably, under fitness proportional selection, evolution leads to convergence to an optimum with the largest basin of attraction. Since the size of the basin of attraction is not related to its fitness, the entire process may converge to a local optimum.

Another problem with evolutionary algorithms is one of slow finishing. In the closing stages of evolutionary processes as simulated by using genetic algorithms, the entire population may converge to the same region of the search space containing the global optimum. Thus, the entire population will be similar, and correspondingly, their fitnesses close to one another. In such a situation, the genetic algorithm has very little incentive to prefer one solution configuration to another, and therefore converges relatively slowly to the global optimum.

In order to avoid these problems, a number of techniques have been invented, and some of them will be discussed in more depth later in this text (e.g. sub-section 6.10.2, on Parasites). Here, more traditional approaches to the control of these problems will be discussed. The first of these approaches, called fitness scaling, basically alters the fitness landscape in order to re-define its properties in a way that will navigate the evolutionary search more successfully through the search space, whereas the other class of techniques controls the parent selection.

Premature convergence is a recurring problem in Evolutionary Algorithms. Imagine, for example, that in a first, randomly created generation, the EA generates, based on pure coincidence, an extremely strong, but not completely optimal individual. Such an extremely fit individual would soon dominate the population and we should observe that the population is fixed on a sub-optimal genotype. This happens because of the fact that the fitness of this individual entity is still much higher than the fitness of the remainder of the population. In a selection procedure, this locally super-individual gets chosen much more often than does the remainder of the population. Obviously, the population prematurely converges to this local optimum and, since the population diversity is very low and indeed almost non-existent, it has very little opportunity to move out of this position. The only event that can detach the population from this position is that of a favourable mutation. But, bearing in mind that the probability of a mutation is much lower than is the probability of crossover, the probability of occurrence of that favourable mutation might be extremely low.

It is generally accepted that, within an ordinary EA framework, a lot of computational effort
is wasted. The prevailing view is that intermediate solutions are located at local (sub-) optima. This has an extremely negative consequence in situations with low population diversity and thus an inability to move to another, more profitable position in a fitness landscape. In order to ameliorate the performance of an evolutionary algorithm, many different methods have been tested.

Probably the most important factors for promoting the successful performance of any evolutionary algorithm are those of selective pressure and population diversity. But these can, unfortunately, just as well be counter-productive. This means that while the selection criteria is trying to select the most fit individuals, it actually reduces the level of diversity in the population, even as a diversity in the population is extremely important for promoting a successful exchange of information among the different members of the population.

### 6.10.1 Well-established solutions

The obstacle of premature convergence is one of the most serious deficiencies of Evolutionary Algorithms and it has been studied by many investigators. The standard approach is the one of fitness scaling. The main idea is to alter the raw fitness (i.e. that calculated directly by application of the fitness function) in such a way to introduce more diversity into the population. It is then the scaled fitness and not the raw fitness that is used in the selection procedure. In an earlier work (Babovic et al., 1994) linear scaling has been employed, although some authors (Michalewicz, 1992) strongly recommend the use of non-linear scaling techniques. Moreover, in this case the scaling is non-steady in the sense that, in the early stages of evolution, the fitnesses of the stronger individuals are decreased and those of the weaker ones are increased, causing a very high diversity in the population and an intensified sampling of the search space. In the later stages of evolution, the scaling parameters are altered so that the fitnesses of stronger ones are increased and those of weaker ones are decreased even further. The consequence is that the selection process ‘chooses’ only the ‘best’ individuals, wasting very little effort on evaluating less strong individuals.

The other approach, termed crowding, was introduced by De Jong (1975). In his work, the use of overlapping generations is proposed in which new individual strings replace the strings that are genotypically similar. Obviously, individuals compete in a population of a constant size, so that one super-individual never comes to occupy the whole population.

Another approach, clearly inspired by natural phenomena, is that of introducing a sharing function. It has been proposed by Goldberg and Richardson (1987) and basically promotes a situation in which phenotypically similar individuals occupy the same ecological niche and use the same amount of resources. Similarly, within the context of evolutionary algorithms, the raw fitness is reduced by an amount calculated by a sharing function. The sharing function is directly proportional to the number of individuals in a ‘niche’, and consequently promotes the diversity. The only problem with such an approach is that the sharing function is strongly domain-dependent, often difficult to define and consequently demanding for practical use (due to phenotypical similarity).

Other, more recent approaches take a line of selective or restrictive mating. In certain cases, only similar individuals are allowed to mate in order to stimulate inbreeding and positive
associative mating (Crow, 1986). In the other case, of using a strategy of incest prevention (Eshelman and Schaffer, 1991), a completely opposite effect is achieved, namely that of negative associative mating (or forced outbreeding).

All of these approaches seem to have achieved excellent results in selected environments and for selected problems. However, all of these techniques require a rather high degree of domain-dependent knowledge if they are to be incorporated into an algorithm, so making that algorithm more difficult to implement and use.

As in so many cases, the right inspiration comes from the study of nature. In natural populations there are no superimposed rules forcing individuals to mate with a certain partner so as to maximise the pay-off of the population as a whole. Furthermore, there are no forces that alter (or 'scale') the individual's fitness. At the same time, the immense genotypic and phenotypic diversity of species on the Earth deny the possibility of natural evolution suffering from 'premature convergence', or of being locked into some super-individual, dominating the population - and all that after three million years of evolution.

6.10.2. Parasites

One of the most successful means for maintaining a high population diversity in numerical environments, to date, is the introduction of a co-evolutionary processes. The basic idea is extremely simple and was originally presented by Hillis (1992): it is to introduce in the evolutionary process a set of competing and even counterproductive entities - parasites - which interact with the test cases. The role of the parasite is to select those test cases on which the GP expression performs poorly and eliminate the corresponding genetic material.

The parasite's persistence, modelled through its 'length' plays an important role. For example, if the number of training cases is N (see Figure 82), the parasite's length would correspond to the number of cases that is manipulated by the parasite, say $m < N$. The parasite's fitness is inversely proportional to the fitness of the GP expression. In such a co-evolutionary environment, the role of the parasites would be to generate a subset of all training samples on which an expression performs poorly and eliminate the corresponding genetic material.

The consequence of such a co-evolutionary process is that the population is kept in a continuous transition from 'epidemic' to 'immune' (that is GP expression discovers the 'trick' to handle difficult subset in training data). The systems with fixed, non-dynamic fitness tend to get fixed on a sub-optimal individual, while the systems with dynamic fitness show no such tendency, even after very long simulations.

The other benefit with such a co-evolutionary approach is that the action of the parasites alters the fitness landscape, and in principle makes is more 'walkable'. This is depicted in the Figure 89. Whereas the upper part of the figure provides a 'static' view on a fitness landscape, the lower part illustrates how such landscape is altered under the action of

---

60 In effect, an inappropriate use of these techniques might actually cause entirely negative effects: the convergence rate might actually be much lesser than that obtained with an ordinary algorithm.
Figure 89 The upper part of this figure presents a 'raw' fitness landscape with most of the solutions locked on to local optima, while the lower part illustrates how the fitness landscape is altered under the action of parasites, in such a way as to provide a more 'walkable' landscape and thus the possibility for a solution point to converge towards a global optimum.
parasites, decreasing the number of local optima and therefore making the task of the GP algorithm easier.

However, the use of parasites requires extreme caution. Lethal parasites in nature become, at some stage, self-destructive: they eliminate their hosts and soon have no place to live. A very similar phenomenon can occur here. If the parasite is chosen to be very strong (say to be of the length of \( m=5 \) out of \( K=100 \)) it means that the (dynamic) fitness of a GP expression would be evaluated on the basis of these five sample points only. Since these five sample points do not represent the data set in its entirety, there is very high risk of aliasing the data set into something completely different from that which entered the original problem. The way parasites are used in the present framework is essentially exploratory. Initially, a rather weak parasite configuration is chosen, and then after every 50-100 generation its length is shortened, making it stronger and stronger. At the end of the co-evolutionary process, it is often observed that few completely new algebraic expressions have been created that these are much more fit than those that were the most viable before parasites were introduced.

6.10.3. Panmictic and Distributed Populations

Although initially a cause of much controversy in population genetics, it is now widely accepted that the spatial structure of a population plays an extremely important role. Generally speaking, evolving population can be seen as 'uniform', or panmictic (the view presented by Fisher 1930), or spatially bounded and correspondingly divided into a number of sub-populations, or demes (as expounded by Wright, 1932). The differences between the two approaches devolves upon the question of the prevalence of panmixia in natural populations. Wright's point of view seems to be more appealing in the present context, so that it is accepted that the structure of the population is extremely important in the simulation of evolution. Moreover, Collins (1992) performed a large number of numerical experiments showing the superiority of 'demetic' populations against panmictic, at least within the artificial environment of genetic algorithms.

The main idea is, again, very simple. The entire population is divided into a number of pools or demes. Demes are genetically semi-independent sub-populations that remains loosely coupled to neighbouring demes by migrations. Through the application of a migration operator, the contents of two, randomly selected demes is mixed, and then separated again. In such a way, two sub-populations that are possibly trapped at local optima, exchange information that can be critical to further, successful evolution. The probability of applying migration is very low in order to allow for an independent evolution. The local stochastic effect (genetic drift) associated with the spatial structure can have a dramatic effect on the composition of the gene pool of a population. The relatively small size of demes allows drift to play an important role in the evolution of the population, without driving the whole population toward fixation. The shifting balance of small populations results in a rather fast and successful search. Even if drift were to drive every local subpopulation to fixation, each deme would be fixed on a different genotype, so maintaining the diversity of the population as a whole.
Figure 90 Panmictic view on the population. The entire population is structured in a number of demes (pools) which are migrating with a certain (very low) probability.

Figure 91 Software perspective on a panmictic population.
6.11. Some Applications of Evolutionary Algorithms

6.11.1. Calibrating hydrodynamics models by means of simulated evolution

Adaptive evolution can be regarded as a complex combinatorial optimisation process. This is not, however, an optimisation in the usual sense of the term. Rather, one thinks of evolution as a strategy for exploring and adapting a set of genotypes that will provide phenotypes that can survive in complex ‘fitness landscapes’. Evolutionary algorithms together constitute a category of algorithms which represent a crude simplification of natural evolution—but provide efficient and robust search strategies. This chapter describes an application of a genetic algorithm to the problem of the automatic calibration of a hydrodynamic model.

6.11.1.1. Introduction

There is constant need in technology generally, and in water-related activities in particular, to measure the performance of modelling tools by calibrating and validating them using measurements made in the real-world. For example, in sewerage system modelling, measured hydrographs are typically available at several control nodes. In a so-called structure-orientated modelling paradigm, one then searches for a set of parameters that are to be 'fed' into the model, and which model will, in its turn, provide an output that is in accordance with the observed data. Once the set of parameters governing the model’s behaviour is found, the model can be used to predict outputs for a number of events that differ reasonably widely from those providing the ‘training’ sets.

Calibration is therefore an extremely important task. However, it requires an auxiliary knowledge from an engineer. For example, it requires knowledge about the model’s behaviour, which is often not an engineer’s strongest point. Moreover, calibration is a time-consuming exercise: it requires a lot of ‘test-runs’ and patience.

Furthermore, in the hydroinformatics era (Abbott, 1991) in which the modelling systems are more and more used to control processes in real-time and are interfaced to the real-world through SCADA systems, the level of requirements is even higher. The data from the physical environment are available on-line and the need for a robust and efficient algorithm which monitors the values of, for example, roughness coefficients, and adapts them automatically to meet the required accuracy is of prime importance.

6.11.1.2. The formulation of the problem of automatic calibration

There are four major steps in preparing to use the canonical genetic algorithm on a fixed-length character string to solve this problem and they involve (Koza, 1992):

1. determining the representation scheme;
2. determining the fitness measure;
3. determining the parameters and variables for controlling the algorithm;

and

61 The work described in this subsection was published in a similar form as Babović et al, 1994
(4) determining a way of designating the results and a criterion for terminating the run.

(1) Representation scheme

In the present application each chromosome, a genotypical individual, defines a corresponding phenotypical representation $\overrightarrow{p} = (M_1, ..., M_i, ..., M_n)$, the sequence of Manning numbers ordered according to the number of the specific pipe to which the individual Manning number is allocated. Thus, there is a $P = P(r,t) = (\overrightarrow{p}_1(t), ..., \overrightarrow{p}_i(t), ..., \overrightarrow{p}_n(t))$. When the Manning number is kept constant in time at each trial $r$, then we have simply $P = P(r) = (\overrightarrow{p}_1, ..., \overrightarrow{p}_i, ..., \overrightarrow{p}_n)$.

Firstly, we describe a methodology for transcribing the genotypical representation of a chromosome, given as a bit string, into the phenotypical representation of Manning number given as a rational number. In our case this is done directly using mapping $M: g_i(t) \rightarrow \overrightarrow{p}_i(t)$, which transforms the genetic representation into the corresponding Manning number. Following the procedure outlined earlier, the Manning number vector representing the entire system is given by:

$$\overrightarrow{M} = (M_1, M_2, ..., M_n)$$

with

$$M_i = M_i(g_{i,1}, ..., g_{i,m}, ..., g_{i,n})$$

If $M_i \in [\overrightarrow{M}_i, \overrightarrow{M}]$ we could use:

$$M_i = \overrightarrow{M}_i + \frac{\overrightarrow{M} - \overrightarrow{M}_i}{b^{i-1}} [\sum_{m=1}^{m} a_{i,m} b^{m-1}]$$

where

$$b = \begin{cases} 2 & \text{for binary coding;} \\ 10 & \text{for natural number coding;} \end{cases}$$

In order to reproduce the next generation, the parents should be selected from the current population by performing the operator selection $\sigma_{pc} : I^p \rightarrow I^p$, where $I^p \subseteq I^p$ is the space of parents of the next generation. New individuals represented genotypically by the set of chromosomes can be produced by applying the recombination. Through this operation, $\Theta_{pc} : I^p \rightarrow I^o$, where $I^o$ is the space of offspring, representing the new generation. Finally, $\Theta_{pm} : I^o \rightarrow I^p$ is introduced to maintain the diversity of the population.
(2) Fitness measure

Our objective is to calibrate on Manning numbers $M_i$, $i \in \{1,2,\ldots,n\}$, over all pipes $i$, such that the distance $\rho(X_s, X_m)$ taken over the set of all simulated results $X_s$ and the set of all corresponding measured $X_m$ is minimized. For each node point $j$ and each time level $n$ for which both measured and simulated results are available, $X_s=\{(x_s^n)_j, j=1,2,\ldots,J; n=1,2,\ldots,nn\}$ and $X_m=\{(x_m^n)_j, j=1,2,\ldots,J; n=1,\ldots,nn\}$, where we use the notation $\{\}$ to represent the set so enclosed.

In order to minimize $\rho(X_s, X_m)$ we execute a number of simulations $r=1,2,\ldots,r$ corresponding to each set of the measurements, keeping the set $\{M^n\}$ constant over $n$, so that $M^n=M_i \forall i$, but varying the set $\{M^n\}$ with $r$, so that $M^n=M_i$. We thus generalise our set of simulated results further by introducing $X_s=\{(x_s^n)_j, j=1,\ldots,J; n=1,\ldots,nn\}$. We may then pose the objective of our automatic calibration procedure as follows:

Minimise: $\rho(rX_s, X_m) |_{X_s=\{(x_s^n)_j, j=1,\ldots,J; n=1,\ldots,nn\}, X_m=\{(x_m^n)_j, j=1,\ldots,J; n=1,\ldots,nn\}}$

We can, as usually, define the distance function $\rho(X_s, X_m)$ in several different ways, and primarily and in the present case as follows:

\[ \rho(X_s, X_m) = \left[ \sum_{j=1}^{J} \sum_{n=1}^{nn} (x_s^n_j - x_m^n_j)^2 \right]^{1/2} \]  

\[ \rho(X_s, X_m) = \left[ \sum_{j=1}^{J} \sum_{n=1}^{nn} (w_j^n(x_s^n_j - x_m^n_j)^2) \right]^{1/2} \]  

\[ \rho(X_s, X_m) = \max_{j,n} |(x_s^n_j - x_m^n_j)| \]

The difference between (a) and (b) arises from our desire to put more weight upon the greatest discrepancies between simulated and measured results. For example, some points in the model may at a certain time provide much greater discrepancies than other points, but the locations of these greatest discrepancies may change during the course of the simulations. Clearly our weights must also satisfy the usual condition that they sum to unity at any time
step, i.e.

$$\sum_{j,n} W_j^n = 1$$

One possibility in this case is to set:

$$W_j^n = \frac{(x_j)^n}{\sum_{n=1}^{n_0} (x_j)_j^n}$$

The third difference formulation, (c) above, has proven interesting as a means of accelerating convergence toward the end of the optimization process.

For the purpose of obtaining the set \{X^n\} at any r we use the MOUSE system, while for the purpose of varying the simulation of MOUSE by setting the \(M_i^n\), and so for proceeding from one value of r to the next, a genetic algorithm (GA) has been used.

Each sequence \(\bar{p}\) of Manning numbers gives rise to a set of simulated results extended over the networks of node j and time level n, \(r, X_s = (x^n_j)\), and thus a distance \(\rho(X_n, X_m)\). We can define the fitness function of the population in many ways, but we shall consider the following:

$$\phi_r = 1 - \frac{\rho(X_s, X_m)}{\max_{r=1,...,r} \rho(X_s, X_m)}$$

This has the desirable property that \(0 \leq \phi_r \leq 1\) and it is \(\Phi_r\) that must be minimised in r.

(3) Algorithm control

The control of the genetic algorithm involves the determination of its parameters with the purpose of accelerating convergence towards the solution of the problem. In the present context this means:

- the determination of the population size and its influence on the convergence rate
- the determination of the influence of fitness scaling on the convergence
- the determination of the influence of crossover on the convergence rate

As a result of a detailed analysis (Wu, 1994), the population size was set to 50, with a
uniform crossover operator and with a linear scaling of fitness.

**Figure 92** The plan of the Ålborg sewerage system
(4) Termination of a run

The termination of a GA run is a non-trivial operation. Indeed, where is the 'end' of an evolution? It is often rather difficult to measure the degree to which a potential evolution has evolved. In our case, the termination of convergence can be imposed in a number of ways: by the degree of agreement between simulated and measured hydrographs; by the maximum number of fitness evaluations and, within the real-time control paradigm, by the time available to perform the calculations.

6.11.1.3. Calibration results and discussion

A typical urban drainage system, that of the city of Ålborg in Denmark, has been studied in order to investigate the behaviour of the genetic algorithm. The system is simplified into a network of 117 pipes in which there are nine measurement points - six node water levels and three pump station discharges. For each pipe, one parameter, the Manning number, is specified to be calibrated so that the total set of parameters which are calibrated is composed of the 117 Manning numbers.

A genetic algorithm, based on the discussion and denotation shown above, has been implemented for the automatic calibration of the MOUSE pipe flow model using the algorithm already outlined. The calibration results are presented in Figure 95.

![Image of graphs](image-url)

**Figure 93** The selection of some typical calibration results
The calibration results, as shown in Figure 95, appear to be very satisfactory. They have been obtained as a result of 10-generation simulations. Bearing in mind that the population size has been set to 50, the MOUSE hydrodynamic simulation has been called 500 times to evaluate the goodness of fit using difference type (c) for each genotypical individual. The maximum difference between simulated and measured values in level is less than 10 centimetres in this calibration.

![Calibration Test](image1)

**Figure 94** The presentation of degree of success of GA during the 10 generation long convergence

Figure 94 depicts the results of another calibration procedure. In addition to the behaviour of measured and simulated water levels during the simulation time, the figure illustrates the best performance of MOUSE system with Manning numbers set in the generation 0. It should be observed that the roughness coefficients at the first generation were randomly generated. Therefore, the difference between the resulting simulation patterns with initial and semi-optimal parameters exemplifies the success of the GA during the 10-generation long simulation.

It was thus demonstrated that the genetic algorithm, as applied successfully by pioneers in other areas, has another potential application field, namely: the higher-dimensional nonlinear optimisation problems of Hydroinformatics.

6.11.2. Rainfall Runoff Modelling

There is a continuing interest in hydrology to model the relationships between rainfall and stream flow. The issue of developing faster and more accurate rainfall-runoff models still occupies one of the central problem-areas in the research-orientated hydrological community. At the same time, the application-orientated part of the community still requires even simpler, transparent, but still acceptably accurate models.

Several modelling paradigms can be easily distinguished. On the one side, a physically-based, deterministic modelling environment such as the Système Hydrologique Européen - SHE (see Abbott et al. 1986) provide the possibility for a representation of all relevant hydrological

---

62 The work published in this sub-section was published in a similar form as Babović, 1995c
processes and the means to synthesise these into a description of the full hydrological cycle. However, the intensive input-data requirements that correspond to the computational and predictive abilities of this type of model are set at so high level that they are not usually employed for rainfall-runoff modelling. Recently, a case study of a 4955 km² large river basin in India has been reported by Refsgård (1992), in which the Système Hydrologique Européen was used as modelling platform. The discussion on data requirements supports the necessity of using an expensive data, and the author clearly indicates the need for simpler, but still comparably accurate, rainfall-runoff models that can be easier and cheaper to use, over their limited range of validity. Of course, when changes are being introduced into the catchment, an instrument of the SHE type is almost mandatory. Similarly, even in the case of rainfall-runoff models, the treatment of the antecedent conditions may limit the range of validity of the model.

At the same time, these simpler models are not always as accurate as we would like them to be. This statement applies in one way or the other for all of the members of the conceptual, lumped-model family as RORB, NAM, CARP and ARX. Of course, the simplicity of the application of a conceptual model, makes this technique extremely attractive in everyday use, regardless of its obvious drawbacks.

Recently, the pursuit of more accurate but still relatively simple-to-use models has been reinforced by yet another representative, namely that of artificial neural networks (ANNs). The encouraging results obtained by Hall and Minns (1993) have clearly indicated a number of desirable properties that ANNs offer. The continuation of this work (Minns and Hall, 1995) has further proven that ANN indeed represents a valuable contribution in a quest for the rainfall runoff model. Nevertheless, it must be carefully noted that the ANN is a model of a non-symbolic nature, in the sense that it manipulates the data at a level below that of the symbolic. This particularity of the ANN enables it to perform very flexibly and accurately, but at the same time it introduces the most serious (if not only) drawback of the technique - the difficulty to interpret the model’s behaviour and accuracy in a meaningful way. The application of an ANN commonly ends with a statement like: The model works, and it works well, but we do not really know why!

6.11.2.1 Synthesis of rainfall-runoff time series.

Genetic programming offers an alternative which is as accurate as ANN, and possesses a possibly even greater proficiency, and its productions have the possibility to acquire symbolic representations (i.e. algebraic expressions that can be read and interpreted physically). In order to contrast the performances of ANN and GP, exactly the same set of synthetic data has been used as in Minns and Hall (1995). Here, we first briefly recapitulate the characteristics of the Monte Carlo simulations which were used to generate the training data:

a) Rainfall data characteristics

1. normally distributed storms with a duration of 20 h and a standard deviation of 6 h;

2. rainfall depths lognormally-distributed, with a mean of 25 mm and a standard deviation of 2 mm;
3. the shapes of storm profiles are based on those of the UK Flood Studies Report, admitting early-peaked, late-peaked and symmetrical events;

4. the inter-event times were taken as double the previous storm duration minus one hour

Three sequences of 14 storms were generated. The first, with duration of 764 hours, was used for training purposes, and the other two, with a duration of 794 hours, were used for verification.

b) Runoff data generation

A well known conceptual model, RORB (Mein et al., 1974), has been used to produce a discharge series. This corresponds to the routing of rainfalls through a single non-linear reservoir, in which the relationship between storage $S$, and outflow $Q$, is given by:

$$S = KQ^n$$

Three rainfall-runoff relationships were produced using different values of the exponent $m$, in order to cover the range of possible catchment behaviours:

1. $m=0.8$ to represent the typical non-linear relationship encountered in practice;
2. $m=1.0$ to represent the extremely linear case; and
3. $m=0.6$ to represent an extremely non-linear type of behaviour.

In all cases, a hypothetical catchment area of 31.25 km$^2$ was assumed with a main channel length of 8.6 km and no impervious area. For simplicity, no losses were separated-out and the $K$-value was set to 20h. For a fuller description of training and verification data sets, reference may be made to (Minns and Hall, 1995).

For the purposes of this work, we shall concentrate on the differences between the various verification sets applied to induced models. Normal verification would correspond to the set in which ordinates of rainfalls and runoffs are in the same range as in the training set and the rainfalls are generated with a different random seed. Extreme verification, corresponds to the wider range, of the extremes in the validation set. However, out-of-range verification corresponds to the data which are considerably greater in magnitude than those applied for training purposes.

6.11.2.2. Models Induced by Genetic Programming

For each of the three conceptual model outputs, a new run of the GP was performed, each providing a different collection of rainfall-runoff models. In all cases, the training data consisted of a single set of output values - a set of values of discharge. Input data consisted of concurrent and thirteen antecedent rainfall depths, as well as two antecedent sets of flow values.

The performance of the induced models were measured by means of a coefficient of
determination, defined as:

\[ F = 1 - \frac{1}{m} \sum_{i=1}^{m} (q_i - \hat{q}_i)^2 \]

\[ F = 1 - \frac{1}{m-1} \sum_{i=1}^{m} (q_i - \bar{q})^2 \]

where the \( \hat{q}_i \) are the model estimates of the flow ordinates, \( q_i, i=1,2,..., m \), and \( \bar{q} \) is the mean of the \( q_i \). We observe that when antecedent flows are used in the training process, so that the induced expression draws in its turn upon these values, it may be more appropriate to use the difference between the predicted flows and the antecedent evaluation.

We shall shortly present GP induced results, case by case. In all instances, the induced expressions are (surprisingly!) simple and extremely easy to implement as computational models. Although in all cases but one, only the two best-performing expressions will be presented here, the computer implementation of the GP has a built-in capability to store all good expression (i.e. expressions with a performance above a certain threshold). The reasoning behind such a facility is that the system user can decide among the similarly performing expression the one that he or she finds most appropriate as the final solution. And, of course, this decision need not be reached on accuracy achievements alone. Other factors, such as those arising from the sensitivity analysis and the bootstrapping performance can prove to be more significant. However, since the performances of the various expressions are very similar - and for the sake of brevity - we present in most cases only the two best formulae per case study.

6.11.2.3. Linear catchment model

The following expressions were induced by the GP for the linear-case reservoir (m=1.0). We present two, similarly accurate algebraic expressions with similar properties. The coefficients of determination for the GP expressions as well as for ANN performance are summarised in Table 1.

\[ q(t) = q(t-1) + 0.3r(t-1) - 0.29\sqrt{r(t-14)} \]  

(d)

\[ q(t) = q(t-1) + 0.31r(t-1) - 0.27\sqrt{r(t-10)} \]  

(e)
Table 1  Performance overview for four-layer ANN and two best-performing GP expression for the linear catchment model (m=1.0)

<table>
<thead>
<tr>
<th>model</th>
<th>training sequence</th>
<th>verification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>normal</td>
</tr>
<tr>
<td>ANN</td>
<td>0.9993</td>
<td>0.9993</td>
</tr>
<tr>
<td>Expression (d)</td>
<td>0.9987</td>
<td>0.9992</td>
</tr>
<tr>
<td>Expression (e)</td>
<td>0.9981</td>
<td>0.9991</td>
</tr>
</tbody>
</table>

Figure 95 Time series plots for the linear catchment model

Figures 95 and 96 present the performances of the induced models graphically. In the figure 96 time series of the training data, together with calculated time series using expression (d) and (e) are presented. The results are so similar that it is almost impossible to distinguish optically between the three lines in the figure.

We observe in this respect that the numerical coefficients that enter into these expressions must be associated with definitive dimensions if these equations are to be dimensionally homogenous. Following from this again, the nature from this coefficients could also be elucidated using GAs when given enough data, and in that case the search space would be constrained by the dimensionality condition.
Figure 96 Scatter plots (actual against calculated) for GP-induced expressions. Expression (a) is at the top and (b) is on the bottom of the drawing.
6.11.2.4. Typically non-linear catchment model

Similarly to the results for the linear case, the two most accurate expressions are presented here.

\[ q(t) = 0.1052(q(t-1) - 1.2993r(t-12) + 0.032) + q(t-1) - 0.6636r(t-11) - 0.0031 \]  
\[ (f) \]

\[ q(t) = 0.10(q(t-1) - 0.201r(t-10) - 0.0032) + q(t-1) - 0.7214r(t-11) - 0.0032 \]  
\[ (g) \]

Table 2  Performance overview for a four-layer ANN and the two best-performing GP expression for a typically non-linear catchment model (m=0.8)

<table>
<thead>
<tr>
<th>model</th>
<th>training sequence</th>
<th>verification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>normal</td>
</tr>
<tr>
<td>ANN</td>
<td>0.9998</td>
<td>0.9996</td>
</tr>
<tr>
<td>Expression (f)</td>
<td>0.9952</td>
<td>0.9951</td>
</tr>
<tr>
<td>Expression (g)</td>
<td>0.9949</td>
<td>0.9950</td>
</tr>
</tbody>
</table>

6.11.2.5. Extremely non-linear catchment model

Finally, for the extremely non-linear case the best-performing results are presented. Here, in addition to the two most accurate expressions, we include a somewhat less precise, but very intriguing formula, (k).

\[ q(t) = q(t-1) + r(t-2) - r(t-8) \]  
\[ (h) \]

\[ q(t) = q(t-1) + r(t-1) - r(t-8) \]  
\[ (j) \]

\[ q(t) = 2q(t-1) - q(t-2) \]  
\[ (k) \]
Table 3  Performance overview for a four-layer ANN and the two best-performing GP expression for an extremely non-linear catchment model (m=0.5)

<table>
<thead>
<tr>
<th>model</th>
<th>training sequence</th>
<th>verification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>normal</td>
<td>extreme</td>
</tr>
<tr>
<td>ANN</td>
<td>0.9997</td>
<td>0.9836</td>
</tr>
<tr>
<td>Expression (h)</td>
<td>0.9978</td>
<td>0.9973</td>
</tr>
<tr>
<td>Expression (j)</td>
<td>0.9974</td>
<td>0.9975</td>
</tr>
<tr>
<td>Expression (k)</td>
<td>0.9942</td>
<td>0.9944</td>
</tr>
</tbody>
</table>

6.11.2.6. Discussion

One should note that evolution (in nature or anywhere else) does not appear to us as a purposive or directed process. That is, there is no evidence to support the assertion that the goal of evolution is to produce *Homo sapiens*, for example. In effect, the entire process reduces to a competition among various individuals for common resources in a common environment. Some are more successful in this rivalry than others. The better ones are more likely to survive and propagate the genetic information (i.e. knowledge) that they carry. This implies that the 'solutions' we present are not definitive ones. They are only intermediate results, with a certain relatively satisfactory accuracy. In every experiment presented in this article, the GP works through only 500 generations (which roughly corresponds to 20 minutes on an IBM-compatible PC with an Intel 80486 DX processor).

Genetic programming has demonstrated that it is capable of inducing expressions that posses excellent generalisation properties and perform equally well on training and validation data. Of course, since the reported experiments were performed on synthetic data, one might argue that GP achieved nothing else but to induce the conceptual rainfall runoff model used to generate the data. This is certainly true! Real-world data are corrupted with noise and physical processes in nature are much more complex than the simple rainfall-runoff model considered here. Nevertheless, investigations of the performance of the GP on the real data is ongoing activity and some preliminary results do indeed show similar encouraging results.

Clearly, there is no purpose in pressing the accuracy requirements beyond the physically-reasonable, and the quest is more in the direction of developing better tools that will promote physical understanding. Although we have not even attempted to interpret (physically) the GP expressions (d)-(g) here, it is immediately observed that the variables employed in the algebraic formulae have to do with the time lags of the catchment. Although this superficial conclusion might easily be the correct one, it is our feeling that much more real-world data needs to be analysed before serious 'physical' explanations can be advanced. Evidently, the purpose of this work is to introduce a revolutionary new tool for promoting such an analysis, and one that may help us in opening new perspectives on the problems that we are attempting to solve.
In the ANN work (Minns and Hall, 1995), the authors extensively discuss the problem encountered with the normalisation of data, and clearly demonstrate difficulties encountered by ANNs to approximate out-of-range verification sets. The results presented suggest that the GP does not suffer from such deficiency. The reason for such a behaviour is that the ANN learns patterns, whereas the GP induces relations between data. The other reason for the ANNs relatively poor performance in such circumstances could be that it was over-fitted on the training data, therefore decreasing its generalisation properties.

In short, a new evolutionary method for evolving algebraic models has been presented. This suggests a new research direction, directed towards a better understanding of the laws of nature. This understanding is clearly not restricted to that of rainfall-runoff modelling.

6.12. A machine learning algorithm based on evolutionary principles

The application of expert systems and knowledge-based systems has achieved considerable success in a variety of domains since the mid-1960s. Although impressive and practical, the knowledge-based approach has its own weaknesses and limitations, which make it unsuitable as an adaptive problem-solving approach. Generally speaking, knowledge-based systems cannot acquire knowledge automatically and they cannot perform intelligently beyond the limits of the knowledge with which they are supplied.

In this section, an alternative approach based on genetic algorithms is presented. This approach is capable of learning and generalising symbolic concepts from the data it processes. The technique is introduced using a problem from the control domain, but it is certainly not limited to this area. The technique performs a classification task and can be applied to problems as diverse as fault-diagnosis and weather forecasting. More fundamentally, we have to do here with an algorithm that connects the subsymbolic paradigm to the symbolic paradigm.

6.12.1. Intelligent control

Intelligent systems are increasingly used to automate a wide variety of decision-making tasks in hydroinformatics. Real-time control of sewerage systems is just one of areas that demonstrate and utilise the potentials of such an ‘intelligent’ control (see for example, Babovic, 1991; Amdisen et al, 1994). By an intelligent control system we understand a system that has, among its other functionalities, a capability to reason about operating conditions, a capability to discover possible malfunctions of the system and the possibility to implement actions necessary to avoid further propagation of this faulty behaviour. Moreover, such a system should be capable of learning about ongoing processes and applying the knowledge so acquired in order to control the process in the most appropriate way. The main aim of control in general is to improve the performance of the system under dynamic loading. Performance is expressed primarily through the ability of the controlled system to (a) utilise an existing infrastructure and resources to an optimal degree and (b) to satisfy multiple and often conflicting objectives.

Traditionally, the control of systems has been performed by a human operator from some centralised location. The operator makes decisions that are passed on to the system’s effectors...
212 Evolving Solutions

and the effectors implement these decisions in the controlled system. Our present effort is directed towards automating this decision-making process with the objective of assisting the human operator, rather than of replacing him. Thus, the problem has to be addressed here is that of creating a potentially large knowledge-base that can act as an integral part of a complex intelligent decision support system.

It is desirable that the knowledge stored in a knowledge base is organised in such a way that it can be easily accessed and used. It is also desirable that a knowledge-based system should improve upon its actions autonomously by building up its fund of own experience. One typical situation occurs when a control system that has been running for many years is to be replaced. It would then be most appropriate to use the accumulated knowledge obtained through these years of traditional control practice by organising this knowledge in an appropriate knowledge-representation formalism and starting the system again with this accumulated and appropriately organised knowledge. Although data collection is one of the principal features of most control systems in operation, the associated 'knowledge mining' component is automated, or even considered. Further to this intelligent control system should be able to continue building upon existing knowledge in order to arrive at even more competent control strategies. Thus, through the problem of control, we are brought back to problems of so-called pure artificial intelligence, in that our objective becomes one of creating a virtual machine capable of learning and performing intelligent actions.

6.12.2. An intelligent control system is expensive to build

Formalising knowledge and implementing knowledge bases are major tasks in the construction of large AI systems. The hundreds of rules and thousands of facts required by many of these systems are generally obtained by interviewing experts in the domain of application. Representing expert knowledge as facts or rules (or as expressions in any other knowledge-representation formalism) is a typically tedious and time-consuming process. This difficulty has often been referred to as the knowledge-acquisition bottleneck. The knowledge acquisition bottleneck refers to an agglomeration of problems related to the extraction and formalisation of expert-knowledge and its mapping onto artificial media. Techniques for automating this knowledge acquisition process constitute a line of major advance in AI technology.

6.12.2.1. KADS

The knowledge acquisition bottleneck can be overcome in three principal ways. Firstly, special editing systems might be developed for use by experts who possess expertise about the problem domain, but who are not necessarily computer programmers or knowledge engineers. This system allows a direct communication with the knowledge bases of AI systems. Almost every large AI system has some kind of graphical user interface that helps in editing knowledge bases. However, recent advances have augmented the pure interaction between computer and knowledge engineer through the introduction of so-called 'interpretation models' (Breuker et al., 1985), a development of which was in turn inspired
by work by Clansey (1983). This resulted in a so-called KADS\textsuperscript{63} methodology which is becoming a \textit{de facto} European standard in knowledge acquisition. The principle can be expressed as one of modelling expertise in several layers: a \textit{domain layer}, where mainly domain-dependent objects and their interrelations are described, and several higher order layers, namely the \textit{control knowledge}, \textit{task knowledge} and \textit{strategic knowledge} layers. KADS methodology offers a full coverage of the entire KBS development life cycle together with a library of generic models situated at the various levels of expertise that can be used in order to make the interpretation of verbal data from experts more effective. For an overview of the state-of-the-art in KADS, reference is made to Schreiber (1993).

\subsection*{6.12.2.2. Natural language processing}

A second possible way of improving knowledge acquisition is based on natural language processing. Using this approach, the process whereby a human expert gives verbal instructions, or even written material with diagrams and other non-textual material to a computer is partly automatised. Although this represents a valuable \textit{promise} for the future, outside of a few AI laboratories there are very few such systems in everyday, practical use.

\subsection*{6.12.2.3. Machine learning}

A third approach to the solution of the knowledge acquisition bottleneck problem leads in the direction of machine learning. Machine learning represents an answer to some obvious shortcomings of the so-called symbolic approach to Artificial Intelligence (see Section 3.2. Intelligence in Machines). \textit{Homo sapiens} and other animals seem to have impressive capacities for learning from experience. It has long been expected that these knowledge acquisition processes could be somewhat eased by automatic learning mechanisms built into AI systems.

There are several varieties of machine learning and they can be roughly divided into \textit{discovery learning} and \textit{generalisation learning}. Discovery learning implies brave leaps into an unknown direction that only later may perhaps prove to be fruitful. It is, to say the least, an \textit{extrapolation} from observations and experience.

It has been claimed that there is only one system in the entire AI world that is capable of performing discovery learning, namely AM, a system developed by Lenat (1982). To describe this in a very few lines, AM has been provided with descriptions of certain basic mathematical concepts from set theory as inputs expressed in LISP, and instructions for constructing unions of sets, intersection of sets, etc. On the basis of this initial knowledge, AM has succeeded in forming many useful mathematical concepts outside of the domain of expertise with which it was originally supplied. The strategy adapted in AM was that of

\begin{footnote}
\textsuperscript{63} There is no strict explanation of the acronym of KADS. Initially it stood for Knowledge Acquisition for Decision Support, and later it altered into Knowledge Acquisition and Design Support. The two are most often used interchangeably, and most of the researchers indeed find resort to referring simply to KADS (whatever that may mean)!
\end{footnote}
altering the code describing underlying ‘interesting’ functions written in LISP in order to generate new, even more ‘interesting’ functions. Since the connection between mathematics and LISP is very close, AM actually succeeded in many cases in constructing non-trivial relations, and alterations of fragments of LISP code indeed resulted in new, interesting concepts. Later, Lenat’s efforts (1983) towards development of a code - EURISKO - that would learn in arbitrary domains was, by comparison a failure. As Ginsberg (1993) points out:

"There is something in common between [EURISKO’s fleet management game] and the original performance of AM - in both cases, the program needed to exploit a loophole of some sort in order to be successful. In EURISKO’s case, the loophole was in the rules of the fleet game. In AM’s case, it was the close connection between LISP and mathematics. It appears that discovery learning that avoids these loopholes will rest on much deeper understanding being investigated. Alternatively, one can argue that all discovery learning - including that performed by humans - works by exploiting loopholes of one sort or another. This is not a debate I wish to consider here."

Exploration and discovery of similar ‘loopholes’ in various knowledge domains in order to establish discovery learning systems currently appears as an unsurmountable task. Our present ambitions are thus restricted to generalisation learning. Generalisation learning uses existing observations to establish certain behavioural rules and it can be inductive and deductive.

6.12.3. Why learn?

The principle reasons for employing learning at all is that it can be continuous, it has an ability to cope with brittleness and, in cases when learning is performed on symbolic models, there is a transparency in the decision model. In this last case, the ability to incorporate human knowledge into induced systems plays also an extremely important role.

Continuous learning exemplifies the capability of the control system to adapt to the changes that take place at the level of the controlled system. This can, for example, be linked with the physical arrangement of the system. If, for example, a part of a sewerage system is reconstructed, so that the diameters of the pipes, roughness coefficients, properties of pumping stations, etc. change, a control system should ideally be capable of adapting itself to these, newly arisen conditions. More classical control strategies would have to be reconstructed.

---

64 EURISKO was, among other application domains, used to learn to play a naval fleet management game. According to the rules of this game, each player is given certain, fixed resources to develop a fleet, arm it and attempt to overcome the other players. EURISKO even won the American national tournament in this field! The strategy of EURISKO was to develop very small and fast boats in large quantities, but without any armament. These small boats would then be capable to retreating very rapidly, while suffering almost no casualties, and indeed in many cases no casualties at all. Since EURISKO’s fleet had no arms it could cause no damage to the opponent fleet either. According to the rules of this game, the winner is the one that destroys the opposing fleet entirely. Thus, EURISKO’s strategy guaranteed at least a draw, since its own fleet could never be destroyed. It was, however, only by using this particular strategy that EURISKO could win the national tournament.
entirely in order to accommodate the changes of only a part of the controlled system.

It has often been said that symbolic models of intelligence (i.e. knowledge-based AI systems) are brittle in the sense that as soon as they are confronted with the question on which they do not have an explicitly coded answer, their performance degrades suddenly (i.e. a brittle failure). On the other hand, some other, sub-symbolic models of intelligence, like artificial neural networks (ANNs), degrade gracefully. This means, that while the exact answers are not provided, the intelligent system is somehow capable of extrapolating knowledge that is encoded and providing an answer that may not necessarily be absolutely correct, but is still often accurate enough. Learning provides a possible enhancement of symbolic models of intelligence, so that in the cases in which an explicit answer may not exist in the knowledge-base it may well be created by applying a learning algorithm, thus reducing the brittleness.

When decisions are taken, reassurance with respect to the competence of the decision-making procedure is vital. The ability to indicate exact conditions and the reasoning behind a particular decision is often required. As computer assisted decision-making is still relatively uncommon in water resources management, most human operators do not entirely trust machine-generated decisions and require a comprehension of the decision process in a form that they can understand, and thus, preferably in a symbolic format. Because of this feature, some decision makers remain suspicious of the use of neural networks and other 'ultimate black box' techniques for making decisions which cannot be readily interpreted.

It is also important to have an understanding of the reasoning process in order to leave open the possibility to improve the model. For example, if a decision-making module ceases to operate correctly for some reason, it can only be corrected if the reasoning processes internal to the control module are well understood. Thus, in the case of ANNs, the only operation available to introduce a correction is to train an entirely new network. In the case of learning using symbolic structures, due to the transparent appearance of the module, correction of the model's performance may be obtained rather easily by a human operator. Moreover, the models that have a symbolic form can be enhanced through the incorporation of human knowledge.

We shall here consider a machine-learning approach that addresses most of the above considerations (and especially those appertaining to continuous learning, the ability to cope with brittleness, the transparency of the decision model and the ability to revise models and enhance them through the incorporation of human knowledge).

6.12.4. Generalisation learning algorithms

Because discovery learning relies on a wealth of deep knowledge, most of the AI community works with what it calls 'generalisation learning'. All approaches to this kind of learning have a similar flavour, in that the program is induced using a set of training instances and the output is expected to be a routine capable of classifying subsequent instances. In addition to learning classifier systems (Section 6.8), several learning paradigms have been proposed, such as Probably Approximately Correct (PAC) learning (Valiant, 1984), Version Spaces (Mitchell, 1982), Neural Networks (Rosenblatt, 1962; Minsky and Papert, 1969), ID3 (Quinlan, 1986; 1987), and Explanation Base Learning (Laird et al., 1987; Minton et al. 1989).
The model learning or discovery process is a useful technique in almost every industry, (e.g. finance, manufacturing and distributing) whenever on-line databases exist and important predictions have to be made on a regular basis about new cases before they enter the database. Not surprisingly, there are many different fields that address this problem as one of their central concerns. In artificial intelligence this is referred to as a ‘classification’ or ‘induction’ problem. Techniques include tree- and rule-learning algorithms. In statistics this same problem is referred to as a ‘discrimination’ problem, and common techniques are the linear models used in the financial and banking world for credit assessment. In pattern recognition, reference is more commonly made to ‘supervised learning’. In neural network studies, one speaks of a classification and ‘generalisation’ problem and this is routinely investigated using a number of ANN architectures. These diverse fields are really, however, all studying the same problem: learning to predict.

Learning algorithms are currently quite popular in the AI world. Indeed, artificial neural networks (ANNs), probably represent the most popular learning algorithm in and outside AI. Indeed, in hydroinformatics, there has been quite an interest in the assessment of the performance of ANNs (e.g. Hall and Minns, 1993; Minns and Hall, 1995; Massood, 1995).

In more recent times, so-called symbolic empirical learning (SEL) has been a rather active research area (see for example Michalski and Kodratoff, 1990). This approach to machine learning addresses the process of creation and alteration of symbolic descriptions whose structure is known a priori. The most commonly addressed problem is that of concept generation using a collection of examples. This, supervised learning is then concerned with the creation of production rules (classifiers) that are capable of ‘explaining’ the concepts in the training set. It is rather obvious why SEL is a preferred approach towards machine learning in that part of the AI-community that is orientated towards expert systems and knowledge bases, for in this, symbolic approach, knowledge can concurrently be treated as a combination of data to be acquired and a code to be executed. The present-day, standard, data-driven systems, like OPS5 (Forgy, 1981), consist of a collection of production rules whose precondition parts monitor internal and external messages continuously and in parallel and on this basis ‘decide’ about the activation of a particular rule and the broadcasting of an internal message. Thus, the knowledge in such a system resides in a form that is transparent to the human operator.

A common inference task - and it is the one that we employ here as a test-bed - is that in which we learn to make a discrete prediction about some entity given other details about the entity. In monitoring a water resource system, measurements such as discharge rates, water levels and pumping rates are continuously recorded and screened in order to decide whether the system is operating normally or not. If the system operates abnormally, one might want to identify the type of abnormality present. This identification task is the basic task of many knowledge-based systems. This problem is in more technical vocabulary referred to as one of classification, in that the objective is to classify each of the cases on the basis of some readily accessible attributes. These attributes in turn form a so-called feature space.
One common technique for developing a system that perform this classification is to examine historically collected instances. Assume that hindsight tells us which is the correct classification for each case in the database. On the basis of this database a suitable technique may be used to form rules - classifiers - that would perform the classification task on cases entering the database. The entire process is schematised in Figure 98. The process requires two main forms of input: an operator, preferably a domain expert who is capable of giving advise about the nature of the problem, helping in the configuration of the system, of checking the appropriateness of the classifiers, and other such tasks, and a database of correctly classified cases that is used as an experiential foundation for the learning process.

In principle, two different SEL techniques can be distinguished: decision-tree induction and rule induction. Decision trees are an extremely powerful technique for representing complex logical relationships. Constructing decision trees by hand can be extremely time consuming. The techniques for decision tree induction require the most appropriate shape of tree to be identified beforehand. This occurs when the domain expertise of the human operator is used to, for example, create trees of failure modes and the effects of these on the system operation, as based on knowledge and experience. In this, top-down process, a shape of a tree is initially identified and later further refined in order to induce the entire decision tree. There have recently been several substantial developments in decision-tree induction (Quinlan, 1986, 1987; Brieman et al., 1984), but, this is not a technique that we shall consider here.

Rule induction, which is our immediate concern in this place, is orientated towards the process of learning how to classify combinations of variables into logical sets and expressing the classifier so generated in terms of rules of the 'if... then...' format. The inductive process establishes general relationships from the more specific base of examples. Probably the best-known algorithm for rule induction is AQ (Michalski, et al, 1986). Induction of rules is an
‘off-line’ process, which can then be used in operational systems, as schematised in Figure 98.

![Diagram of on-line and off-line systems](image)

**Figure 98** Learning is depicted as off-line, whereas application of a system as an on-line process

Rule induction is more of a ‘bottom-up’ approach. Here, firstly a collection of rules are identified that describe sets of specific examples. All of these rules collectively form a knowledge base. The rule-induction quite clearly offers advantages in the cases in which it is difficult to assume the most appropriate shape of the tree beforehand.

6.12.5. An evolutionary learning algorithm

One approach to searching for patterns in data is to attempt to formulate a ‘complete’ model in the sense that for any collection of input variables \( \bar{x} \), there will be, as output, the best guess for \( y \). Here, however, a different approach is followed. The main driving force is to detect patterns in the data, or stated slightly differently, to detect inter-relationships among particular configurations of independent variables \( \bar{x} \) and certain dependent variable values \( y \). It will become clear shortly that evolutionary pressure will drive the evolving entities towards configurations in which the probabilities relating \( \bar{x} \) and \( y \) are as high as possible. Thus, the induced collection of rules may not be complete - there may not be an appropriate relationship for a given input. The model in its current form then searches for the *pockets of predictability* (Packard, 1990) in the data.

The present approach makes no assumptions about the data upon which it is trained. The training data may be incomplete, in the sense that they do not cover the entire operational range. No particular consistency is assumed for the data. There may be two samples with exactly the same collection of independent variables \( \bar{x} \) that point to two different dependent variable values \( y \). Thus, it makes no sense to induce a ‘complete’ model from ‘incomplete’ data. Our present efforts are attempts towards the induction of rules that have clear and strong relationships, so that a hypothesis can be postulated, with some reasonably high probability, whenever conditions take a particular form. The closure of the induced model is nevertheless possible. This possibility will be discussed later.

Consider data that are obtained from the operation of a system which we wish to control. The
system may be characterised using a large number of variables that can be highly coupled. Such data originating from real-world domains are usually infected with a high degree of observational errors originating from imprecision of measuring equipment and low-dimensional chaotic dynamics, and sometimes limitations on the amount of data available.

If we recall now the system identification problem (sub-section 6.9.1, page 184), and restrict the values of the independent variables \( \bar{x} \), and the dependent variable \( y \), to only a finite sets of values \( \bar{x} \in \{1, 2, ..., D_x\} \) and \( y \in \{1, 2, ..., D_y\} \), where in principle \( D_x \neq D_y \) and \( D_x \) and \( D_y \) are thus numbers of states, the independent variables \( \bar{x} \) may take the appearance of factors that are distributed in space, or they may appear as the time lagged values of the same variables (e.g. rainfall intensities). For the present moment, the algorithm we use is restricted to finite values. Transformation of continuous variables to a collection of finite states may be performed using well described clustering techniques (e.g. Hartigan, 1975), or fuzzy logic (Zadeh, 1965). Quite obviously, this technique lends itself naturally to processing ‘qualitative variables’.

### 6.12.5.1. Hypothesis induction through conditional probabilities

In the continuation we follow rather closely Packard (1990) and his description of hypothesis presentation and its induction using conditional probabilities. The task we attempt to solve is that of discovering consistent subsets within the space of independent variables \( \bar{x} \) and of identifying which hypothesis \( y \) these subsets most probably imply. These subsets are referred to as conditional sets. One approach to formulating a hypothesis \( C \) through independent variables is:

\[
C = \{ x_i = c_1 \land x_{i2} = c_2 \land ... \land x_{im} = c_m \}
\]

where the \( c_i \) with \( i = 1, 2, ..., m \), are constants. This condition specifies a set of points, \( X_C \):

\[
X_C = \{ \bar{x} \mid x_{i1} = c_1, x_{i2} = c_2, ..., x_{im} = c_m \}
\]

In this case \( X_C \) can be interpreted as a hyperplane in the \( n \)-dimensional space of independent variables, and the dimension of the hyperplane itself is \( m \), the number of conditions.

In order to enrich the syntax of this simple formal language, a logical operator OR may be introduced, so that the description of a hypothesis \( C \) may be re-formulated as a conjunction of disjunctions:

\[
C = \{ (x_{i1} = c_1 \lor x_{i1} = c_1) \land x_{i2} = c_2 \land ... \land x_{im} = c_m \}
\]

This is equivalent to the concept descriptions presented in VL (which is simplified version of Variable Valued Logic System - Michalski et al., 1986) which themselves provide a widely accepted language to represent input events for any program operating in an attribute-based space.
A description of a concept \( C \) is a disjunction of complexes

\[
\chi_1 \lor \chi_2 \lor \ldots \lor \chi_m \Rightarrow C
\]

where each complex \( \chi_i \) is expressed as a conjunction of complexes, each complex being a triplet \( \{\text{attribute, relation, set_of_values}\} \), e.g. \( J = R \).

As has been suggested by Packard (1990), each individual coordinate \( \{x_i = c_i \lor x_i = c_2 \land \ldots\} \) can be regarded as an elementary proposition \( \chi_i \), so that the hypothesis \( C \) may be written in a more compact way as

\[
C = \chi_1 \land \chi_2 \land \ldots \land \chi_m
\]

A more general hypothesis may be written by taking the disjunction \( \lor \) of conditions:

\[
C = C_1 \lor C_2 \lor \ldots \lor C_p
\]

\[
= (\chi_{i_1} \land \ldots \land \chi_{i_1}^1) \lor (\chi_{i_2} \land \ldots \land \chi_{i_2}^2) \lor \ldots \lor ((\chi_{i_p}^1) \land \ldots \land (\chi_{i_p}^p))
\]

which corresponds to the union of corresponding conditional sets

\[
X_C = X_{C_1} \cup X_{C_2} \cup \ldots \cup X_{C_p}
\]

Since

\[
NOT(x_i = c) \iff x_i = c_i \lor x_i = c_2 \lor \ldots \ (\text{for all } c \neq c)
\]

because of the finite number of possible values of \( x_i \), the propositions for \( NOT \chi \) \( (\neg \chi) \) are contained within the set of possible propositions. In this case, the proposition above is said to be in disjunctive normal form. Following standard mathematical logic (Enderton, 1972), it becomes clear that each possible Boolean expression involving the elementary propositions is tautologically equivalent to the one in disjunctive normal form. This proves that this language, although possessing such a simple grammar, is rich enough to cover the space of all possible hypotheses.

Given an hypothesis \( C \) which is specified through the independent variables, the question becomes one of assessing the quality of the relationship between the independent variables \( \bar{x} \) and \( y \). Basically, an empirical estimate must be constructed for the conditional probability distribution of \( y \), given \( \bar{x} \in X_C \):
where $N_C$ is the cardinality of the conditional set $X_C$. The function $\delta(y-y')$ counts a number of 'hits'; thus $\delta(y-y')=1$ when $y=y'$, and $\delta(y-y')=0$ otherwise. Summation is performed over the entire conditional set. This probability $P_C(y)$ is a measure of correctness of a certain hypothesis $C$ given the conditional set $X_C$, and is here used to form the basis of a fitness expression.

6.12.5.2. The learning algorithm

The problem is to construct a system that 'learns concepts', i.e. one that determines decision rules that account for all positive and no negative examples. The ideal system should be able to predict a classification of previously unseen examples, or suggest classifications of partially specified descriptions. This is an NP-complete problem, and, as indicated earlier, one effective approach to solving it is to use a randomised search algorithm.

The learning algorithm that is used here is a version of the genetic algorithm described previously in Section 6.5. Moreover, we use a genetic algorithm that manipulates chromosomes of variable length (sub-section 6.5.7.4) and employs a syntax-preserving crossover. This situation, however, may be equalised with the fixed-length chromosomes, where the chromosome will code for $m$ genes, $m$ being a number of independent variables. Each gene can take a value # (do not care symbol), or one or two literal values. In the cases when two (or even more) literal values occupy the same locus, these coordinates should be joined using the disjunction operator. For example,

$$(2,\#,\#, (8,6), \#, \#) \Leftrightarrow x_1=2 \land (x_4=8 \lor x_4=6)$$

The complete expression for fitness must also include a term that introduces the notion that $y$ is only an estimate, based only on finite number of sample points. A term proportional to $1/N_C$ is then introduced to devalue fitnesses in the conditional sets of very few points. Packard (1990) in his original text discusses the dependency of the probability distribution on best possible \textit{a priori} guesses. In the absence of any \textit{a priori} knowledge, a uniform probability distribution may be used, or, as Packard suggests, a probability distribution for $y$ with no other conditions. For more detailed descriptions and derivations of fitness expression, reference is made to the original text of Packard. Taking these into considerations, a more general fitness expression then becomes:

$$F(C) = \sum_y P(C) \left[ \log \frac{P_C(y)}{P_0(y)} \frac{\alpha}{N_C} \right]$$

where $\alpha$ is a parameter that determines a degree of dependency on the size of $N_C$. The fitness
expression thus emphasises the well-determinedness of \( y \) (expressed though \( \sum P_c(y) \log P_c(y) \)). Other terms that enhance the fitness of simple logical conditions may be formed by adding penalty terms that are proportional to the number of coordinates with a condition.

### 6.12.6. The ‘four-reservoir problem’

The ‘four-reservoir problem’ appears to have been originated by Larson (1968), who had obtained solutions using Linear programming (LP) and State Incremental Dynamic Programming (SIDP). Heidari et al. (1971) applied Discrete Differential Dynamic Programming (DDDP), and more recently Esat and Hall (1994) applied genetic algorithms (GA) to solve this problem. We shall now adopt the same problem as a test-bed for our rule induction technique based on evolutionary principles.

The physical system, consisting of four reservoirs (see Figure 99) supplies water for both irrigation and power generation. The state variables are the storages in the reservoirs, \( s_i \), \( i = 1, 2, 3, 4 \), expressed in standardised units. During operation, the inflows to the systems are kept constant, \( q_1 = 2 \) and \( q_2 = 3 \). The decision variables are the discharges \( u_i \) from the reservoirs \( s_i \) at discrete time steps \( k \). In the original problem, there were 12 time steps, which combined with 4 reservoirs implied 48 individual decisions to be taken during one problem cycle.

The volumes of the reservoirs are limited, so that:

\[
0 \leq s_1 \leq 10 \\
0 \leq s_2 \leq 10 \\
0 \leq s_3 \leq 10 \\
0 \leq s_4 \leq 15
\]

The discharges from the reservoirs are constrained as well, such that:

\[
0 \leq u_1 \leq 3 \\
0 \leq u_2 \leq 4 \\
0 \leq u_3 \leq 4 \\
0 \leq u_4 \leq 7
\]

The equations describing the system of four reservoirs are equations of mass continuity, formulated for each operating time step, \( k \), as follows:
\[ s_1(k+1) = s_1(k) - u_1(k) + q_1 \]
\[ s_2(k+1) = s_2(k) - u_2(k) + q_2 \]
\[ s_3(k+1) = s_3(k) - u_3(k) + u_2(k) \]
\[ s_4(k+1) = s_4(k) - u_4(k) + u_3(k) + u_1(k) \]

Figure 99 Schematisation of the four-reservoir problem

In this particular optimisation problem, the aim is to maximise the combination of the benefits obtained through power generation at the reservoirs and the benefits from irrigation that are obtained through the release of water from reservoir 4. The benefits are formulated as linear functions of the flow (i.e. \( b_i u_i \)). There is also a specified requirement for the volume that the reservoirs should contain at the end of the simulation period namely that \( s_1 = s_2 = s_3 = 5 \) and \( s_4 = 7 \). Should these volume not be retained, a penalty function, \( g_i \) is established. If the reservoirs store more water than required, no additional benefit is obtained. The values of appropriate benefit coefficients, \( b_i \), \( i = 1, 2, 3, 4, 5 \), and penalty functions \( g_i \) can be obtained from Heidairi et al. (1971), and are not reproduced here. The optimisation problem then becomes one of maximising an objective function of the form:
$I = \sum_{i=1}^{4} \sum_{k=1}^{12} b_i(k)u_i(k) + \sum_{k=1}^{12} b_5(k)u_4(k) + \sum_{i=1}^{4} g[s_i(12), d_i]$

Our present objective, however, is not to optimise this expression, but rather to use the results of optimisation as a training set, on the basis of which the rule induction algorithm will generate operation rules that are capable of achieving the same objective. Thus, we simply use the results, as summarised in Figure 100, and employ these as a basis for inducing relationships.

<table>
<thead>
<tr>
<th>k</th>
<th>$s_1(k)$</th>
<th>$s_2(k)$</th>
<th>$s_3(k)$</th>
<th>$s_4(k)$</th>
<th>$u_1(k)$</th>
<th>$u_2(k)$</th>
<th>$u_3(k)$</th>
<th>$u_4(k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 100 Optimum policy*

Training was performed in four consecutive runs for every reservoir independently. For each particular run, a separate table of data was formed, so that, for example, for reservoir 1, the rules governing $u_i(k+1)$ were formed by algorithms processing $u_i(t), u_1(t), u_2(t), u_3(t), s_1(t), s_2(t), s_3(t)$. Thus, the induced rules are supposed to determine the operation of the reservoir on the basis of the status of the entire system at the previous time step.

6.12.7. Induced relationships

Perhaps surprisingly, the genetic algorithm performs the search for the fittest conditional distributions extremely efficiently, and with very little effort. It must be emphasised, however, that in this particular application of genetic algorithms we are interested in the entire evolving
population, rather than only the fittest individual - there is obviously more than a single
operational rule per reservoir. Of course, certain distributions will be more 'fit' than others
(with the corresponding rules), and this is the criterion that indicates the probability of a
particular hypothesis holding, given the certain values of the feature variables. The word
probability is deliberately used here, since there is no guarantee that the most probable
hypothesis will imply the correct hypothesis. This is due to the presence of indifference
symbols in the rules' pre-condition parts. Thus, there may be situations when more than a
single rule may qualify to be applied. This conflict must be resolved by applying a certain
meta-control. In this case meta-control is resolved by the degree of determination of
individual rules. Clearly, rules with more literals should be applied before rules with a lower
determination.

Once the hypotheses are induced, they can be applied in a variety of ways. In principle, the
induced system will be presented with a collection of inputs on the basis of which the induced
system will make hypothesis \( y \). The rules induced through the application of the genetic
algorithm for each of the four reservoirs are summarised in Figures 101, 102, 103 and 104.

| IF \( S_3(t) = 3 \) \( S_4(t) = 0 \) \( U_1(t) = 3 \) \( U_3(t) = 4 \) \( U_4(t) = 7 \) THEN \( U_2(t+1) = 3 \) | IF \( U_2(t) = 2 \) \( U_6(t) = 7 \) THEN \( U_2(t+1) = 2 \) |
| IF \( U_3(t) = 0 \) THEN \( U_2(t+1) = 0 \) |
| \( U_1(t) = 3 \) \( U_3(t) = 4 \) \( U_4(t) = 7 \) THEN \( U_2(t+1) = 3 \) |

_Figure 101 Operational rules for reservoir 1_
IF $U_i(t)=3$  
$S_i(t)=0$  
THEN $U_i(t+1)=4$

IF $S_i(t)=5$  
$U_i(t)=1$  
THEN $U_i(t+1)=2$

IF $U_j(t)=0$  
$U_j(t)=4$  
THEN $U_j(t+1)=4$

IF $S_j(t)=7$  
$U_j(t)=0$  
THEN $U_j(t+1)=2$

IF $U_k(t)=0$  
$U_k(t)=4$  
$U_k(t)=7$  
THEN $U_k(t+1)=3$

Figure 102 Operational rules for reservoir 2

IF $U_i(t)=2$  
$U_i(t)=4$  
$U_i(t)=7$  
THEN $U_i(t+1)=4$

IF $U_i(t)=4$  
THEN $U_i(t+1)=0$

IF $U_j(t)=0$  
THEN $U_j(t+1)=0$

IF $U_j(t)=4$  
THEN $U_j(t+1)=4$

Figure 103 Operational rules for reservoir 3

IF $S_i(t)=7$  
$U_i(t)=0$  
THEN $U_i(t+1)=7$

IF $S_i(t)=5$  
THEN $U_i(t+1)=2$

IF $S_i(t)=6$  
$U_i(t)=0$  
THEN $U_i(t+1)=7$

IF $S_i(t)=5$  
$S_i(t)=6$  
THEN $U_i(t+1)=0$

IF $S_i(t)=4$  
$S_i(t)=4$  
THEN $U_i(t+1)=0$

Figure 104 Operational rules for reservoir 4
Let us reiterate the direction in which the genetic algorithm evolves. The probability $P^-$ is the most dominant component in the fitness expression. It drives the population towards the configurations (hypotheses) that will maximise the number of 'hits', while minimising the cardinality of the conditional set $N_c$, that implies this hypothesis. Should, for example, the pre-condition part of a particular hypothesis be composed of indifference symbols only, the cardinality of the conditional set $N_c$ would be equal to the number of all samples in the training tables - the maximum possible number. This affects fitness adversely. Evolution thus drives towards the most probable distributions. All other, less probable, possibilities are defined as exceptions to the most probable hypothesis, and these exceptions are coded using more literals, and correspondingly less indifference symbols.

The induced rules that are presented in Figures 101, 102, 103, and 104 cover the entire range of operation for the four-reservoir problem. Thus the induced knowledge base is 'complete'. Such a result is not a consequence of luck or a well-chosen (or a toy) problem. It is a consequence of employing a fitness-sharing mechanism (see sub-section 6.5.7.2.). The application of fitness-sharing guarantees the evolution of diverse, good solutions. In the present case, this property was sufficient to provide a 'complete' model.

It may indeed be argued that the four-reservoir problem itself is a small one and incomplete, in a sense that only a limited part of the operational range was used as training data (e.g. $q_1=\text{const.}=2$ and $q_2=\text{const.}=3$). In cases in which more complete data are used in training, one should test the coverage of this training set by the means of rules, and should this coverage not be complete, one should add (either manually or by applying some learning mechanism again) a few rules that cover these exceptions. What we really want to emphasize is that, after running for some time, the genetic algorithm identifies conditions that lead to a fit of conditional distributions. The question of usefulness of the discovered patterns remains open, however. The induced patterns may be extremely fit, but the real question is whether the data used are statistically significant. 'The genetic algorithm is mining the data, and is very effective in finding statistical flukes' (Packard, 1990).

Figure 105 demonstrates a possible implementation of the four reservoir problem in a commercial knowledge-based system. In this case, implementation has been done in Neuron

---

65 Packard (1990) offers an additional consideration that may be used in order to address the problem of getting trapped in incomplete data:

"This question may be answered by comparing the fitness learned from the data with the distribution of fitnesses of conditions learned from a distribution of test data generated with a particular null hypothesis. For example, the null hypothesis is commonly no correlation between the data samples, i.e. a distribution that is independent and identically distributed (IID). Particular realisations from this distribution are easily created by taking the original data and shuffling it well, destroying correlations while maintaining the original distribution. For each of many realisations, the learning algorithm may be run, and a distribution of 'fitness of the fittest condition learned' may be formed. The fitness learned from the original data must then be compared to this distribution. If it is near the mean (measured in units of standard deviation of the distribution), the learned pattern is not very significant. If it is far from the mean, it may be counted as significant. Unfortunately, this procedure is very time consuming."
Data's object-oriented KBS-system, Smart Elements. As indicated, each reservoir is modelled as an individual object, and each of these objects relies on a part of the induced knowledge base that relates to its operation. This implementation is indeed transparent and that its alteration and expansion is now quite an easy task.

Figure 105 A part of the hypothesis generation process for the reservoir 1

The symbolic model of intelligence has received a lot of criticism earlier in this thesis. The rule induction technique presented here is, however, essentially a sub-symbolic model of intelligence. The knowledge level (sensu Newell) appearance of the evolving phenotypes is a residue of a process of representation and interpretation of genotypes in the problem domain. In this way, standard rule-based systems acquire somewhat more prominent qualities which can now be summarised.

Rule-based reasoning using simple condition-action rules is the basic structural component of the knowledge-based system architecture. Rule-based systems have an advantage in modularity (meaning that each rule or instruction is more or less independent of its location in the rule set), and the rules can specify transitions in the environment (in order to make predictions and to specify actions).

Many observations and conclusions about the various worlds (of hydroinformatics) are not based on Boolean logic. Therefore the ability to manipulate uncertain values is an important capability of an intelligent system. Many techniques have been proposed, including Bayesian
calculus (Babovic, 1990), the introduction of ‘certainty factors’, fuzzy logic (Amdisen, 1992) and various bidding systems. Learning systems support reasoning with uncertainty in several ways. Firstly, the probability of each individual rule functions similarly to the certainty factors in other rule-based systems. Secondly, all classifiers are forced to compete with one another and the most fit are those that find the best balance between the evidential support derived from the input stimulus and their appropriateness to the current situation.

Domain-specific knowledge that controls general inference mechanisms can greatly improve the efficiency and effectiveness of knowledge-based systems. Humans have their own strategies to solve a particular problem and these may not be easily captured explicitly by context-free rule-based representations. Learning mechanisms that are based on evolutionary principles use empirical credit assignment to assemble symbolic structures that discover these domain-dependent strategies by utilising feedback from the problem domain.

6.12.8. Some other potential applications

Rule induction has been successfully applied in a wide range of areas ranging from medicine to financial market forecasting. In engineering, and in hydroinformatics in particular, the use of rule induction has been rather modest. Here, we would like to identify some other areas where the use of rule induction can be advantageous.

Fault diagnosis and condition monitoring is probably the most obvious application area where the rules for identification of malfunctioning components might be identified. Condition monitoring is a key element for planned maintenance. For example, it may be beneficial to monitor the performance of components of the physical system as they degrade, and thence to order their replacement or repair before a major fault takes place. In this way, a sort of quality control of the processes is established, since the performance of the system’s components and the system in total are maintained within certain limits that guarantee product quality (e.g. Abbott et al., 1983). The four-reservoir application described in this chapter falls in the knowledge elicitation area. Many knowledge-based systems have failed because it has not been possible to acquire and represent the vast complexity of the knowledge that they need in order to function in a reasonable time. Even in many successful expert systems it has taken several years for the rule base to reach a usable size and structural and logical consistency. Human experts usually find it easier to explain their decisions through examples. Here, rule induction can be extremely useful by inducing rules from examples so that an expert can access their validity. Should the expert produce examples that do not fit the induced rule set, then these can be added to the example set and new rules induced. This iterative process can proceed until the expert is satisfied with the rules and their coverage of his knowledge, or until the rule set stabilises. This use of rule induction can be put to use in establishing complex logical behavioural relationships.

There are situations in which it is necessary to reduce the load on the ‘human in the control loop’. At the same time, it is extremely difficult and sometimes even impossible to produce efficient control algorithms using classical control techniques, whereas humans can learn to control these processes without much trouble. In order to reduce the operator work load at critical stages of control one may employ decisions induced by a machine. This process involves recording actions of a skilled operator in a response to the information presented to
him concerning the system states. The information representing the system states is treated as the attributes and the control action as the classes and the data subjected to the rule induction. The resulting rules provide a representation of the skilled human control actions which can then be used to control the system. Extracting control rules from humans in some automated fashion is only one example of determining the behaviour of the system. Recording the behaviour of any system in response to its stimuli and subjecting the recording to inductive inference does establish a set of rules governing the system. This can be useful in domains like weather forecasting and hydro-meteorological forecasting in general.

6.12.9. Emergent intelligence

The results of the experiments performed using a machine learning code that utilises evolutionary principles points towards a much more general issue. One of the principal objectives of this thesis is to demonstrate the potential of a multi-agent approach to computation. These agents are situated in an environment that consists of both fellow agents and a description of a problem domain. It is the interaction of an individual agent with its environment that results in behaviours that might be of interest in our engineering and control applications.

It is this interaction that is of paramount importance. The interaction with the environment feeds back to an (evolving) agent information on the degree of success of its behaviour in an environment. This is a form of credit assignment. Indeed it is an empirical credit assignment. The essence of the process is in the interactional-experimental nature of the mechanism. Through this process of 'experimentation' performed by applying various genetic alteration operators, the pertinence of the agents' components is established.

<table>
<thead>
<tr>
<th>Property</th>
<th>Emergent Intelligence</th>
<th>Artificial Intelligence</th>
</tr>
</thead>
<tbody>
<tr>
<td>task-specific knowledge</td>
<td>separate from problem solver</td>
<td>integrated into problem solver</td>
</tr>
<tr>
<td>operators</td>
<td>representation-specific</td>
<td>task specific</td>
</tr>
<tr>
<td>processing</td>
<td>local and opportunistic</td>
<td>centralised and global</td>
</tr>
<tr>
<td>knowledge-representation</td>
<td>emergent</td>
<td>explicit</td>
</tr>
<tr>
<td>knowledge content</td>
<td>dynamic and adaptive</td>
<td>static</td>
</tr>
<tr>
<td>credit assignment</td>
<td>representation-specific, empirical and dynamic</td>
<td>task-specific and static</td>
</tr>
</tbody>
</table>

*Figure 106 Recapitalisation of relations between Artificial Intelligence and Emergent Intelligence (adapted from Angeline, 1993)*

In our present case, the induced system corresponds to Newell's knowledge level (1982). However, since the underlying genetic algorithm has no problem-solving knowledge specific to the domain (the four-reservoir problem in this case) and certainly not at Newell's knowledge level, it acquires this knowledge through interaction with its environment and
through being assigned a credit based on its empirical performance. Thus, an intelligent system performing at a knowledge level *emerges* with a knowledge that is not available to the problem-solver in any explicit form. Thus, an evolutionary search accumulates information about the search space by continuously exchanging information among evolving agents and receiving information about its successes and failures from the search space in the form of fitness. This information effectively gets incorporated into new trials, and so the process continues. This is the essence of the process that Angeline (1993) refers to as *Emergent Intelligence* (see Figure 106).

Angeline outlines how emergent intelligent systems encode a general approach to problem solving by relying on representation-specific knowledge, rather than task-specific knowledge. Task-specific knowledge implies definitive references to the problem environment, whereas representation-specific knowledge refers only to the structure of an evolving agent. In certain cases it is more appropriate to code for an agent’s genotype using binary strings, and yet in another it is more appropriate to employ RPN trees. Everything beyond this representation-knowledge then emerges through interaction of the agent with its environment.

### 6.13. Model Induction from Data

Now in the further development of science, we want more than just a formula. First we have an observation, then we have numbers that we measure, then we have a law which summarises all the numbers. But the real *glory* of science is that we can find a way of thinking such that the law is evident.

*(Feynman et al., 1963, p. 26)*

Inferring models from data is an activity of deducing closed-form explanation based solely on observations. These observations, however, represent always and in principle only a *limited source of information*. The question emerges how this, only limited flow of information from a physical system to the observer, can result in a formation of a model that is complete in a sense that it can account for *all* the ranges of phenomena within the physical system in question - and to describe even the data that are outside the range of previously encountered observations.

In this chapter we outline two, at first sight rather different, but essentially closely comparable techniques to induce two models of different appearances. In the first instance, a now already familiar genetic programming, and in the second instance we use an artificial neural network. Moreover, in the case of the ANN, we apply a genetic algorithm to generate the most appropriate topology and to determine the values along the interconnecting weights.

The present effort is characterised by a search for model that is capable of acquiring semantics from syntax. Clearly, every assumed model has its own syntax. The question is whether such a syntax can capture the semantics of a system that it attempts to model. Certain classes of model syntax may be inappropriate as a representation of a physical system. As Crutchfield (1992) points out, one may choose the model the representation of which is complete in the sense that a sufficiently large model can capture the data’s properties to a
degree of error that decreases with an increase in the model size. Thus, one may decide to expand in Taylor of Fourier series to a degree that will decrease the error to a certain, arbitrarily given degree.

This study, though, attempts to construct models from a limited syntax and only then observe their generalisation properties. The artificial neural networks are constructed using relatively simple computational devices - corresponding to the imposition of a very constrained syntax. Genetic programming as well applies to only a small class of functions, and these may not be complete, nor even appropriate for a given domain. Thus, this approach is qualitatively different. It assumes a restricted syntax, but aims at a complete semantics.

At this moment there are no clear indications as to which of the two modelling techniques will perform best. Actually, we do not direct our efforts towards a more accurate model. The appropriateness of the model’s performance should not be expressed in these terms only. Other qualities, such as model’s expressiveness, its transparency and its generalisation properties should play equally important roles.

One more, extremely important process must be underlined here. This is the process of what we call the consolidation of semantic content. The present study utilises a numerical model to a ‘replace’ physical reality. Every model is a collection of signs that serves as a sign (Abbott, 1992). This model consolidates the semantics of a physical world into a collection of output numbers - which is to say, in semiotic terms, that it provides a transformation of sign vehicles. These sign vehicles are then further transformed in yet another level of consolidation into a GP-induced expression, or trained artificial neural network.

### 6.13.1. Additional Resistance to Flow Induced by Flexible Vegetation

Recent floods of the Mississippi river in 1993 and 1994, the rivers Rhine and Maas in 1994 and 1995, the Russian river in Napa Valley in California in early 1995, the Missouri in the spring of 1995, and many other such instances have raised the question concerning the water management of large rivers in Europe and North America to the level of public consciousness. To manage such large bodies of water, and in particular when the managing agency finds itself at the downstream end of these bodies, so that there is almost no means to manage them (i.e. flow of Rhine through the Netherlands) presents almost impossible problem.

Traditionally, protection from floods has been undertaken by the means of dikes. It is a common knowledge among hydraulic engineers that the introduction of dikes, by constricting the cross-sectional area, forcing the same body of water to pass through a smaller space, resulting in higher discharge rates, velocities and water levels, which all in turn demands higher and stronger dikes. Apart from creating a tremendous maintenance problem, there is a constant ‘political’ pressure to make the land near dikes available for industrial and recreational use, which exacerbates the already extremely difficult water-management problem, with very expensive flooding risks as a consequence. The traditional, ‘dike-based’ approach management was seriously put into question in the USA in the case of Mississippi and almost resulted in a major catastrophe in the Netherlands in the February 1995.

Based on these bitter experiences, many pressure groups have promoted the restoration of
natural wetlands that would act as natural ‘sponges’ capable of absorbing access water and thus reducing flooding risks. Of course, the dikes would remain; they have become a necessity. But many of them should now provide the means to control the timing and rate of flooding lands, rather than alternatively, to prevent this flooding altogether. Thus, the degree of risk that they pose be considerably reduced. The Dutch Ministry of Public Works is, for example, currently pursuing several of those wetland restorations projects. The Government of California has also embarked on a major wetland protection and restoration task as well.

This present section is not directed to dike operation and water-management problems however. It is concerned more with the underlying formulation of physical processes that should form the basis of correct managerial decisions. The wetland restoration projects favour the growth of reeds and other similar vegetation within river basin. The presence of vegetation influences the flow conditions, and in particular the bed resistance, to a large degree. However, the influence of the rigid and flexible vegetation on a flow conditions are not very well understood. Some laboratory experiments using physical scale modelling have been performed (Tsujimoto et al, 1993; Larsen et al, 1990), but only over a limit range and with variable success. Similarly, although field experiments are continuing, the data available remains scarce.

More recently, a numerical model has been developed with the intention of deepening the understanding of the underlying processes (Kutija and Hong, 1995). This model is a one-dimensional vertical model based on the equations of conservation of momentum in the horizontal direction. This numerical model is employed here as an experimental apparatus in the sense that this, fully deterministic (even if highly parameterised) model, is used as a source of data that are then further processed by two apparently different methodologies in order to induce a more compact model of the additional bed resistance induced by vegetation.

6.13.1.1. Short description of the Kutija-Hong model

As indicated earlier, the Kutija-Hong model is based on a differential equation of the conservation of motion

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + w \frac{\partial u}{\partial z} + \frac{1}{\rho} \frac{\partial P}{\partial x} - g i - \frac{1}{\rho} \frac{\partial \tau}{\partial z} + \frac{F_x}{\rho \Delta x \Delta z} = 0$$

where

- $u$ horizontal velocity
- $w$ vertical velocity
- $P$ pressure
- $\rho$ density of water
- $g$ gravity acceleration
- $i$ bottom slope
- $\tau$ shear stress
and the term

\[ \frac{F_x}{\rho \Delta x \Delta z} \]

represents the additional specific drag forces.

The model is developed in order to address the effects of flexible vegetation in steady flow conditions only, so the convective momentum terms and pressure term are neglected. The numerical model is based on a finite difference implicit approximation for the unknown velocities at the discretisation points (see Figure 107).

![Figure 107 Schematisation of domain for numerical simulation](image)

The model takes into account the effects of shear stresses at the bed and the additional forces induced by flow through vegetation and it possesses a facility to adapt initial distributions of velocities over discretisation points on the vegetation stems. In addition to representing these influences, the model takes into account the bending of the reeds under the loads produced by the drag forces. Due to bending, the effective reed height is reduced, which implies a
lesser drag, and this in turn implies a reduction in the drag induced deflections of the vegetation. These processes are resolved through an iterative numerical process.

For a more detailed description and a discussion of the specifics of the model, see Kutija and Hong (1995). Before continuing with description of how the results obtained from this model are used, it is first necessary to review some features of artificial neural networks and to relate these to the possibilities opened by genetic programming.

6.13.2. Artificial Neural Networks

Simulations of interconnected networks of simple threshold-based processing units called artificial neurons have been an object of scientific investigation ever since the earliest days of the computer (McCulloch and Pitts, 1943; Hebb, 1949). These neurons are in fact a special form of general automaton (Hopcroft and Ullman, 1979) that in fact form a conceptual basis for any computer programming language. The McCulloch-Pitts neuron is depicted in Figure 108.

![Figure 108 Schematisation of a McCulloch-Pitts neuron](image)

Rosenblatt (1958, 1962) proposed, following Hebb (1949), an algorithm referred to as a perceptron learning rule that adapts weights in a neuron (or, in Rosenblatt's terminology perceptron) in order to perform a certain, pre-defined task. Minsky and Papert (1969) showed, however, that a single perceptron, although a promising technique, is not capable of solving linearly non-separable problems, like exclusive or (XOR) problems.
The breakthrough and re-formation of so-called *connectionism* occurred only through the research directions that utilised collections of simple computational units with a set of weighted interconnections (Hopfield, 1982; Ackley *et al.*, 1985; Rummelhart and McClelland, 1986). The repeated and parallel execution of these simple numerical operations at each and every neuron *emerges* in a useful behaviour of a network as a whole. The performance on the network depends strongly on its architecture, or topology of connectivity among individual neurons. As in the simple perceptron, the weights are adjusted during the training process.

![Schematisation of a multi-layer, feed forward neural network](image)

*Figure 109 Schematisation of a multi-layer, feed forward neural network*

A typical network of interconnected neurons (providing an artificial neural network - ANN) is shown in Figure 109. It receives input signals in the form of numbers. These numbers propagate through network, where they are amplified using weights along the connections. Every individual neuron receives as input the weighted signals from the neighbours with which it is connected. These input signals are then summed at every individual node and the individual output signals are calculated using:

$$\frac{1}{1+e^{-\sum x_i \cdot w_i}}$$
where $\Sigma x_i$ is the sum of all input signals that the node receives and $\theta$ is the threshold value associated with the node. It has been shown that an artificial neural network, given a sufficiently long training time and a sufficient rich topology, can act as universal function approximator (Hornik, 1989).

The network without any loops or multiple connections between individual neurons is called a feed-forward; otherwise it is said to be recurrent.

6.13.2.1. Training of Artificial Neural Networks

Following Barto (1990) two forms of learning within the ANN paradigm can be identified: parametric and structural. Parametric learning implies learning only the weights within the neural network, whereas structural implies a learning of an appropriate connectivity within the ANN.

There are many, more recent, training techniques, all of which are intended to search more effectively for the most appropriate collection of weights. They vary from a form of simulated annealing (Ackley, Hinton and Sejnowski, 1985) to a generalised learning rule, called a delta learning rule (Rumelhart and McClelland, 1986).

The 'standard' structural learning algorithms are based on simple heuristics and statistical indicators of individual nodes. Roughly speaking, one can distinguish between an incremental approach, in which simple network is enriched with additional nodes and connections in order to meet certain accuracy requirements (see, for example, Ash, 1989; Fahlman, 1991), and decremental methods, in which large networks are pruned to arrive at simpler, yet accurate network topologies (see for example, Mozer and Smolensky, 1989; Hassibi and Stork, 1993).

The principal approach adopted in these structural learning methods are nevertheless rather constrained. On the one side, the direction in which the network topology evolves is fixed and monotonic, and thus either incrementally either decrementally. On the other side, all of these methods employ a form of macro operator, or collection of certain elementary transformation to the network (i.e. add another fully-connected hidden layer). These operators, although useful, and in many cases based on useful heuristics, still restrict the syntax of all possible modifications that could take place.

6.13.2.2. Training of Artificial Neural Networks by Evolutionary Means

Various combinations of evolutionary algorithms and artificial neural networks have been attempted. Again, several flavours may be easily distinguished. Some authors address only parametric learning by means of evolutionary algorithms (Montana and Davis, 1989; Collins and Jefferson, 1991). Some authors allow for structural learning using evolutionary algorithms, while parametric learning is performed through the application of more standard, ANN-native training techniques (i.e. back propagation) in order to establish weight values (Miller, Todd and Hedge, 1989; McInrney and Schraudolf, 1992). Similar studies, attempt to co-evolve a network topology and induce weights (Das and Whitley, 1992; Angeline, 1993).

In addition to these, a somewhat different approach to simultaneous parametric and structural
learning was proposed, initially by Kitano (1990) but improved later by Gruau (1994), based on a cellular encoding of neural networks. This utilises grammatical descriptions of structure but restricts the weights to either +1 or -1.

The problem of structural learning is a difficult and often deceptive one. At the root of many inconveniences is in epigenetic mapping that relates genotypes and phenotypes. This mapping may be of a many-to-one form. In such cases, two identical phenotypes (ANNs) may have different underlying genetic codings. Under the action of crossover, the resulting offspring may contain redundant components, thus losing the parents' computational abilities. This competing conventions problem (Schaffer, Whitley and Eshelman, 1992) grows exponentially with the number of hidden units.
Another problem encountered with evolutionary induction of neural networks when two networks that are structurally identical, but have different weights are crossed-over. It is well-known that a given topology of connections allows for multiple solutions, each coded for different weights along these connections (Smolensky, 1988). The neural networks are essentially a representation, or more correctly, a distributed representation (Hinton, McClelland and Rumelhart, 1986) and it has been shown that the removal of a small number of hidden nodes does not affect the outcome of calculations as much as the changes of weights along connections. The role of an individual neuron is determined through its interconnecting weights. If we return to the problem of crossover among networks, it now becomes apparent that crossover between two structurally identical, but parametrically different, networks may result in rather poor offspring performances.

Our present interest is orientated towards simultaneous structural and parametric learning using evolutionary algorithms. From this point of view two principal components within the evolutionary algorithms to neural networks framework must be identified, namely, the mapping between genotypes and phenotypes and the fitness-measure evaluation.

**Mapping**: a relation between an individual's genotype (e.g. binary string) and its phenotypical interpretation (ANN) must be established. This in fact corresponds to the epigenesis \( I \times G \rightarrow \mathcal{P} \), as defined in sub-section 6.1.4. In the present case, the mapping is established in the manner indicated in Figure 111. It should be observed that even input signals are subject to evolutionary processes. Thus, for example, genetic algorithm might detect that there is no need to incorporate a particular input channel in its topological structure.

**Fitness measure**: an evaluation of individual's phenotype performance must be established. Since the main present concern is to minimize the error between a neural network's productions and observations, the main driving force was defined in the usual way as one of minimising the mean squared error (MSE). No restrictions were placed on neural network's topology or size. Thus, in principle, the number of hidden layers and associated neurons can grow indefinitely in order to meet the accuracy criteria. In order to invalidate such a behaviour, a more natural control of an evolving network's size was built into the fitness measure. It is well known that neural networks, and in particular of those of the large size, can easily be overfitted during the training process (see for example, Smith, 1993). As a result,
the network describes the training data too closely and loses its generalisation properties - that is, its ability to describe a wide range of data, and not only those for which was trained. In our particular case, the entire training data set was split into two sub-sets, a training sub-set and a test sub-set. The training is, of course, performed on a training sub-set, and the MSE calculated for this. After training, a validation is performed on a test sub-set and a new MSE calculated. The combination of these two MSEs is then used to establish a fitness measure of each and every individual. In this way, the generalisation properties of ANN became a part of the fitness measure, and they can be promoted equally during the evolutionary process.

6.13.3. A Genetic Programming Approach

The focus here is on applying the genetic programming paradigm to a system identification problem, as introduced earlier in this work (in Section 6.9). We accordingly view an artificial evolutionary process in the light of a structure synthesis, where this structure takes the form of an individual symbolic structure. Here, however, the previously introduced notion of GP is expanded to introduce so-called automatically defined functions - ADFs (Koza, 1994a).

6.13.3.1. The Automatic Definition of Functions

The automatic definition of functions is an extension of GP directed towards the automatic decomposition of an evolving expression. This is to say that the evolutionary process is enriched with another facility that identifies, parts of the evolving expression and defines these as new, automatically-defined functions. These chunks of functions are then not threatened by disruption deriving from future crossover operations, but, on the contrary are preserved from the possible application of genetic operators applied at the level of the expression evolving as a whole. However, ADF do not provide a static structures. Every evolving individual expression has its own ADFs. These ADFs are engaged in their own evolutionary process. This is to say that genetic operators are still applied at the level of ADFs as well, but still only between ADFs. Thus, the evolving individual may be perceived as a simultaneous evolutionary process of several RPN trees in which the main tree stands for the principal function that can call other ADF trees, and then at least one but usually more ADF trees.

GP enriched with ADFs is qualitatively nothing new, but it may be more efficient. First, and above all, with ADFs, GP is capable of evolving more complex programs with less apparent effort. Should the size of an evolving expression increase extensively, the alterations of individual instructions has little effect. With ADFs, the structural complexity (see sub-section 6.9.4.2) is kept under control, and genetic operators can be, on average, more effective. Secondly, since ADFs provide, so to say, evolving sub-routines or even-macro-operators, the evolutionary process is able to take larger jumps across the fitness landscape.

It should be noted that each evolving individual 'breeds' its own ADFs. The isolated ADFs may have no clear meaning, or no 'semantic content' from the point of view of the problem at a hand, and neither from the perspective of a part of the problem, or from the perspective of another evolving expression.

Similar to the ways of working of ADFs, Angeline and Pollack (1993) have proposed a
process of *module acquisition*, that *freezes* and *de-frosts* parts of evolving expressions. Although there are some subtle differences between the two techniques, in principle they are much the same.

Both ADFs and module acquisition are randomised processes. Rosca and Ballard (1994) have however, proposed somewhat more guided processes based on Kolmogorov complexity (Li and Vitanyi, 1993), that serve to compress (or even define, for that matter) the parts of expression that have the highest information content.

### 6.13.4. Results of Experimentations

Returning now to the Kutija-Hong model, used as a generator of data, this model was in effect used as an *emulator*, or a truthful representation of a physical reality, while providing the conveniences of fast calculation and an ability to produce results with any degree of scale refinement. In this way, the numerical model not only replaced physical scale modelling facilities within this exploratory environment, but introduced several intrinsic advantages over scale models. It is well-known that so-called *roughness scaling* is one of the principal difficulties in the development of physical models. Since the roughness is the primary phenomenon in question here, the issue of its physical correctness remained critical. The numerical model is based upon rather well-known and well-described physical phenomena, whereby this ‘open question’ becomes principally one concerning the adequacies of the eddy-viscosity and Newtonian-fluid approximations. All the other elements of the numerical model are well-posed and whenever compared with field and experimental results, appear to produce sound results. Since the ‘realism’ of the complete numerical model was reasonably well-proven against experimental data in the case of shift (non-flexible) (see Kutija and Hong, 1995), one of the main reasons for using this model as a data generator was that large numbers of different experiments could be performed in a short time with virtually no costs involved, a situation which would be quite different in the case of similarly extended scale model test and field models.

The model has been run with a wide ranges of input parameters in order to create training data for both genetic programming and artificial neural networks. As the first attempt towards the development of a model of additional roughness, only the effects of stiff (non-flexible) reeds with high stiffness were simulated.

The results of Kutija-Hong simulations were presented as a dimensionless ratio \( \eta \) of an original Chezy number - that corresponding to an absence of vegetation - and a new Chezy number - that corresponding to a developed vegetation. This ratio \( \eta \) can be conveniently incorporated in the Chezy formula for velocity under steady flow conditions:

\[
    u = \eta \sqrt{C} R i
\]

The additional roughness term \( \eta \) can, in principle take any value in a range \( \eta \in (0, 1) \). However, for \( \eta = 0 \), the resistance to flow becomes infinitely large, thus stopping the water flow, which is, physically a highly unlikely situation. The smallest values of \( \eta \) experienced
Evolving Solutions with the Kutija-Hong numerical model were \( \eta = 0.1 \). For \( \eta = 1 \), the influence of vegetation on the roughness is zero.

Altogether, some 4800 training data sets were generated. The training data consisted, in the first instance, of dimensional numbers formed from: the water depth \( h_w \), the reed height \( h_r \), the reed diameter \( d \), the number of individual reed shoots per square meter \( m \), and a numerical parameter \( p \) related to the eddy-viscosity approximation and its further relation to the vegetated layer height. The Kutija-Hong model was then used to calculate \( \eta \) for input data that were varied in the ranges:

\[
2.5 \leq h_w \leq 4.0 \\
0.25 \leq h_r \leq 2.25 \\
50 \leq m \leq 350 \\
0.001 \leq d \leq 0.004 \\
0.4 \leq p \leq 1.0
\]

The performance of the induced models is expressed through collection of statistical measures of correspondence between training data and model outputs. These measures were those recommended by Diskin and Simon (1977) and were: the Average Error (AE), the Mean Square Error (MSE), the Root Mean Square Error (RMSE), the Maximum Error (ME) and the Coefficient of Efficiency (F).

6.13.4.2. Results obtained using genetic programming

In the case of genetic programming, a coarse distributed panmictic island model of population was used. There were 5 sub-populations, each consisting of 100 members, organised in an hierarchical manner. Four of these were used in an ordinary procedure providing a continuous source of diversity, whereas the fifth was created by copying the five best-performing entities from each of the other four. It thus, in effect represented a reservoir of ‘elite individuals’. The exchange of members among these sub-populations was scheduled at every 10 generations. Although techniques for promoting the parsimony of evolving formulae were employed, a sharp limit of the 50 nodes was still set on the size of evolving entities as well. 10 independent runs were performed in each case over the duration of 250 generations and statistics of the two best-performing expressions were calculated and presented.

a) Dimensional values

In the first attempt, the data were used in their original, dimensional form. Such an approach was adopted in order to introduce the least possible level of ‘pre-suppositions’ in the model evolution. The results obtained were still quite satisfactory. The best performing formula in RPN notation is presented in Figure 112.
\[
\begin{align*}
&\left(\frac{1}{\exp(-d/hr)} \right) \\
&\left(-\left(\frac{r\log\left(\frac{r\log\left(\frac{m}{hw}\right)}{\sqrt{\sqrt{\frac{d}{d}}}}\right)}{\sqrt{\sqrt{\frac{d}{d}}}}\right)\right) \\
&\left(* \left(\frac{hr}{\sqrt{\sqrt{\frac{d}{d}}}}\right)\right) \\
&\left(* \left(\frac{\left(r\log\left(\frac{r\log\left(\frac{m}{hr}\right)}{\exp(-d/hr)}\right)\right)}{\exp(-d/hr)}\right)\right) \\
&\left(* \left(-\left(-d/hr\right)hr\right)-0.00410\right)p}}
\end{align*}
\]

**Figure 112** The best-performing expression, in reverse Polish notation, for dimensional case

The following Figure, 113, illustrates the success of the evolutionary process for all five evolving sub-populations simultaneously over the period of 250 generations. Figure 114 demonstrates the degree of accuracy obtained with this expression in the form of a scatter plot. There is some dispersion for the higher values of \(\eta\), and this causes the statistical measures to take somewhat more modest values. They in fact take the following values:

- Average Error (AE) = 0.0512
- Mean Square Error (MSE) = 0.0410
- Root Mean Square Error (RMSE) = 0.0029
- Maximum Error (ME) = 0.3346
- Coefficient of Efficiency (F) = 0.8774

**Figure 113** Illustration of the success of **Figure 114** Scatter plot for best-performing GP-evolutionary process for all five sub-populations induced expression from Figure 112

**b) Dimensionless values**

Another set of experiments has been performed, but in this case, a collection of dimensionless numbers has been used. The introduction of these dimensionless quantities increases the accuracy to a certain degree and improves the convergence. An explanation for this may be that introduction of dimensionless quantities collapses search space, so that exploration is more effective. The dimensionless ratios introduced were defined as follows:
In addition to these $p$ and $m$, were used without any changes. The best performing expression is presented in Figure 115.

$$h_{rel} = \frac{h_w}{h_r}$$

$$w_d = \frac{h_w}{d}$$

$$r_d = \frac{h_r}{d}$$

$$h_{whrd} = \frac{h_w - h_r}{d}$$

The best performing expression, in reverse Polish notation for dimensionless case

```
(sqrt (sqrt (/ (pow wd hrel) (+ (/ m
   (sqrt (/ m hrel))))
   (/ (/ (/ (/ (/ (/ m hrel)
      (sqrt (/ (sqrt p)
         (/ m hrel))))
         (sqrt (pow hrel hrel)))
         (sqrt p))
         (sqrt p))
         (sqrt (sqrt (sqrt (/ (sqrt p)
            (/ m hrel))))) ))))
```

Figure 115 The best performing expression, in reverse Polish notation for dimensionless case

Statistics for this expression are summarised as follows:

- Average Error (AE) = 0.0525
- Maximum Error (ME) = 0.2662
- Mean Square Error (MSE) = 0.0410
- Coefficient of Efficiency (F) = 0.8824
- Root Mean Square Error (RMSE) = 0.0029

Following figure, 116 is a graphical illustration of accuracy of this expression. Figure 117, on the other side demonstrates how the tree size develops during the evolutionary process. It is clear that tree size grows in initial stages of evolution. However, is 'stabilises' in later stages, so that becomes obvious that the size of expression is not everything.
c) GP search using ADFs

A lot of expectation have been placed upon application of ADFs in this context. However, the results were rather poor. It appears that, with introduction of ADF, the size of the search space increases dramatically, and that ADF did not contribute in any sense to formulation of a better expression. Following the same procedure we presents a best-performing formula in Figure 118 and scatter plot in Figure 119. In addition to these, in Figure 120 a development of mean standardised fitness over one evolutionary run is presented.

\[
\text{MAIN:} \\
\quad \left(- \left( \frac{\log \left( \frac{\log 0.15706}{\log \left( \log \left( \log \left( \log \left( \log \left( \log ADF1 \right) \right) \right) \right) \right) \right)}{-0.28447} \right) \right)
\]

\[
\text{ADF0:} \\
\quad \text{hw}
\]

\[
\text{ADF1:} \\
\quad \left( \frac{\log \left( \frac{-m}{\log \left( \log \left( \log \left( \log \left( \log \left( \log \left( \log \left( \log m \right) \right) \right) \right) \right) \right) \right)}{-0.56867} \right)}{-0.56867} \right)
\]

\[
\left( \frac{\log \left( \frac{-0.56867}{\log \left( \log m \right)} \right)}{-0.56867} \right)
\]

\[\text{Figure 118} \text{ The best-performing expression, in reverse Polish notation, for the case with ADFs}\]
Statistics for the presented expression is:

- Average Error (AE) = 0.1503
- Mean Square Error (MSE) = 0.0982
- Root Mean Square Error (RMSE) = 0.0829
- Maximum Error (ME) = 0.6667
- Coefficient of Efficiency (F) = 0.1377

6.13.4.3. Results obtained using artificial neural networks evolved by genetic algorithm

A similar set of experiments has been performed using artificial neural network. Here, a genetic algorithm has been run for 50 generations, with a population size of 100. No particular differences have been observed between the performance of the artificial neural network in the dimensional and dimensionless cases. Again, this is because the neural network is a form of representation. It learns patterns of data by distributing weights along interconnections. The fundamental question is whether dimensionless numbers define these patterns better than the dimensional. In this case, it appears that this is not the case. Thus, we summarise the further results for evolutionary process on ANNs which make use of dimensional numbers only.
The topology of the best performing network is schematised in Figure 121. It takes all five input fields, has one hidden layer composed of four neurons and its connectivity is as indicated in this Figure. The performance of the network is summarised as follows:

- Average Error (AE) = 0.0512
- Mean Square Error (MSE) = 0.0410
- Root Mean Square Error (RMSE) = 0.0029
- Maximum Error (ME) = 0.3346
- Coefficient of Efficiency (F) = 0.912

It is interesting to note that the second best performing network has a topology characterised by six hidden nodes with almost full connectivity. In this case the coefficient of efficiency takes a value F=0.907 and average error is 0.012. Thus, the complexity, in topological terms, does not contribute much in terms of fitness. In this particular case, the performance of the second-best network was good on its training sub-set but considerably poorer on the test sub-set, thus degrading both its generalisation properties and its fitness value.

A corresponding scatter plot (Figure 122) demonstrates the accuracy of the induced model. It graphically demonstrate the degree of correspondence between the training data and ANN approximation to this.

![Scatter Plot](image)

**Figure 122 Scatter plot showing the desired output values of \( \eta \) as calculated by Kutija-Hong model and the corresponding ANN values**

In addition to providing more intuitive measures of the accuracy of the induced ANN, the three figures that follows (Figures 123, 124 and 125) demonstrate the responses of the neural network to changes in its input parameters. Basically, these figures illustrate the sensitivity of the neural network output to changes in input values. Figure 123 and 125 are surfaces upon which the influences of changes of water depth \( h_w \) and reed height \( h_r \) and reed density \( m \) and reed diameter \( d \), are presented, respectively. Figure 124 depicts the influences of the parameter \( p \) related on the eddy viscosity approximation.
6.13.4.4. Verification data

Two independent verification sets were performed. In the first test, the Kutija-Hong model was used to generate verification data that were in the same range as the training data. This formed the basis of normal verification. Second verification test was perfomed in order to assess the generalisation properties of induced relationships. Here, the data were generated that produced results outside the range of the training data. The results are summarised as follows.

Table 4 Summary of performances for GP-induced relationships in dimensional, dimensionless and ADF cases, as well as ANN performance

<table>
<thead>
<tr>
<th>Model</th>
<th>Normal verification</th>
<th>Out of range verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensional case</td>
<td>0.872</td>
<td>0.867</td>
</tr>
<tr>
<td>Dimensionless case</td>
<td>0.879</td>
<td>0.874</td>
</tr>
<tr>
<td>ADF</td>
<td>0.137</td>
<td>0.133</td>
</tr>
<tr>
<td>ANN</td>
<td>0.906</td>
<td>0.845</td>
</tr>
</tbody>
</table>

6.13.5. Genetic Programming versus Artificial Neural Networks

But, are these two techniques really all that different? It might be claimed, of course, that the ANN is of a sub-symbolic nature, whereas GP is of a symbolic nature. This is not really true, however. Both of the techniques are sub-symbolic in the sense that a manipulation of data occurs at a level which is below that of the symbol. The tokens which are manipulated have at best indicative signs, and have in and by themselves no expressive capability, so that they are not even expressive signs in the standard sense of Husserl (1900/1913//1970). GP manipulates tree structures that only acquire any meaning at all (and thus any semantic) once they are tested in a problem domain through the application of the already familiar notion of empirical credit assignment. Similarly, the ANN propagates errors through a series of simple summation and threshold devices (conveniently described as neurons) without any necessity to relate to any particular domain. The fact that GP appears at a higher level of abstraction is only a consequence of a particular interpretation of the RPN tree by fitness functions as explained earlier in this work within the context of general machine learning algorithm based
on evolutionary principles (Section 6.12).

At the cost of repetition, it must be emphasised that the difference between the two techniques resides essentially in the fact that a neural network is a form of representation and genetic programming is a form of search algorithm. The neural network learns how to represent by altering its weights, whereas GP searches for an appropriate tree structure. It has been proven that ANN is universal function approximator (Hornik et al., 1989) provided enough hidden layers and units are made available. Since GP evolves trees that are composed of terminal nodes and functions (generic and custom), it can be equally strongly claimed that, given the appropriate collection of functions, GP is a universal function approximator. Only in this respect, GP and ANNs are equivalent.

Evolutionary algorithms (EAs) and ANN are two similar, and yet basically different modelling techniques, and they have correspondingly much to offer to one another. It cannot be simply claimed that the one is better than the other. The choice of which one to use depends, at the very least, on a particular application. Should a robust, general, low-level representation is needed, then the ANN is probably the right answer. Should a RPN tree with a symbolic appearance in a multi-modal environment be the main requirement, then GP appears to provide definitive answer. Given this situation, however, the general lesson of the theory of semiotics as applied in hydroinformatics (Abbott, 1993) shows that the best results are most likely to be obtained when these techniques are combined, so as to work together in an efficacious way.

6.13.6. Symbolic and Sub-symbolic

Genetic programming and connectionist models are both sub-symbolic models (of intelligence). The primitive components constituting emerging system have a meaning only within the context of the rest of the system. That is, a single branch of an RPN tree, or a computational model of a neuron in the same sense, has very little meaning-as-such, and indeed it may in principle not be interpretable without the remainder of the system. Once again, in the nomenclature of Husserl, it is necessarily an indicative sign but not necessarily an expressive sign.

Intelligent behaviour in such a sub-symbolic paradigm arises only through a complex amalgam of low-level interactions and is a collective property of the totality of the system. The global behaviour of the system is therefore intimately connected with the functioning of underlying components. In contrast, the Physical Symbol System Hypothesis (Newell, 1980) asserts that the essence of intelligence lies in the logic of a symbol processing system and that the details of how such systems are implemented are not important. There is an ongoing a

\[This\ downplays\ some\ of\ the\ statement\ from\ the\ introduction\ to\ this\ section.\ It\ has\ been\ claimed\ in\ the\ introduction\ that\ ANN\ has\ a\ non-complete\ syntax.\ However,\ the\ statement\ that\ ANN\ are\ universal\ approximators\ -\ given\ enough\ hidden\ layers\ and\ hidden\ units\ and\ long\ enough\ training\ time,\ basically\ asserts\ that\ ANN\ syntax\ is\ fundamentally\ complete\ -\ thus\ it\ can\ approximate\ virtually\ anything.\ This,\ however,\ does\ not\ mean\ that\ our\ introductory\ statement\ was\ entirely\ wrong.\ In\ all\ practical\ cases\ one\ operates\ with\ ANNs\ of\ only\ limited\ size,\ thus\ their\ practical\ syntax\ is\ limited\ as\ well.\]
debate concerning which of these two models is to be preferred when attempting to model intelligent behaviour. It appears, however, that all sides will reach a consensus that both views make an important contribution and that, in a full accord with observations of semiotics generally, a proper account of intelligent processes must take both into account (Belew and Forrest, 1988).

Connectionism assumes that intelligence arises from the interaction of a large number of simple numerical computing units - abstract neurons - where interconnections between neurons contain or encapsulate knowledge that originates from the data and which is extracted by the application of a certain learning rule. One possible implication of this is that the cognitive processes can be depicted as process of association as instance of pattern matching. Connectionism is a paradigm that is symbol- and rule-free, in which knowledge about the domain is extracted directly from the domain through the application of, in some sense always, statistical method, such as that realised through backpropagation of error.
7. Emerging Solutions

Ever since mankind came to self-consciousness in the Biblical Fall, it has been concerned with its environment. In the first instance, the concern was directed towards the cropping of the world's natural resources in order to secure survival. Little by little, the allotment of these assets became limited, and the further survival of mankind jeopardised. Nowadays, almost everyone is talking about the environment and its protection: so-called 'green' movements, governmental institutions, house-keeping ladies, local farmers ... indeed almost everyone.

Everyone expects answers from scientists and engineers to such questions as: "What will happen if we construct this structure here?" The politician asks: "Will my voters be satisfied with the results? Shall I be re-elected, if I support this venture within the government?" Many persons are interested in the immediate consequences of such interventions and how these may change or influence their surrounding environment and their commodious living. Everyone demands correct prognoses.

Thus such concerns over the fragility of the Earth's resources resulting from centuries of human-related activities have captured the attention of environmental policy- and decision-makers, as never before. Correspondingly, the most serious problems challenging researchers, decision-makers and educators during this current period are the preservation, management,
conservation and restoration of the Earth’s diminishing resources. To address these problems, more advanced environmental monitoring, assessment and simulation systems need to be developed that can aid in making more informed policy decisions about these resources. This observation is echoed by Lubchenko et al (1991) who states that:

"Citizens, policy-makers, resource-managers and leaders of industry all need to make informed decisions concerning these resources. Ecological knowledge is one critical facet of the information required for making complex environmental decisions. Ecological understanding and knowledge are urgently needed to detect and monitor environmental changes, to evaluate the consequences of a wide range of human activities, and to plan for the management of sustainable ecological systems. ...Ecological understanding of complex phenomena is essential if society is to anticipate and ameliorate the environmental effects of human activities."

While this may seem in many ways an insurmountable task, it is still one of the central concerns of many agencies of national, provincial and local governments to predict temporal changes of the environment. Examples of this concern can be seen in the efforts currently being made to test global warming hypotheses, or to examine the impacts of pollution on sensitive ecosystems. A research-orientated society involved with these projects is aware that the goal of prediction is a long stretch from current technology. Jørgensen et al. (1992) express the utmost concern in this respect:

"The twentieth century is almost over. In it a whole new era of man-nature relationships was ushered in following the Industrial Revolution. All present indicators point to a continuance of current trends toward a deteriorating capacity of the planet to sustain a high standard of civilised human life. As ecologists, and ecological modellers, we three authors have asked ourselves whether our science today is really developing the prerequisite knowledge to allow humanity to keep pace with increasingly complex requirements of existence in a rapidly changing environmental future. Our answer is unanimous. We think that present-day ecology, including modelling, is failing to produce an adequate science of large-scale, unbounded and interconnected complexity that exists as a fundamental property in the world's ecosystems. These systems cannot be reduced to component mechanisms without losing the essence of their holism. The challenge of irreducible complexity and breaking the riddle of wholeness cannot be met by the mechanistic reductionism of traditional science, not even carried to computer-assisted modelling where mechanisms inferred as important between aggregated variables are translated non-uniquely into mathematical formulations."

Thus, the question is: can we really make correct prophecies? Do we really know that much about our environment? Do we really understand underlying processes so that we are capable of anticipating future phenomena? If we judge on the basis of the current constellation of the sciences, as contrasted with the rate at which humanity is exploiting natural resources, it will be very difficult to claim this.

In particular it is the trends in the deterioration in natural conditions that are the most alarming. Obviously, there are many reasons for such a state of affairs. We can direct our
blame at an ever increasing industrialisation, world-wide, as a main source of pollution, for example. Or, we can direct our attention to increasingly intensive kinds of food production and soil treatment. However, as professional modellers and scientists, we cannot ethically avoid responsibility for our current inability to predict many of the consequences of our 'management' of the environment (World Bank, 1992, 1993). Current modelling techniques employed in ecological modelling especially fail to reproduce the interconnected complex of relations that exists as a fundamental property of real-world ecosystems.

Perhaps this statement may appear as too strong and even unfair. Biology, chemistry, physics and all other, so-called 'natural sciences' have accumulated a vast amount of knowledge and data and has armed the practitioner with such a level of erudition and such a range of technologies that he might be supposed to be capable of answering the challenges of preserving the homeostasis of Mother Nature. But still we are facing new threats and are witnesses that the support that is in principle provided by science is not being utilised in the best possible way.

Although ecological modelling has become more sophisticated with the use of coupled models and super computers, more complex and qualitatively different models need to be developed in order to evaluate, monitor and simulate the functioning of ecosystems more thoroughly. While ecosystems are composed of a large number of simultaneous processes, they also introduce emergent processes resulting from such interactions. It is the idea of such emergent properties that Pattee (1973) and Grobstein (1973) identified as critical issues if biotic systems are to be modelled.

The simulation of an ecosystem using computer modelling requires the design of a modelling environment which tends to emulate a natural system in its essence. Such a modelling paradigm should attempt to answer the questions to which current ecological modelling practice cannot answer fully. Some of those questions are posed by Lubchenko et al. (1991) as follows:

"How do morphological, physiological and behavioural traits of organisms interact? What factors explain the life history adaptations of organisms? What are population-level consequences of these adaptations? What are the feedbacks between biotic and abiotic portions of the ecosystems and landscapes? How do climatic, anthropogenic, and bionic processes regulate bio-geochemical processes?"

Before we concentrate on the description of a novel approach to ecological studies, let us introduce some modelling formalism, in order to clarify the target goals that any model has to fulfil. According to Casti (1988), a model is a mathematical representation of the modeller's reality, a way of capturing some aspects of a given reality within the framework of a mathematical apparatus that provides us with a means for exploring the properties of that reality mirrored in the model.

According to the British physicist Maxwell (see also Casti, 1988, p. 1): "The success of any physical investigation depends upon the judicious selection of what is to be observed as of primary importance." In the continuation, Casti paraphrases Maxwell's declaration as 'the success of any modelling venture depends upon a judicious choice of observables and means
for encapsulating these observables within the framework of a convenient formal mathematical (and logical) apparatus." And indeed, our capacity to perceive explicitly determines our capacity for description and model-making. Thus, the conceptualisation of a natural system is in the first instance based on the specification of the observables appertaining to such a system and a designation of the manner in which they are linked.

In this chapter a new idea in the area of ecological modelling is introduced. This new concept even has a particular charm for in it we suppose that every form of life demonstrates intelligent behaviour, and that an ecosystem is the consequence of the interactions of different forms of life exercising their individual intelligent behaviours. This is unequivocally in line with other, more classical descriptions of ecology, such as (in the Concise Oxford Dictionary): "Ecology is branch of biology dealing with organisms' relations to one another and to their surroundings; (human) ecology, study of interaction of persons with their environment".

In order to describe the behaviour of the individual life-forms, we are utilising an already defined artifice of an 'intelligent agent'. The intelligent agent here represents a way of modelling a life form. Thus, the intelligent agent is a model, which means an encapsulation of some slice of the real world in terms of signs, using our available knowledge about this portion of the world within the confines of the relationships constituting the universe of discourse.

While this approach certainly opens avenues for future research applications, from the present perspective two key questions arise that should guide the development of an approach for natural system simulations. Firstly, how do patterns and processes at one spatial and temporal resolution affect those at other scales of resolution? Secondly, how can multiple interactive models be linked within a dynamic spatial modelling system? What is required is a framework that will allow properly structured questions to be posed and adequately addressed.

This work constitutes a basic framework for the development of a new modelling paradigm that will attempt to find answers to at least a few of the posed questions. This venture is initiated with a most fundamental notion of synthesising contributions from various scientific disciplines (e.g. biology, ecology, computer science and hydraulics) in order to create unique modelling paradigms. The ultimate objective of the study is a better understanding of the aquatic environment. One can observe the natural system at different levels of abstraction and describe the interrelationships among the elements of the universe of discourse in a more profound or a more superficial way as required. We trust that the uniqueness of this proposed modelling approach will enable the research community to gain more insight into the functioning of natural aquatic systems, so that it will consequently make a better management of these natural resources possible.
7.1. Current Approaches to Ecological Modelling

"The question of what the major current problems of Biology are cannot be answered, for I do not know of a single biological discipline that does not have major unresolved problems... Still, the most burning and as yet most intractable problems are those that involve complex systems"

Ernst Mayr, 1982

The late 1960s saw a major breakthrough in the field of hydraulics as it became possible to solve complex, non-linear conservation equations for fluid flow through the use of numerical techniques and computational machinery. The ability to predict the changes in water levels and discharges caused by a given structure or a given operating policy was sufficient - and indeed provided a great improvement in the reliability of engineering studies.

Today, the solutions of the hydrodynamics and transport-dispersion equations are almost standard and recent advances in the modelling of sediment transport and ecology have taken over as the most interesting developments on the numerical modelling side. Indeed, in many studies, hydrodynamic conditions are only of indirect interest. The engineer is rather asked to determine, for example, how the prevailing current and waves affect the spreading of sewerage discharges and thence to determine the impact on the ecosystem. A good knowledge and accurate modelling of the hydrodynamics is essential, but in practice this is just the 'carrier' or 'platform' for the study of other subjects that are the ones of real interest.

In conventional biological models, one usually takes as a starting point some, more or less well described phenomenon, and tries to find a representation that replicates this phenomenon. The most widely used ecological models, which are said to be of the Lotka-Volterra type, after the names of their originators, perfectly reflects this ideology. In such an approach, the description of an entire ecosystem is reduced to the characterisation of variations of the biomass of the lumped species components. The variations in these quantities are then expressed in terms of parametrically defined rates of biomass exchange between these components and their similarly defined rates of energy consumption. Time delays may be incorporated to account for such features as hatching and breeding cycles, but otherwise the descriptive system reduces to one of sets of simultaneous ordinary differential equations, expressed in the most simple form as (Maynard-Smith, 1973):

\[
\frac{dx}{dt} = ax - bx^2 - cxy \\
\frac{dy}{dt} = ey + c'xy
\]

where:
- \( x \) denotes the prey biomass
y denotes the predator biomass
a, b, c, c' and e are calibration parameters.

The above equations are the most simple variants of mathematical models aimed at describing species interaction, namely that of prey-predator relationships. There naturally exist more refined descriptions which take into account age structure, functional response of predators, breeding seasons, etc., but they still hold to this basic biomass form of description. This modelling paradigm is only outlined here with the purpose of displaying its underlying assumptions; for example the density of a species per unit area can be adequately represented by a single numerical variable; changes in its density can be adequately described by deterministic relations and the effects of interactions between species are instantaneous even if these effects are lagged in time, and so on. But, on the other hand, none of these assumptions holds in nature!

As pointed out by Abbott (1991), a major reason for this position is ideological in that we regard the environment as being entirely "for us", and we forget the interests of the other creatures that share our planet and to whom we have a direct responsibility as the stewards of the creation.

The present Lotka-Volterra models are usually mounted as superstructures upon hydrodynamic (HD) modelling platforms and coupled in parallel with advection-diffusion (AD) superstructures. The HD platform then provides a field of velocities and water depths, the AD platform advects and diffuses all the species (including salt, heat and chemical substances of interest) and the ecological model regularly allows these various species to interact. Although this approach has been found satisfactory for many applications involving lower-trophic-level organisms, with good agreement often being observed with measurements over extended time and space scales, it is often found to be considerably less than satisfactory when describing the higher trophic levels.

The divergence between computed and physically-observed results can be attributed to many factors. At the higher trophic levels it is the individual organism that makes a decision at every instant, and this decision is in turn based on the organism's internal state and its immediate, perceived environment.

There are two major problems that emerge through application of the broad class of Lotka-Volterra models as exemplified in the above equations to real-world simulations. Firstly, the accuracy of the solution to these equations clearly depend on the calibration parameters. Since 'behavioural' similarity to the natural system is comprehended by the equations themselves, it is the calibration parameters that are supposed to 'capture' the correspondence between the model and the real-world. However, these calibration parameters are not well-defined in nature. One even may ask what is the 'physical' meaning of these parameters. How well are they 'grounded', and indeed are they 'grounded' at all? A similar question had been posed earlier in this work, but so far in the context of modelling human intelligence. It has been argued that the symbolic approach cannot not be accepted as appropriate, simply because of the tremendous problem of 'symbol grounding'. An analogous situation is observed here. Calibration parameters might have certain meaning to us (as in the Punch cartoon reproduced on page 44 as Figure 14), or we might be able to read a certain meaning into them (so as to
provide a 'physical interpretation': this is what the classical sciences were all about), but they do not exist-as-such and are thus 'disconnected' in a certain fundamental way from the world that they are supposed to model.

On the more pragmatic - but closely related - side, the problem of providing calibrating parameters is an apparently overwhelming one. This is partly due to the genetic-induced variability obtaining at the level of individuals, the variability occurring within communities, the difficulties of providing functional group definitions and of capturing the processes of adaptation and evolution. Despite these, quite obvious and serious inconsistencies, in practice these parameters must be calibrated using some real-world measurements in such a way that the output of the model approaches as close to the measurements as possible. Should the situation in the environment change, the model must be calibrated again to establish new values of these calibration parameters. This brief example demonstrates the principle deficiency of such an output-orientated modelling as a predictive medium. Since one of the main goals of the modelling activity is to capture the behavioural essence of the natural system in the model, so that the model of the system can be used for predictive purposes, the drawbacks of the output-orientated approach to modelling become rather obvious.

Secondly, the output orientated approach to modelling is an essentially top-down one. As has been pointed out previously, the behaviour of natural ecosystems in fact emerges from a local interactions of the agents that constitute the ecosystem. It is extremely difficult (and in many cases even impossible) to describe such an intricate network of relationships through the language of ordinary or partial differential equations. Moreover, it is unnatural to superimpose a model from the top, when in practice it spontaneously materialises along bottom-up lines. Even in physics, where behavioural aspects of individual agents constituting a global system play a less important role, there are very few laws where statistical averaging of micro phenomena are valid at the macro-level of the system's behaviour. Navier-Stokes equations and (other) ideal gas descriptions are two of the very few examples where simplified formulations based on the statistical averaging of micro-phenomena can have a practical range of validity at the system level, and even then we are obliged to introduce new concepts, such as 'viscosity' and 'heat conduction' in order to realise even this.

Even as society demands an ever higher level of predictive capability in ecological modelling, the existing Lotka-Volterra-type models continue to demonstrate inadequacies in the description of the behaviour of the higher trophic levels. These inadequacies spring essentially from an ideology that denies the complexity of the mental world of the individuals at the higher trophic levels. The approach presented here is given over to introducing these mental worlds, but initially only in a (predicate-) logical way through use of existing knowledge-based-system (KBS) shells.

There is a certain irony in this respect, in the fact that engineers and scientists, once the arch-exploiters of the earth's natural resources, now present themselves as the environment's best hope for survival. Over the last decade, developing the technology to protect, rather then to plunder, the environmental larder has become of crucial significance.
7.2. Intelligent Agent Approach to Modelling

"If - as I believe - physics and chemistry are conceptually inadequate as a theoretical framework for biology, it is because they lack the concept of function, and hence that of organisation... Perhaps, therefore, we should give the ... computer scientists more of a say in the formulation of Theoretical Biology"

Christopher Longuet-Higgins, 1969

In many, if not most, cases the divergence between computed and physically observed results can be attributed to an insufficient account being taken of the behavioural processes that guide the behaviours of the higher-trophic level and generally more intelligent species. Moreover, these species are by no means of uniform composition but themselves exhibit variations in behaviour, often associated with an age structure and genetic-induced variations which may often also express itself in taxonomical differences, which also influence the overall behaviour of the species class. This observation leads to the notion of describing intelligent behaviour of animals at higher trophic levels as intelligent agents (Abbott, 1991, pp.65-70). Intelligent agents are computer-represented objects which possess 'mental states', so that they will receive messages of certain classes from other organisms as represented by other agents and will react to these messages by acting (or refraining from action) and sending on further the messages that announce these actions to other objects, which may in turn also be agents. A computer code built up from such message-passing agents is commonly said to have a multi-agent, message-passing architecture.

We observe that when introducing such an entity we implicitly divide the world into two parts. We have implicitly placed an envelope around an entity, separating it from its environment, and have chosen to focus on the transactions across that envelope. Of course, a theory of intelligence must not only describe these transactions but it must also give a clear picture of the entity responsible for these transactions.

Ecology is the study of plants, animals, people and institutions as these interact within a common environment. Any such study can only be made from the particular point of view of the individual or group of individuals conducting the study. The particular individual or group that conducts the study then imposes upon the object of its study a particular system. We call the matter about which the study is conducted within such a system an ecosystem. Intelligence is any strategy for advancing the interests of the individual organism through its actions upon its environment. We suppose that every form of life demonstrates intelligent behaviour, and an ecosystem is the consequence of the interactions of different forms of life exercising their individual intelligent behaviours. Thus we may say that ecology is the study of plants, animals, people and institutions in which each exercises its own intelligence, and an ecosystem an emergent property of these interacting intelligences.

Individual intelligences are embodied in models to which we refer as intelligent agents, but then of situated agents (Rosenschein, 1985; Rosenschein and Kaelbling, 1986). To develop models of these, situated agents is a difficult problem because of their close interaction with
the environment in which they are situated. Our ultimate aim in designing and developing situated agents is to have them perform a rich set of tasks correctly; that is, at each moment we want them to carry out actions that are coherent in relation to their situations and goals. In order to specify correct behaviour, a precise relationship is required between the internal state of an agent and conditions in its environment. Once this relationship is made precise, we can give clear specifications of desired behaviour for an agent in an environment and then generate a program for manipulating the internal states of the agent that satisfies these specifications.

We shall suppose that what an intelligent agent knows about itself and its surroundings is represented in its database by a set of (first order) sentences. These accordingly encapsulate the beliefs of the agent. Since our agent has beliefs, and since other agents are part of the ecosystem about which we want our agents to know, we must make it possible for agents to have beliefs about the beliefs of other agents and beliefs about their own selves. For example, a species preying upon another species will frequently need to know something about what these other species believe. In general, we say that phenotypes of one species know something about phenotypes of other species.

The ecosystem is in most cases, and at some level of resolution, dynamic in the sense that information is at best incomplete, perhaps erroneous, and frequently changes over time independently of the agent’s actions. This being the case, we can anticipate that their actions in the world become a difficult problem for many species. Actions and events in a dynamic world must be monitored by the senses in order to coordinate a genotypical agent’s actions within its surroundings. This allows the agent to predict and plan potential future situations, and especially exceptional situations, even while acting in the present.

The most fundamental aspect of the approach that we are advocating is that of emergent behaviour; but now natural life emerges through interactions between a large number of constituting agents. This emergence occurs at virtually every level: from interactions of non-living molecules without the existence of a global control that is responsible for the orchestration and coordination of the actions of the various parts to many aspects of self-organisation within various ecologies, as exemplified in, for example, the schooling behaviour present in many marine organisms. ‘Natural life on earth is organised in at least four fundamental levels of structure: the molecular level, the cellular level, the organism level and the population-ecosystem level. A living thing at any of these levels is a complex adaptive system exhibiting behaviour that emerges from the interactions of a large number of elements from the levels below’ (Taylor and Jefferson, 1994).

Obviously, every individual agent exhibits its own behaviour, and the macro-phenomena is itself a qualitatively different behaviour that emerges from all local interactions among individual behaviours. The more than 2000 years old philosophical statement ‘The whole is more than the sum of its parts’ strongly supports this perspective. It is this bottom-up, distributed, concurrent, collection-of-locally-determined behaviours that defines the methodological approach that we are pursuing.

In order to deal with this multi-level complexity, a broad methodological shift is in progress in the biological sciences today. This approach has been particularly strongly propagated in
Emerging Solutions

the closely related field of Artificial Life. As pointed out so eloquently by Langton (1988):

"This approach studies natural life by attempting to capture the behavioural essence of the constituting components of a living system, and endowing a collection of artificial components with similar behavioural repertoires. If organised correctly, the aggregate of artificial parts should exhibit the same dynamic behaviour as the natural system."

This bottom-up modelling technique is universal in the sense that it can be applied at any level of the hierarchy of living systems in the natural world. On the one side of the scale are molecular dynamics processes that occur on microsecond time scales, and on the other side are the evolutionary process in populations that occur over millennia. 'At any such level, behavioural primitives are identified, and rules for their activation in response to the conditions in the local environment are specified. If such an approach is followed, the primitive, elementary behaviours are naturally organised similarly to their native counterparts, and the behaviour of interest is allowed to emerge 'on the shoulders' of all the myriad of local interactions among these low-level primitive behaviours taken collectively' (Langton, 1988).

Such an approach provides many advantages, several of which were in fact recognised more than thirty years ago. For example, as pointed out by Crosby (1963):

"For some biologists, experiments with living organisms are hardly practical. For example, many problems of evolution would obviously need too much time. As an alternative, experiments with realistic models ... would go far towards overcoming this difficulty, if a sufficient speed of operation could be achieved. This is where the electronic computer can became a valuable tooi for modelling...

The computer can simulate, in mathematical terms, complex genetical and evolutionary systems, and mathematics can largely be eliminated by forming, within a computer, electronic 'organisms' possessing electronic 'genes'. These organisms can reproduce themselves in any way we choose, obeying the ordinary laws of genetics or any other pattern of heredity; while the natural variability of real biological systems can be initiated with fair realism by the use of computer-produced numbers (pseudo-random numbers) in sequences which effectively imitate true randomness."

Thus, analogy leads us further towards the synthesis of artificial worlds within artificial media, like those of the computer. However, the traditional approach to modelling, as applied in many cases in the past, is also rather inappropriate and indeed it does not suffice to provide the most fundamental requirements that the proposed model should fulfil. Traditionally, a computer program is based on a centralised control structure with some pre-defined access to a large set of pre-defined and fixed data-structures. For the present purposes and ambitions, a new approach to modelling is also necessary, this being one which moves the ongoing dynamics to the central place in the modelling arena, rather than to the final result.

One of the most essential components that the model of a natural system must accommodate arises from the observation that natural living systems are fundamentally parallel. This is
reflected in the ‘architecture’ of natural living systems, which consists of many millions of parts, each one of which has its own behavioural repertoire. Again, as noted by Langton (1988):

"Living systems are highly distributed, and quite massively parallel. If our models are to be true to life, they must also be highly distributed and quite massively parallel. Indeed, it is unlikely that any other approach will prove viable."

The essential features of such a ecosystem models would be (Langton, 1988) that:

- they consist of populations of simple programs or specifications,
- there is no single program that directs all of the other programs,
- each program details the way in which a simple entity reacts to local situations in its environment, including encounters with other entities,
- there are no rules in the system that dictate global behaviour,
- any behaviour at levels higher than the individual programs is therefore emergent.

Accordingly, the most natural alternative is to model the higher trophic levels as though they were composed of a large number of interactions among a number of individual agents each of which makes its decisions locally. Such a structure-orientated approach places at the core of the modelling activity local micro-interactions, and simply observes macro-phenomena as they emerge at the level of an ecosystem.

The shift we are presently experiencing from analytically and numerically treated models to models studied by simulation is profound and has fundamental methodological and ontological consequences. An important side effect of this shift from models allowing for analytical (and numerical) solutions to models which are themselves provided by simulations is that instead of studying a class of model systems (i.e., the different parameter values and different initial conditions), one studies just one, fully specified system at a time. This is, of course, why analytical solutions are supposed to be more valuable than simulation results. However, there are a number of reasons why this latter conclusion is not necessarily correct (Hogeweg, 1988, 1989), as follows:

- The class of models studied in analytical solutions is shaped primarily by solvability, and not by problem-orientated considerations, and its relevance therefore can be limited.
- The class of agent-orientated modelling systems is of variable structure, and only this property genuinely provides a possibility to study intrinsically variable-structured natural systems.
- The behaviour of distributed, agent-orientated systems is not studied in terms of predefined changes in variables and predefined structural changes, but in terms of
novel concepts which emerge during the simulation.

Such systems need a novel technology, not only for the specification of the system, but in particular for the observation of the system.

The usefulness of studying fully specified systems is particularly evident when the model is supposed to represent one specific system (i.e., a particular lake, or the Orkney-Shetland region of the North Sea). Such models are of fundamental importance in the engineering context of ecosystem management, when, for example, a managerial decision has to be reached that deals with the use of pesticides.

As Schull and Levin (1964) point:

"Digital computers ... now afford us the opportunity to seek numeric answers to some of the problems which have thus far proven to be mathematically [analytically] intractable. Simulation methods admittedly lack the appeal of explicit mathematical statements, but if one is pragmatic, these methods hold great promise for an early insight into a variety of interesting and important problems. In fact, at this juncture, it may well be that numeric analysis is more rewarding than an analytic, mathematical approach."

Table 126 Comparison of classical and intelligent agent approaches to modelling (adapted from Hogeweg, 1989)

The situation in which we currently find ourselves is similar to the one when, at the beginning of modern science, Bruno, having heard Galileo's theories concluded: "There are many worlds like our own". And we might add, "And under valid modelling assumptions they can all be captured within artificial media, like those provided by the computer."
7.3. Description of the Intelligent Agent’s Structure

Obviously, one of the major concerns of the proposed approach is the generation of the intelligent behaviour of an individual agent. The structure of the general-purpose agent has been described in Chapter 2. Here, the emphasis is placed on the intelligent agent’s elements that are ‘responsible’ for life-like behaviour. The most logical starting point is to identify the mechanisms by which behaviour is generated and controlled at the level of individual agents in natural systems and to re-create these mechanism within artificial systems.

Firstly, a word of clarification. An intelligent agent is here understood as an entity which is used to model both the behaviour and the physiology of an individual organism or collection of organisms (e.g. a herring school). In the continuation of this text, then, whenever the term ‘agent’ is used, it implies either an individual organism or an association of such organisms.

The agent, being a model of an individual, should represent the animal’s motivational state. Motivational state is defined (McFarland & Bosser, 1994) as a combination of the physiological and perceptual state of agent, which includes factors relevant to its incipient activities as well as to its current behaviour.

The motivational state of an organism in any particular state depends on its physiological state, on the cue state arising from its perceptions of the external world and on the consequences of its current behaviour. The behaviour of an organism has in its turn

67 Just as an example. Valiela (1978, p. 373-374) reports on a tendency of plankton to form swarms of individuals that can be conveniently treated as individual agents within our approach. Thus:

"The shape and size of clusters are very fluid, but there is a strong tendency for social plankton to reconstitute clusters even after disturbances. There is a clearly strong impetus to remain in the swarm, as shown by motion pictures of swarms of pelagic mysids. Individuals that lag behind their swarm make rapid leaping movements to regain their place in the school, and even overcompensate for their temporary isolation by swimming well into the swarm on their return. Smaller groups of mysids coalesce with larger groups if they happen to be less than about 25 cm away from the larger swarm.

Underwater observations and motion pictures of mysid swarms show that, in spite of great variability in the size of the swarm (13000-32000 individuals), the average nearest neighbour distance is quite constant (1-4 cm) and is maintained at less then 4 mysid lengths. Similar small distances are maintained by schooling fish and euphausids. Euphausia pacifica swarms with an average distance of 1-2 cm among individuals, while the density of Euphasia Superba can be 1 individual per cubic centimetre.

Some aggregations are due to reproductive phenomena, not necessarily involving social behaviour. For example, eggs and larvae may be patchy due to the occurrence of reproduction in one area. Such aggregations may last for a considerable time. Copepod and mysid clusters are made up primarily of individuals at specific stages in the life history or at similar stages of sexual maturity. In Euphasia Superba high numbers of cast molts are often collected in specific net hauls, suggesting that the molting of individuals within a cluster is synchronised."
consequences that depend on the situation in which the behaviour occurs. These consequences influence the organism’s motivational state in five main ways: by altering external stimuli perceived by the organism, and hence altering the cue space; by direct feedback onto the motivational space; by altering the organism’s physiological state; by preventing the organism from engaging in some other behaviour; and by energy expenditure. Energy, and other physiological commodities are expended as a result of all behaviour to a degree that depends on the level of activity, weather, etc.

The consequences of behaviour can be represented in a motivational state space. The motivational state of an organism is portrayed as a point in the space. As a consequence of the organism’s behaviour, the state changes, and so the point describes a trajectory in this space.

Collins and Jefferson (1991) have addressed the issue of the design of the intelligent agent’s controller. In the study they reported on analysis of the richness of several representation formalisms. The six properties that an artificial organism representation should possess were identified as: (1) syntactic closure, (2) smoothness of phenotype operations under genetic alteration, (3) scalability to a large number of inputs and outputs, (4) symmetric behaviour, (5) the ability to code for both discrete and continuous behaviours, and (6) a computational model that is uniform.

The analysed representation formalisms were those of: LISP-like S-expressions, finite state automata (FSA) and artificial neural networks (ANNs). The conclusion of this study was that the ANN provides the most rich representational properties; however, FSA and S-expressions offer possibilities for rich repertoire encoding, while still being symbolic. Although Collins and Jefferson choose ANN as their representational formalism, for the purposes of our present work we shall retain the other two representational possibilities as well.

FSA (and for that purpose S-expressions as well) have, in certain situations, advantages over ANNs. Above all, the motivational state of an animal is naturally embodied in the FSA. Secondly, it fully accomplishes the discrete event requirement inherent to the higher trophic levels. Thirdly, the ANN’s appearance is strictly sub-symbolic, thus preventing the biologist from intervening, whereas the FSA’s appearance is symbolic, often allowing a straightforward explanation of the model behaviour. And lastly, there is the possibility to induce FSAs automatically by means of system identification procedures and evolutionary algorithms (Koza, 1992, Fogel, 1992).

This last aspect has another, and considerable impact on the performance of the proposed system. We will surely find that the essential machinery of living organisms is very different from the machinery of our own invention, and our perceptions of ‘what the organisms should be doing’ may be erroneous, and we would be quite mistaken to attempt to force our

---

68 Jefferson’s preferences towards ANNs were not founded on the representational richness only. The long term objective of studies carried out by Jefferson in particular (see Jefferson, 1992), is the elucidation of the interaction between evolution and learning, and ANNs, with their generally well-behaving error propagation routines, lend themselves particularly well to such an objective.
preconceived notions of abstract machines onto the 'machinery' of life. The difference, once again, lies in the exceedingly parallel and distributed nature of the operation of the 'machinery' of life, as contrasted with the singularly serial and centralised control structures associated with machines of our own invention. The main idea with the use of evolutionary techniques (Genetic Programming in particular) for the induction of FSAs is that the user would be able to identify a database of animal behaviour (consisting of environmental stimuli, organism states and responses) and the evolutionary algorithm would itself induce the model of the animals's behaviour. This indicates a strong link with the data collected in the real world. In such a way, the model's relation with the real world's natural system would be maintained at all times. The consequence is that the FSA's appearance is symbolic (as opposed to the ANN's sub-symbolic form), but this symbolic system, because it is induced on the basis of observations, is grounded.

This is in full accordance with the 'nouvelle AI' advocated earlier in this thesis as well. The fundamental position adopted is that an intelligent system must have its (symbolic in this case) representations grounded in the physical world. And the best possible model of the world is itself and not some agent's internal (usually much simplified) representations of this world. To build a model of an agent in accordance with such a grounding hypothesis, one needs induction modelling technique which makes high level abstractions concrete. The human engineered models are typically not grounded in this sense. One such technique is genetic programming. Accordingly, we evolve solutions instead of engineering them. An example of an application of Genetic Programming to a behaviour-generation problem is reported by Koza et al. (1992). Naturally, in certain cases and if required, FSA could be modified by manual design and coding, giving the fullest freedom to the user of the system.

7.4. The Agent-Orientated Superstructure

The approach that is proposed within the current framework necessarily consists of hydrodynamic, advection-diffusion and eutrophication platforms coupled to an Agent-Orientated Superstructure (AOS).

The basic principle is one in which the platform-software will act as a model of the world in which individual, autonomous, intelligent agents (represented as individual ANNs or FSAs) are situated. The agent-orientated superstructure is the software that makes possible the creation and parallel execution of programs representing individual agents.

Following the requirements set previously, the AOS might best be described as a general purpose simulation package which supports the investigation of concurrent and distributed computational systems. These systems are the ones in which a large number of autonomous agents interact with one another and with a dynamically (HD-simulated) changing environment. The AOS provides general-purpose utilities for designing, implementing, running and analyzing such multi-agent systems. From an architectonic point of view, the AOS supports bottom-up modelling in its very essence.

Once the AOS is coupled to the HD platforms, it will forms the core of a powerful new kind of tool for ecological simulation, capable of modelling population behaviour to a finer level of detail than any of the presently available, top-down modelling tools. This new tool is called ECCO (Ecological Consequence).
In order to underline the fundamental principles again, but ‘in the opposite direction’, so to speak, our approach follows from the observation that the life of an organism is in many ways similar to the execution of a program, and that the global (emergent) behaviour of a population of interacting organisms is best emulated by the behaviour of a corresponding population of co-executing programs. Under ECCO, each individual organism is represented as a separate object, executing its own code based on its own perception of its own neighbourhood and possessing its own memory. Environmental processes (weather, seasons, nutrient replenishment, etc.) are each represented by separate programs as well. Most of the environment (if not all) in which agents are situated, will ultimately be modelled by the means of existing hydrodynamic modelling tools (e.g. two-dimensional MIKE 21, and even the three-dimensional Telemac, or SYSTEM 3 in its fourth-generation, MIKE 3, metamorphosis). Just as organisms are born, live, interact with each other and with the environment, reproduce with modification, and die, so too can animal programs initiate, execute, learn, communicate, interact with environment-simulating programs, replicate with modification and terminate. Following these fundamental principles, the AOS simulates a population directly, sometimes over many generations and correspondingly long time spans, by concurrently executing all of these programs.

Similar, agent-orientated simulation models have been developed in the USA and Japan and were used for mosquito control (Fry et al., 1989), prey-predator simulations (Taylor et al., 1988), population dynamics analysis with spatial and temporal resolutions (Toquenaga et al., 1994), as well as in Europe as exemplified in the MIRROR modelling paradigm (Hogeweg, 1988). A similar approach has recently been applied to a range of diverse problems extending from the analysis of prebiotic evolutionary processes (Boerlijst and Hogeweg, 1992) to the study of interaction among Orang-utangs and their environment (Boekhorst and Hogeweg, 1994), However, to the best of our knowledge, no specific developments suited to the aquatic environment have yet been advanced.

7.5. The Shift in Paradigm

There are two possible perspectives on life: as an embodied phenomenon and as a concept. Foucault (1973) points out that up until the end of the 18th century; ‘life’ did not exist: only living beings existed. Living beings were understood as a special class in the series of all things in the world. To speak of life was to speak only in the taxonomic sense of the word. Natural history dominated the classical age and this was first and foremost an exercise in naming. ‘The naturalist is the man concerned with the structure of the visible world and its denomination according to its characters. Not with life.’ (Foucault, 1973, p.161). There was a concurrent interest in how things work, but it was an interest that remains disconnected, resting in a state of tension with the naturalist tradition.

In the early 19th century, natural history made a decisive step towards biology when the notion of ‘character’ became subordinate to the notion of ‘function’; that is, when classification became comparative anatomy. Life was conceptualised as something functionally organised, but organisation remains foreign to the domain of the visible. A character is weighted according to the importance of the function(s) to which it is linked. In contrast to
the classical age, characters are seen as signs of an invisible deep structure. Causal argumentation is reversed. A character is not important because it occurs frequently, but rather it occurs frequently because it is functionally important. Life as a concept becomes manifest as organisation, organic structure, that is, organism. ‘In the classical period, living beings are perceived largely as points in a coordinate system of names’ (Fontana, et al., 1994).

Within the current change of paradigm, organisms require an additional ‘space of organisational structure’, if only because of the emphasis on individuals which are then allowed to aggregate and engage with other individuals, whereby the question of social organisations springs up ‘spontaneously’ in the centre of the stage. This is particularly important, since this social organisation is not fixed explicitly by the modeller, but is allowed to emerge as the individual agents interact. This sets the stage for considering the problem of the modifications and transformation of organisation. The view taken here is that the intelligent agent architecture is the only possible approach that provides a conceptual framework capable of addressing these issues.

One of the goals of the AOS superstructure coupled to the model of the physical (and chemical) world is to ‘save’ ecologists from having to deal with the computer scientific issues involved in the implementation of such concurrent, highly-parallel modelling environments. For example, an ecologist investigating zoo plankton, can easily describe procedural functionalities that are the most relevant to the particular species. For instance, the algorithms describing movements of *Calanus finmarchicus*, along with its breeding rate, its senses and sensitivities and its ageing processes, can easily be specified, even if not so easily determined. In addition, the ecologist can provide his or her agent with specific instincts, or provide it with experience-collecting algorithms. In short, he or she can bestow upon the organism whatever characteristics are deemed necessary within the context of the work. Therefore, the specialist can concentrate on the issues relevant to his or her own discipline, without having to work on the issues that are unconnected with his or her area of expertise.

This modelling paradigm not only permits the user to create a computational model of an individual organism, but to describe its behavioural patterns. Moreover, the open architecture of the hydroinformatic system allows for the integration of various scientific disciplines, such as botany, zoology, sedimentology, etc. It can also be easily up-dated as new facts emerge from fundamental laboratory research and field investigations.

In what is really the same vein, such an approach takes a rather specific form of a tool for electronic knowledge encapsulation. As knowledge about the appearance of various species accumulates, a database on their physiological properties and different behavioural aspects can be easily stored, so as to establish a rich fund of expertise in an electronic form. Such working models of individual intelligent agents can, with time, grow to a large collection, and even form an electronic reserve of organisms that can be used by various interested parties.

Thus the system makes its own kind of contribution to ‘consensus building’ across the entire scientific spectrum. This is the only approach that is capable of demystifying science in this area and allowing it to accumulate knowledge for a larger audience. Its value can therefore be measured in both scientific and social terms. While knowledge is undoubtedly power, the power within our society frequently lies with those who have little specialist knowledge and
are thus ill-equipped to make far-reaching decisions. In the past, the environment has suffered as a result of this phenomenon. An intelligent agent based approach should help to ensure that it fares better in the future.

7.6. A Related Approach - Cellular Automata

Cellular automata (CA) constitute another approach to ecological modelling that has gained in popularity over the last decade as a modelling ideal. Similarly to the intelligent-agent approach, the cellular automata approach is essentially a bottom-up paradigm, allowing the global dynamics to emerge 'on the shoulders' of local transition rules.

A cellular automaton is a discrete dynamical system; it is discrete in time, space and state. Space is thus organised as a lattice of cells. Each cell can be in one of the finite number of states and is itself modelled as a finite-state automaton. Transitions from one cell's state to another are determined through the application of some local rule. Thus, the current state of a cell depends solely on the state of the cell at the previous time step and the states of the nearby cells (neighbours) at this same, previous time step. More formally, a cellular automaton is a tessellation of identical, finite-state automata which occupy cells.

The updating of all the cells in the lattice is performed simultaneously; therefore time in the cellular automaton system advances with these same discrete time steps.

Should the lattice consist of $Q$ sites, and in the case that every site can take any of $t$ possible states, then the global dynamics could be determined by some global transition function $\Delta$ as follows:

$$\Delta : Q \rightarrow Q'$$

The same can be achieved through the application of a local transition function $\delta$ that takes as inputs the state values of $n$ possible neighbouring points. The local transition point then becomes:

$$\delta : Q^{n\rightarrow} \rightarrow Q$$

The most important consequence is that this local transition function $\delta$ can be defined in a finite lookup table. All the local transition functions can be combined into global one. For a particular cell, $c$, this becomes:

$$\Delta(c)_i = \delta(c_{i-n}, \ldots, c_{i+n})$$

Recently there has been a significant increase in the number of articles that deal with cellular automata and ecological modelling. There have also been several investigations into how various aspects of CA affect their usefulness in ecological models. What is of particular
interest with the CA-like, bottom-up approach, is its ability to reproduce in quantitative detail the results of an experimental procedure. Biological systems lend themselves in particular to modelling based on cellular automata. The spatial and temporal patterns that occur in these systems are enormously diverse and often fascinating.

One of the major questions that needs to be addressed is that of the CA’s synchronicity. For example, (Silwertown et al., 1992) have developed a CA model of competition between grass species and this has encountered no problems at all with a synchronous operation of rules. At the same time, Hogeweg (1988), among others, has claimed that the simultaneous updating of all cells is not in accordance with the localness of interactions that is one of the strengths of CA. It has been shown that changing the definition of the CA to allow for an asynchronous updating of cells can dramatically alter the behaviour of the CA (Hogeweg and Hesper, 1993; ALIFE IV, 1994). In particular, frequently an interesting structure seen in the evolution of a CA can be shown, in fact, to be an artifact of synchronous updating.

Another issue that needs to be investigated is the problem associated with multiple scales, both spatial and temporal. Typically, CA models are developed with the assumption that a cell is a physical region of the right size for one (or perhaps a few) individuals. Consider a model of zooplankton (Abbott et al., 1994). It is safe to assume that over a certain horizontal distance, conditions are homogenous; therefore it suffices to limit ourselves to a water column (that is to say that only one spatial dimension is required). In a typical situation, the concentrations of plankton are so high that it is impractical to assume one (or a few) plankton per cell. The relevant scales for the diffusion of plankton nutrients (such as nitrate) is even smaller. Also consider three typical organisms: flagelates, diatoms and copepodes. The ratio of the fastest sinking rate to slowest is 200:1. Thus if we use this information to proceed spatially and temporally, we will also run into difficulties (and again, the temporal scale of diffusion for nutrients is smaller still).

In fact we arrive at a quite fundamental question: "Are ecological systems compatible with action-at-a-distance characteristics?" Returning to the plankton model, we note that during periods of chlorophyll maxima (blooms of plankton) it is not uncommon for the presence of plankton at the surface to decrease the amount of light proceeding to plankton at depth due to shading. Thus, there is an effect that is (rapidly) felt at a great distance. But every CA neighbourhood is local only. Is this incompatible with CA methodology?

7.7. Modelling Ecosystems with Intelligent Agents

Ordinarily a computer user would construct a problem, feed it in, and wait for the machine to calculate its solution - one problem, one solution... [The] chaos researchers... needed more. They needed to do what Lorenz had done, to create miniature universes and observe their evolution. Then they could change this feature or that and observe the changes paths that would result. They were armed with the

---

69 Parts of this section have been published as an article entitled Modelling Ecosystems with Intelligent Agents, written by Abbott, Amdisen, Babovic, Baretta and Dørge that appeared in the proceeding of the First International Conference on Hydroinformatics.
new conviction, after all, that tiny changes in certain features could lead to remarkable changes in overall behaviour.

James Gleick, *Chaos*

As a launching platform for intelligent-agent-based modelling, a somewhat cautious approach has been chosen. An existing eutrophication (EU) module that is believed to produce sufficiently accurate results for the lower trophic levels has been employed as a platform. At the present time, the EU module can itself be mounted upon all of 1-, 2- and 3-dimensional hydrodynamic modelling systems. The formulations of the processes describing carbon, nitrogen, phosphorus, chlorophyll-a, dissolved oxygen, benthic vegetation carbon and zooplankton variations in time and space are dependent on such external factors such as salinity, water temperature, light-influx and discharge. These inputs are provided by the underlying hydrodynamic models. All the mathematical formulations of the processes are described as first-order, ordinary, coupled differential equations.

General experience obtained through using the EU module on a number of studies has proven its high quality as a modelling tool for those lower level components for which the assumption of the well-mixed nature of the processes is valid. The modelling of the zooplankton, however, often fails to produce meaningful results. The cause is then sought in the absence of behavioural components which cannot be at all easily embodied in a system of ordinary differential equations form. It is well known that, for example, zooplankton can propel itself and follow a certain light intensity and temperature gradient. Based on food abundance, it can alter its feeding strategies. Moreover, there can be up to twelve discrete developmental stages of zooplankton, and each developmental stage exhibits a different behaviour. Moreover, there are three different classes of zooplankton (herbivorous, carnivorous and omnivorous) and their behavioural repertoires are markedly different.

It is therefore quite natural to choose the EU module as a launching point for this new approach based on intelligent agents. On the one side, the behavioural aspects necessitate the introduction of an approach where the zooplankton would be shaped individually. Following this idea, an agent will make its decisions locally, based on its perception of its immediate neighbourhood and its own physiological and motivational space, and the behaviour of the entire system will emerge from these interactions.

Within the context of the present efforts, an agent-based model of zooplankton will be developed and interfaced to the EU module, so that the descriptions of the ‘lower’ components remains intact. The description of the zooplankton will be on an IAA basis. The sensory information necessary for each intelligent agent’s behaviour will be derived from hydrodynamic and EU platforms (and possibly a few more, like temperature, light, salinity and season), or modelled through evolutionary algorithms. The actions taken by the IAs will also have some influence on the equations describing the state variables.

As introduced earlier, the present Lotka-Volterra models are usually mounted as superstructures upon hydrodynamic (HD)-modelling platform and coupled in parallel with an advection-diffusion (AD) superstructure. The HD platform then provides a field of velocities and water depths, the AD platform advects and diffuses all the species (including salt, heat and chemical substances of interest) and the ecological model regularly allows these various
species to interact. Although this approach has been found satisfactory for many applications involving lower-trophic-level organisms, with good agreement often being observed with measurements over extended space and time scales, it is often found to be considerably less than satisfactory when describing the higher trophic levels.

We have also observed earlier that in many, if not most cases, the divergence between computed and physically-observed results can be attributed to insufficient account being taken of the behavioural processes that guide the behaviours of the higher-trophic-level and generally more intelligent species. Moreover, these species are by no means of uniform composition but themselves exhibit variations in behaviour, often associated with an age structure which may often also express itself in taxonomical differences, which also influence the overall behaviour of the species class.

We repeat that this observation necessarily leads to the notion of describing the intelligent behaviour of at least some of the zooplankton as intelligent agents, these being computer-represented objects which possess 'mental states', so that they will receive messages of certain classes from other organisms as represented by other agents and will react to these messages by acting (or refraining from action) and sending on further the messages that announce their actions to other objects, which may in their turn also be agents.

We have also already introduced a computer code built up from such message-passing agents as one having a multi-agent, message-passing architecture. In most of the applications of these architectures that have so far been realised, the mental state of an object is represented by the values of its attributes. The object is generally specified only by what it does - so how it reacts to a given message - and not by the way in which it is realised in code. Objects may themselves be grouped together in classes and their attributes may be inherited through subclasses in the usual way of object-orientated analysis (see, for example Coad and Yourdan, 1990, 1991). The model described here constitutes one demonstration of the ongoing ECCO (Ecological Consequence) project.

It is consequence of the construction of objects in this way that the user of the model is obliged to view the denizens of the ecosystem as intelligent creatures, each with an own set of intentions, frustrations and satisfactions. These denizens become, in effect, 'individualised' in the mind of the user of the system, so that this user is led to adopt a new kind of solicitude towards the individual creatures that in fact make up the ecosystem.

7.7.1. The Prototype System Architecture

For the state of simplicity, the prototype system has been built upon a well-proven one-dimensional modelling system, MIKE 11, and, besides using this to provide HD and AD modelling facilities, this prototype makes use also of the MIKE 11 user interface. The prototype has been mounted upon an existing eutrophication (EU) module, which is presently based upon a system of 12 differential equations describing variations of 12 components. This EU module has been extensively tested and applied in many and various studies and has generally provided excellent agreement with measurements so long as species of higher trophic levels were absent. However, serious discrepancies between model results and measurement data have often been observed when zooplankton are present in significant
quantities. Therefore, we have chosen to replace the differential-equation-described zooplankton with 'intelligent agents', while retaining the other 11 equations, for the moment, in order to describe the lower trophic levels.

The prototype system is then said in the present context to have an intelligent agent architecture (IAA) that is run interactively with an existing, production eutrophication (EU) component of MIKE 11. In this space- and time-distributed EU component, however, each vertical through a grid point is now divided into \( n \) 'boxes' which, for the purposes of the present component, can be supposed to be hydrodynamically uniform. The benthic processes are then integrated over these same \( n \) elements in the EU sub-model. At each time step the zooplankton process rates are determined by the IAA component while the EU component process is then sent on its normal way through the next time step using the process rates so provided. The messages transferred between the IAA and EU components are constituted here by ASCII files. The test case used here follows the development of copepods as representative individuals and, although based on a distributed, deterministic description of the copepod physiology, uses a knowledge-based description of the mental processes of the representative individual copepods. The IAA component reads from such results produced by the EU component as egg concentration and nauplius and copepod populations in each box, and it sends a message to be read by the EU component, providing such parametric data as grazing and elimination rates. This message-passing structure is schematized in Figure 127.

![Integration of the IAA and the EU sub-models](image)

**Figure 127 Integration of the IAA and the EU sub-models**

### 7.7.2. The Overall Data and Knowledge Structure of the IAA

As already introduced, the IAA approach is based upon the individual behaviours of a reduced
number of copepods in the water column, i.e. a relatively few ‘individuals’ are used to represent thousands or even millions of copepods. The behaviour of each individual, which constitutes an agent in the code, is the result of an evaluation of its actual state (e.g. hungry or satiated), its environment, and the rules that have been defined for this agent.

A stochastic variable (noise) must be superimposed on the physiological coefficients and rules in order to model different behaviours of otherwise identical individuals. Some copepods may stay and graze where they are while others will move vertically in order to investigate whether food is more abundant elsewhere. Once coupled with the advective-diffusion components, of course, these individuals will also be advected and diffused with the fluid. The movements due to an own horizontal locomotion can similarly be incorporated when required (e.g. Abbott and Warren, 1974).

The following data has then to be stored in the IAA component or sub-system:

- Number of individuals: Egg, Nauplii (stages 1-6 in principle, but initially lumped into one stage only) and copepod(ite) (stages 1-5, adult) in each computational box (where the concentration is read from file B, Figure 127)
- Zooplankton developmental stage and quality (Egg)
- Zooplankton size and biomass (Nauplii and copepod(ite))
- Biological and physiological parameters (taken as local constants)
- Actual state (hungry/satiated, egg producing/egg laying, grazing/moving, etc.)
- Rules (feeding strategies, individual behaviour for vertical migration, energy pay off, etc.).

<table>
<thead>
<tr>
<th>Agent in (Egg, Nauplius-Copepodite, Copepod-Male, Copepod-Female)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Egg:</strong></td>
</tr>
<tr>
<td>States : (Developing, Resting)</td>
</tr>
<tr>
<td>Variable : (Percentage development, Quality)</td>
</tr>
<tr>
<td>Processes : (Development, Sinking rate, Being grazed)</td>
</tr>
<tr>
<td>Remarks : Temperature dependent development.</td>
</tr>
</tbody>
</table>

| Nauplius copepodite:
| States : (Eating, Moving, Eating-Moving) |
| Variable : (Size (=stage), Biomass (internal energy)) |
| Processes : (Grazing, Death, Excretion, Respiration, Production) |
| Remarks : All processes are functions of the individual biomasses. |

*Figure 128 Formalisation of Intelligent Agent’s properties*

When transforming from one stage to another (that is, during metamorphosis) production will stop and some internal energy will be used. After a copepodite reaches stage 5, the adult stage
is reached and it is assumed that 50% of the individuals will be females.

**Copepod-Male:**

<table>
<thead>
<tr>
<th>States</th>
<th>(Eating, Moving, Eating-Moving)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>(Biomass (internal energy))</td>
</tr>
<tr>
<td>Processes</td>
<td>(Grazing, Death, Excretion, Respiration)</td>
</tr>
<tr>
<td>Remarks</td>
<td>High death rate.</td>
</tr>
</tbody>
</table>

**Copepod-Female:**

<table>
<thead>
<tr>
<th>States</th>
<th>(Eating, Moving, Eating-Moving)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>(Egg (7 batch), Biomass (internal energy))</td>
</tr>
<tr>
<td>Processes</td>
<td>(Grazing, Death, Excretion, Respiration, Egg Production)</td>
</tr>
<tr>
<td>Remarks</td>
<td>Egg production is calculated from increased biomass. Egg development period is one week and the history for each day is saved in order to model a varying laying of eggs.</td>
</tr>
</tbody>
</table>

*Figure 129 Formalisation of an Intelligent Agent representing male and female Copepod*

### 7.7.3. The Data Structure of the Physiological Processes

The state of each agent is defined by the following state variables:

- Development stage (egg, nauplii, copepod(ite) in stages 1-5, male or female)
- Structural weight (SW)
- Energy storage (ES)
- Number of eggs as distributed in seven daily batches (only Adult Female)
- Percentage development (only eggs)
- Position in water column.

In order to model the physiological development and the transition from one developmental stage to another, some basic physiological constants, as listed in table 1, are used.

At each time step, the physiological processes for the individuals are calculated based on the state resulting from the previous time step. The results of the physiology routine indicate whether the total weight (SW + ES) increases or decreases, whether a transformation from one stage to another is occurring, and so on.

The internal energy balance is calculated according to the physiological processes schematized in Figure 131.
Figure 130  Physiological constants for Calanus finmarchicus (Gunnerus)

<table>
<thead>
<tr>
<th>AGENT</th>
<th>AVERAGE SIZE</th>
<th>AVERAGE STRUCTURAL WEIGHT</th>
<th>MAX INTERNAL ENERGY STORAGE</th>
<th>MAX SPEC. GRAZING RATE day⁻¹</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mm</td>
<td>µg C</td>
<td>µG C</td>
<td></td>
</tr>
<tr>
<td>EGG</td>
<td>0.05</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>NAUPLII</td>
<td>0.58</td>
<td>2.2</td>
<td>0.94</td>
<td>1.50</td>
</tr>
<tr>
<td>COPEPOD #1</td>
<td>0.94</td>
<td>6.2</td>
<td>2.66</td>
<td>1.48</td>
</tr>
<tr>
<td>COPEPOD #2</td>
<td>1.25</td>
<td>10.4</td>
<td>4.46</td>
<td>1.15</td>
</tr>
<tr>
<td>COPEPOD #3</td>
<td>1.65</td>
<td>21.9</td>
<td>9.39</td>
<td>0.83</td>
</tr>
<tr>
<td>COPEPOD #4</td>
<td>2.10</td>
<td>47.8</td>
<td>20.49</td>
<td>0.31</td>
</tr>
<tr>
<td>COPEPOD #5</td>
<td>2.60</td>
<td>105.5</td>
<td>45.21</td>
<td>0.25</td>
</tr>
<tr>
<td>A. MALE</td>
<td>3.00</td>
<td>186.8</td>
<td>80.06</td>
<td>0.25</td>
</tr>
<tr>
<td>A. FEMALE</td>
<td>3.10</td>
<td>235.7</td>
<td>101.01</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Grazing (GRPC)

\[ \text{Energy balance} = + \text{Assimilation} - \text{Respiration} - \text{Filtration cost} - \text{Movement cost} - \text{Egg production} \]

Losses:

Respiration (REZC)  Death (DEZC)  Excretion (EKZC)

Figure 131  Conceptual diagram of the physiological model in ECCO. The accumulated processes for each water compartment (GRPC, DEZC, EKZC and REZC) are transferred to models for the biological system.
The sub-processes in this case are defined as follows:

**Filtration**

\[ F = k_{F1} \cdot SW^{k_{F2}} \]  \{ml/day\}

\[ k_{F1} = 35.0 \]  \{ml/day/μg C\}

\[ k_{F2} = 0.85 \]

**Ingestion**

\[ I = k_i \cdot F \cdot PC \]  \{μg C/day\}

\[ PC = \text{Conc. Phytoplankton Carbon} \]  \{μg C/ml\}

\[ k_i = 0.75 \]

The filtration loss due to 'sloppy feeding' thereby becomes:

\[ F \cdot PC - I \]  \{μg C/day\}.

**Energy balance (accumulation/consumption)**

For each agent the calculated energy balance results in either an accumulation of energy or a consumption (both expressed in terms of carbon). Two different storage units are modelled: the storage associated with the structural weight (SW) and the internal energy storage (ES). SW consists mainly of proteins while ES consists of easy degradable matter, such as lipids and polysacharids.

The general principle or rules thereby introduced are thus that:

**In case of net energy accumulation**

If SW is less than the average SA for the actual stage, the energy is accumulated into SW; otherwise the ES is increased.

**In case of net energy consumption**

If the ES is sufficient to make energy available the ES is used; otherwise the balance of the energy is taken from the SW.

The resulting representation in pseudo code is presented in Figure 132.
<Calculation of the energy balance, \( dE/dt \)>

if {net energy accumulation}
  if \( SW < SW_{stage} \)
    if {net energy accumulation} \( > SW_{stage} - SW \)
      \( SW = SW_{stage} \) {calculation of remaining energy}
      \( ES = + \) {remaining energy} \( * SW/SW + SW_{stage}/s \) {calculation of remaining energy}
      \( SW = + \) {remaining energy}
    endif
  endif
elseif
  \( ES = + \) {net energy accumulation} \( * SW/(SW + SW_{stage}/2) \) {calculation of remaining energy}
  \( SW = + \) {remaining energy}
endelseif
endif

elseif {net energy consumption}
  if {net energy consumption} \( < ES \)
    \( ES = - \) {net energy consumption}
  endif
elseif
  \( ES = 0 \)
  \( SW = - \) {remaining energy}
endelseif
endelseif.

Figure 132 Pseudo code for energy balance

The following other functional relations and rules have also been introduced:

**Assimilation**

\[
A = k_A * I \quad \text{[\( \mu g \ C/\text{day} \)]}
\]

with:
\[
k_A = 0.75
\]

**Respiration**

\[
R = k_{R_1} * SW^{k_{R_2}} \quad \text{[\( \mu g \ C/\text{day} \)]}
\]

with:
\[
k_{R_1} = 0.15 \quad \text{[day}^{-1}]
\]
\[
k_{R_2} = 0.75
\]
Filtration costs

\[ F_c = k_{fc} \cdot F \]  \( \mu g \ C/day \)

with:

\[ k_{fc} = 0.04 \]

Movement costs

\[ M_c = k_{mc} \cdot (SW + ES) \]  \( \mu g \ C/day \)

if speed = max (240 meter/day) then \( k_{mc} = 0.04 \ \text{day}^{-1} \) else speed is the half, and \( k_{kc} = 0.01 \ \text{day}^{-1} \)

Egg production

if ES > 0 then the number of eggs produced is

\[ E_p = \min (k_{E_{p\text{max}}} \cdot ES/k_{E_p}) \]

with:

\[ k_{E_{p\text{max}}} = 60 \]  \( \text{egg/day} \)
\[ k_{E_p} = 0.2 \]  \( \mu g \ C/\text{egg} \)

and the energy costs (taken from ES) is

\[ E_{pc} = E_p \cdot K_{E_p} \]  \( \mu g \ C/day \)

Egg development

\[ E_d = k_{E_d} \cdot k_{ED2} \cdot \text{Temp} \]  \( \text{day}^{-1} \)

with:

\[ k_{E_d} = 5.47 \]  \( \text{day}^{-1} \)
\[ k_{ED2} = -0.099 \]

Detritus production

\[ D_p = I - A \]  \( \mu g \ C/day \)

(Ingestion - Assimilation)

Transition

If total weight (SW + ES) > (SW^{stage1} + SW^{stage2})/2 + SW^{stage} and enough ES is available (5% of total weight) then transition takes place.

The costs associated with the transition are given by:
\[ T_{c} = k_{Tc} \cdot (SW + ES) \quad \{\mu g \, C\} \]

with:

\[ k_{Tc} = 0.05 \]

Energy is transformed from ES to SW when changing from one developmental stage to another according to the relation:

\[ E_{t} = k_{Et} \cdot (SW + ES) \quad \{\mu g \, C\} \]

with:

\[ k_{Et} = 0.05. \]

### 7.7.4. Simulation Results

The trial model is run for a prototype time of one year using a time step of 3 hours. A very simple setup is used: only one compartment with 80 females and 80 males per m\(^3\) as initial conditions is used. In order to mimic the interaction between the abundance of food and the number of grazers (copepods) without initially introducing any direct coupling to an even more sophisticated EU-model, a very simple interaction is used between times \( n\Delta t \) and \((n+1)\Delta t\):

\[ (\text{Food})^{n+1} = (\text{Food})^{n} + [\text{production}]_{n}^{n+1} - [\text{grazing}]_{n}^{n+1} \]

where:

\[ [\text{production}]_{n}^{n+1} = 0.4(\text{Food})^{n}(1 - \frac{\text{Food}^{n}}{\text{Food}^{n+1}} + 0.5) \]

and \([\text{grazing}]_{n}^{n+1}\) is calculated by the physiological component of ECCO.

![Graph](https://via.placeholder.com/150)

**Figure 133** Modelled Zooplankton Carbon using a stochastic variation in the biological coefficients
The temperature, production of food, and other external factors are kept constant over the year, so no seasonal fluctuation is modelled. These constraints have been introduced only for the purposes of testing the model and they can be - and will be - relaxed later.

The total number of modelled agents is relatively low (100-1000) as only a small representative volume of each compartment is used (0.05 m³). The results, however, reflect the dynamics in the model in a satisfactory way and the volume of the compartments to be modelled can be adjusted simply by re-scaling, with a corresponding redefinition of the spatial resolution.

In Figures 133 and 134 the modelled concentration of zooplankton (summed by the weight of each individual) and the concentration of phytoplankton are shown. A strong correlation between the concentration of food (prey) and the Zooplankton (predator) is seen, but whereas the variation in phytoplankton is observed to be rather monotonous, the modelled accumulated biomass of the copepod population is seen to be very dynamic.

In Figures 135 and 136 the number of copepods (including copepodites) and the number of Nauplii (the stage before copepodites) are presented. Here, a more stable variation is seen.
with blooms associated with good food supply and a much lower number of Nauplii than copepods.

![Figure 136](image)

**Figure 136 Modelled number of Nauplii per m$^3$**

As an example of the biological rates which are calculated inside the system and which can be transferred to any model using a zooplankton description, the accumulated excretion of carbon from the copepods is shown in Figure 137. A correlation to the actual biomass, as given in Figure 133, is seen, but this correlation becomes much weaker whenever the zooplankton blooms. The efficiency of the grazing is different from population to population due to the stochastic description and the actual composition of population (smaller or larger organisms).

![Figure 137](image)

**Figure 137 Modelled excretion of Carbon (summed by each individual). Unit is g C/m$^3$/hour**

In order to see the effects of the stochastic variability, the concentration of phytoplankton and zooplankton from a simulation *without* this facility are shown in Figures 138 and 139. It should be noted that by removing the stochastic variability, the system degenerates to a functional group description as in this case all individuals at the same stage will have the same properties. When compared with the results *with* stochastic variability (Figures 134 and 135), the overall coherence between the amount of food and the size of zooplankton population seems very similar. However, a significant difference is observed at various points
in time between the two populations. An overall 'noise' is also seen to be present in the simulations with stochastic variability.

Figure 138 Modelled Zooplankton Carbon without a stochastic variation in the biological coefficients

Figure 139 Modelled Phytoplankton Carbon (Food) without a stochastic variation in the biological coefficients

The oscillations or 'noisiness' in the populations, which we find in nature, is impossible to model without introducing some kind of fuzzy modelling. In the present approach, the use of stochastic variability of the physiological coefficients for each individual would appear to provide a good reproduction of the complicated biological interactions that we observe in the 'real' world.

By a way of further example, Figure 140 schematizes a small part of the hypothesis generation process as realised using Neuron Data's 'Smart Elements' KBS shell, that was used as a platform for the generation of the Agent Orientated Superstructure.
To conclude, a prototype of an ecosystem model incorporating intelligent agents has been designed and constructed and is currently being tested, calibrated and further developed. Although the results obtained so far are primarily indicative, they serve to demonstrate the feasibility of this overall approach.

### 7.8 Food Webs at the Edge of Chaos

In order to further differentiate among top-down and bottom-up approaches to ecological modelling, and to demonstrate the quite extraordinary potential of the intelligent-agent approach to modelling, in this section some rather abstract experiments based on practical motivations are described.

Ecologists have documented at least 10 properties of ecological communities that are repeatable from community to community. These include, for example, a small number of trophic levels, a tendency of species not to feed at different trophic levels (omnivory), and a rather constant ratio of predator species to prey species. Various theories, which are not mutually exclusive, have been put forward to account for these properties, including the increasing constraints on energy available to organisms at higher trophic levels, the tendency of predators to be larger than prey, the specialisation of predators associated with the consumption of particular types of prey, the inherent dynamical stability of certain configurations of interacting species, etc.
The scientific methodology that is proposed here is very different from the scientific methodology that characterises classical research in biology. Biological analysis involves the deduction of the mechanisms underlying living system in a great detail. Traditionally, this top-down approach, aimed at analysing complex living systems by decomposing them into sub-systems, and further analysing those sub-systems, leads to the deduction of the form and the function of each component. The methodology used here, being bottom-up, is aimed at synthesising complex, life-like systems.

Figure 141 Decomposition of an ecosystem. There are several layers of organisation that are implicit in the organisation of such a model

The objective of this study is to study how simple interactions among ‘simple’ agents lead to such emergent high-level phenomena as the flow of resources in a system, or cooperation and competition in a network of agents (eg. whether these be animal communities, trading networks or arms races).

As pointed out earlier, our experimental setup is here necessarily quite abstract. The simulation takes place on a toroidal grid populated with a number of agents. The entire ‘world’ consists of a lattice of sites. Each is populated by a certain number of agents, and there is a measure of locality within each site. Sites produce different types of renewable resources that provide inflows of energy. Different types of agents use different resources and consume these resources internally. At the same time, being at a certain site, an agent incurs a certain maintenance or metabolic cost.

Agents compete, trade and reproduce. Competing and trading result in an exchange of resources between agents. There may be both sexual and non-sexual reproduction within a population, whereby sexual reproduction results in an offspring whose genomes are a combination of those of its parents, while non-sexual reproduction is meiosis-like with mutation as the only driver towards evolutionary diversity.

Each agent’s genome encodes various genes which determine how it will interact with other agents (e.g. which resource it is willing to trade, and with which kinds of agents it will compete). Some of these genes determine phenotypic traits, or ‘tags’ that can be used by other agents. This allows a possibility for the evolution of social rules and potentially mimicry, phenomena frequently observed in natural eco-systems.
In the present experiment, each agent's controller is modelled as an artificial neural network (ANN). In order to keep this example as general as possible, the initial information supplied externally to the individual agents is kept at the lowest possible level. In this case, agents possess only information on their relative position in a food web, and thus which resources from the environment they can consume. All other events in a resulting system consisting of a collection of these agents is allowed to self-organise and is simply recorded during experimentation.

The most interesting phenomenon to be observed is that called *collectionism*\(^\text{70}\). The dynamics of local interactions in principle determine and merge into the global system's dynamics. The system's dynamics, however, influences local interactions, so that this approach, although materialised as bottom-up, allows for multiple feedbacks and a two-way flow of information. One can try to create an analogy with human culture. Human culture is, in principle, a consequence of interactions of individual human beings. However, such a (non-material) emergent property definitely influences and restricts all possible actions an agent may take. It is, for example, culturally and socially unacceptable for a human to walk naked in an industrialised European town, whereas it may be perfectly in order within indigenous tribes.

\(^{70}\) The first use of this term appears to have been by Chris Langton, at the opening of the Artificial Life IV Workshop held at Massachusetts Institute of Technology in Cambridge, USA.
in some South American tropical forests.

In our present experiment, each agent, through eating its food, collects a degree of ‘energy’, which is supposed to sustain its existence. The energy level is bounded from both sides: when it falls below a certain threshold, an agent ‘dies’, while an energy level above a certain upper threshold causes an agent to exhibit indifference. Energy is consumed in a metabolic process which depends on the degree of activity of the individual agent and a specific location of this agent within the lattice of sites.

There are several basic type of agents in this setup. The experimental setup has allowed for two families of organisms: specialists (either omnivorous or carnivorous) that feed on one type of food only and generalist (omnivorous species) that can feed on two different kinds of food sources. ‘Herbivores’, are thus agents that can feed on ‘plants’71 only. ‘Carnivores’, are consequently agents that can feed on other, usually non-carnivorous agents, and omnivores are agents that may feed on both. Further, three types of species were distinguished: specialist-producer, specialist decomposer and generalist producer-decomposer. Producer organisms fed on nutrients, the decomposer organisms fed on waste products, while the generalist could feed on either food source. The energetic content of food varies. ‘Plants’ are worth a certain fixed amount of energy, whereas in the case of predation, the internal energy is increased by the amount of ‘free energy’ contained in the prey just consumed. Of course, the predator is not getting a ‘free lunch’. In order to catch its prey, the predator spends a certain amount of its own energy, and one which is considerably higher than just the steady-state respiration, or metabolic cost.

The agents’ controllers are conveniently built as artificial neural networks which further brings in the possibility of the agents learning72 about their environment. At the initiation of an experiment, all agents are initialised with random weights within their ANN controllers. The more ‘lucky’ the agents in the sense that their initial weight distribution provides them with comparative advantages contrasted to other agents, the more often they will reproduce to provide more offspring, consequently propelling the information they encode to future generations. In such a way, an evolution in an artificially created world of a self-assembled system can be monitored.

Agents possessing a degree of energy that is above a certain level can mate with agents of their own species. The offspring inherits 40% of their parents’ energy. Mating of agents is performed in a genetic-algorithmic fashion. Each agent’s genome is combined with the one from its mating partner. The genome itself encodes for the initial weight of the agent’s controller (neural network) and the energy content.

There is no explicit fitness function guiding selection and reproduction. In the simulations

71 Plants in this case are materialised as an energy inflow to the system.

72 In all experiment performed until now, the worlds populated by agents that were not allowed to ‘learn’ did not last very long. The random initialisation of agents’ behaviours has never shown to be sufficient. It was only with backpropagation learning that agents were able to live longer and provide more interesting simulations.
carried out so far, our system has demonstrated a surprisingly complex behaviour (including something resembling a biological ‘arms race’ in which two competing agent types develop progressively more complex offensive and defensive strategies), ecological dependencies among different species, and a sensitivity (in terms of the number of different phenotypes) to different levels of renewable resources.

![Diagram](image)

**Figure 143** A part of a genome coding for individual agent’s controller embodied as an artificial neural network - note that exactly the same mapping is used in Section 6.13

An intelligent agent’s genotypes code for an artificial neural network as is demonstrated in figure 143. The network’s learning parameters and the architecture of connectivity are part of a genetic material that can be inherited by offspring. The learned weights in the present set of experiments, however, cannot. Thus, our approach is similar to Darwinian evolution (whereas should the ‘learned’ weights be inherited, we should observe some form of Lamarckian evolution, allowing for genetic inheritance of acquired characteristics).

We present only a few results from some typical simulations. The experiments begin with randomly generating and distributing even numbers of three species on the grid. Experiments were undertaken several times in order to determine the influence of the initial conditions on the experiment’s outcome. No specific influence was detected. All the simulations were run for 70,000 time units.

Populations cultured in constant environments usually developed adaptability more rapidly than those cultured in variable environments. Examination of ‘phenotypic’ properties of organisms indicated that this is due to the fact that variant types are not as strongly selected against in a constant environment. However, the situation reversed when the populations were cultured in the variable environment long enough for them to develop suitable adaptations. On this longer time scale, the adaptable populations cultured in the more variable environment usually became greater in magnitude than the populations cultured in the constant environment.
Figure 144 Results of one typical experiment. In this case energy used by three different species is plotted against simulation time. In the early stages, two species dominate the third one, until the moment when one of the species discovers (learns) the trick to utilise energy from environment to a large degree and starts dominating the eco-system.

Figure 145 ‘Energy’ content per species in a high inflow of energy to eco-system. Two species co-exist, while the third one is extinct.
Figure 146 Similar simulation as in figure 145, with somewhat lower energy inflow

Figure 147 Biomass oscillations related to simulations from Figure 144

The results of the simulations just presented clearly demonstrate that local and temporal
variations cannot be ignored - in fact, from the presented results it seems that they are crucial to the long-term evolution of the ecosystem. This change in perception is grounded partly in new ideas and advances in the mathematics of dynamical systems which show that small changes in the value of a characteristic at the start of a simulation can lead to large changes in the overall distribution of characteristics.

Figure 148 Spatial distribution of artificial eco-system

Until recently, it was very difficult, even on a computer, to define the dynamical behaviour of every individual in a large population - it simply took too long. With the current approach these problems are largely overcome. The problems that the ecologist face here are more along the line of defining the various conditions that may take decades or more to express
themselves. All artificial ecologies must be able to develop a pattern relationships both in space and time, just as the biological ecosystem does.

Artificial ecologies embodying the idea of complexity are demonstrating how nature is critically patterned by organisms living and evolving in it. Complexity is, for example, predicting that highly effective individuals may be prevented from invading an ecosystem by less effective individuals. It is also showing how historical fragments are important in the development of both real and artificial ecologies.
8. Hydroinformatics as Emergent Phenomenon

This is the closing chapter of this work about hydroinformatics, and yet so far no operational hydroinformatics system has been developed, even in prototype. Indeed no attempt at all has been made in this direction. The development of a hydroinformatics system that is any more than a mere toy requires much more than four man-years. It is a demanding task that requires a team of experts covering a number of disciplines, the author is not even an expert in one of these disciplines. Moreover, developing a hydroinformatics system is not only technologically demanding and certainly intellectually challenging, but, the related intricacies of, for example, designing efficient file exchange formats are not necessarily scientifically interesting. Thus, the task of building a hydroinformatics system is left to professional organisations and specifically to the large European and North-American institutions that can cope with the resulting demands in much better way than can any individual author.

However, this is still a work about hydroinformatics. Even if it is not hydroinformatics, per se, it is still commentary of hydroinformatics. It has been argued here that hydroinformatics is itself an emergent phenomenon. It has been argued that hydroinformatics thought in its essence is an emergent phenomenon of the myriad of computational, intelligent agents composing it. It is a new quality that emerges as a consequence of interaction of the
Hydroinformatics as Emergent Phenomenon

intelligent, computational agents that compose it. Subsequently the main course of this work has taken a direction of analysing emergent phenomena. By way of conclusion, however, some general guidelines for a design of a real hydroinformatics system should be given on the basis of experience gathered from work within other emergent paradigms.

Indeed, most of this work has been orientated towards the analysis of emerging collective behaviour. In the case of evolutionary algorithms, it has been demonstrated that a collection of relatively simple, interacting agents may result in a high-level performance the realisation of which would otherwise require the intersection of an (intelligent) human being. These evolving agents have been equipped with what are in principle simple mappings and they have been provided with a capability of sexual and a-sexual reproduction exclusively, but nothing much more than this. In a similar vein, the communication channels among agents and their environment transmitted only information related to empirical credit assignments.

This process has been taken somewhat further in Chapter 7. When both the complexity of the individual agents has been increased and the communication channels among them have been considerably extended. The resulting behaviour was then described as 'highly complex'. Indeed, such an approach has been identified as the only viable approach to modelling complex systems - eco-system modelling being the principal example. It is really only at this level that computer programs start to play the role of more traditional pieces of scientific equipment such as spectroscopes, anemometers or fluviometers. Computer programs are the seen, within this context, to form at least a basis for a primary scientific instrumentarium. By these means researchers, engineers, managers and many others are now given a new possibility to implement a hypothesis in software and to test its validity in a highly realistic fashion.

Hydroinformatics is only too often identified with a 'bag' of computer programs that are somehow put together. We see only too often the dreadful consequences of such a view when it arrives at the level of applications. Getting varied and disparate programs to work together is not an easy task. It requires a major effort in order to learn about the characteristics of the resulting conundrums and to arbitrate between different communication formats and protocols accordingly. Even when something is put together in this way, the resulting systems are usually very rigid. This applies to both the rigid behaviour of a resulting system that allows only for pre-programmed modes of interaction, and to the maintenance of a resulting system that does not allow for simple replacements or updatings of components, but rather necessitates thorough re-construction at each 'update'.

At the outset of this work it was observed that the most promising approach in dealing with issues of complexity of software - software in which even a non-structured 'bag' of computer programs presents complications enough, let alone a hydroinformatics system - and its interaction with the complex physical environment within which it is embedded, is to distribute and devolve the control mechanisms and computational tasks to a number of computational agents. The principal reason for this distribution and devolution of computational tasks has been described as a capability of a society of these agents to perform complex tasks, even though none of the agents individually would be able to accomplish such assignments.
Hydroinformatics must be able to cope with problems that require spacial, functional and temporal distributions of processing. This is becoming increasingly apparent as the hydroinformatics' approach becomes more widely followed by a wider variety of specialists (and non-specialists) in even more diverse environments. It is claimed here that the typical hydroinformatics system of the next century will emerge as a network of cooperating, intelligent, heterogenous agents, which agents can be either hardware component, a code implemented in software, or a human operator.

In order to deal with these problems, hydroinformatics must turn to inter-operability of its agents of which it is composed. Cooperation among these agents is driven not only by the need to share globally limited resources, but also because these agents can be most efficiently designed to be of bounded rationality but capable to solve tasks that are themselves mutually dependent.

It has not become the intention of this work to standardise interfacing procedures of the various programs that may enter into a hydroinformatics framework, nor it has been to assign a hydroinformatics system structure to any particular organisation. These issues are usually determined through more practical considerations, such as security, programming language choice, networking services, and so on. Here, we have only sought to outline prescriptions regarding communication languages, based on one or the other model of the transport of messages among the agents. This process of transport can now be summarised as follows (Finin et al, 1994):

- Agents are connected by unidirectional communication links that carry discrete messages;
- these links may have a non-zero message transport delay associated with them;
- when an agent receives a message, it knows from which incoming link the message arrived;
- when an agent sends a message it may direct to which outgoing link the message goes;
- messages to a single destination arrive in the order in which they were sent;
- message delivery is reliable.

Using such an approach makes possible automated inter-operation among agents. Moreover, by applying such a knowledge sharing technology (Genesereth and Singh, 1995) the agents are capable of carrying an inter-operability overhead, thus leaving the designers to deal with the domain-specific problems of the individual agents without having to deal with the details of every other agent's behavioural style at the same time.

The task of providing consistent solutions to interdependent problems by a society of agents can, however, be difficult one. On the one side, individual agents may not have the most up-to-date, complete and consistent information available. The reasons for this lack of information may be many. Hydroinformatics system is embedded in a dynamically changing environment. In addition to changes occurring in its 'physical' environment, some processes may appear or cease to exist within this society of hydroinformatics agents. This may happen due to the normal scheduling of these processes, but may as well arise due to failures of the hardware or software components within which the individual agents are implemented. In
Hydroinformatics as Emergent Phenomenon

certain conditions, the very heterogeneity of the agents can make communication between then difficult and so place an additional burden on the communication bandwidth, causing necessary information to arrive too late for its practical application - which, for all practical purposes comes down to the required information. Similarly, a selfishness' or 'competitiveness' among agents may result in a lack of willingness of agents to share information.

Hydroinformatics has to resolve these difficulties in transporting messages and it can in principle do so in a two different ways. The first of these may be identified as one of cooperative problem solving, where the cooperative behaviour is hard-coded in each individual agent's architecture. In the second approach, multi-agent systems cooperation emerges out of self-interest. The means to resolve of uncertainties (corresponding to inconsistent, incomplete and incorrect information) must constitute an integral part of a hydroinformatics language, rather than be introduced as some kind of a 'layer' superimposed upon the top of the system.

Moreover, agents function in an environment. The analysis and evaluation of an agents behaviour must always be expressed in terms of agent-environment couplings. The agents interact with their environment autonomously by utilising the environments sensors and effectors. The environment is also populated with other agents, and these form a constitutive part of it. This environment is often noisy and unpredictable. Most generally, agents are objects that communicate with one another.

![Figure 149 Schematisation of the transducer approach (adapted from Genesereth and Ketchpel, 1995)](image)

Hydroinformatics has another, even if related, problem: there are so many already-existing and potentially valuable software products that we would like to use by incorporating these within a hydroinformatics framework. The criterion for agenthood is however, a behavioural one. For such practical purposes, an object is only really a software agent to the extent it can communicate correctly with other agents in a certain agent communication language. But, then, of course, existing software is not in an agent-orientated form. Genesereth and Ketchpel (1995) propose three ways to 'agentify' what they call 'legacy' software. One approach is to
implement a *transducer* that mediates between existing programs and other agents. This can be done by introducing a translation process that interprets communication language's messages into messages in the legacy program's native language. This process can then just as well function in the opposite direction, so as to interpret the program's responses in the communication language. Such an approach has the advantages that it requires no knowledge about the program other than its communication behaviour.

A second approach is to implement a *wrapper*, whereby a program is injected into a structure that allows it to communicate with other agents. In this case, the wrapper is capable of examining and modifying the native program's data structures.

![Figure 150 Schematisation of the wrapper approach (adapted from Genesereth and Ketchpel, 1995)](image)

The third and most drastic approach to 'agentification' however, is to re-write the existing software so as to turn this into an agent. Although, this may evidently provide many advantages, it may well be costly process, and one that may only be viable at all to the extent that suitable tools, languages and other environments are available.

![Figure 151 Rewrite (adapted from Genesereth and Ketchpel, 1995)](image)

Individual agents are difficult to build, and sometimes they are just too difficult. The perspective on the individual agent can perhaps be summarised as by Nilsson and Rummelhart (1993). We use this schematisation to outline one last assertion of this work, which is that, since agents are indeed so very difficult to build *let the code evolve the agents instead!*
In a diagram of Figure 152, all sensory information enter into agent's model of the world through data assimilation processes. An action-computation module determines the most appropriate next action, based on the current state of the model - this is a simple reflexive portion of its behavioural model, and one that can be induced by the means of evolutionary programming. The planning sub-system is fired only if the required performance of agent can be realised in the time available; otherwise a simple reflexive action is favoured. (By way of comparison, see the architecture forwarded by Hayes-Roth, page 20). The learning component can then affect the modelling, the action-computation and the planning-mechanism sub-components, thus enhancing the flexibility of the agent. This learning process can be as well realised by introducing an artificial evolutionary process.

Learning not only allows for a more graceful degradation in performance of agents, but at the same time permits a completely different approach to be taken, namely one that is guided by the following principles:

- In the first place develop a simple, yet complete, agent architecture, and then elaborate upon so that it can handle increasingly sophisticated tasks in increasingly complex environments;
- evaluate each architecture by its ability to learn to achieve its goals;
• introduce learning mechanisms that are, above all, inductive;
• supply the agent initially with the least-restrictive kinds and amounts of knowledge;
• use the simplest possible set of architectural features, elaborating the architecture only when this becomes rather strictly necessary.

In such a way, we can not only expect to evolve effective agents, but also to let them learn continuously and improve their performances persistently. Moreover, we expect that we will have much to learn from the agents that are so evolved.
References


Angeline, P.J., 1993., *Evolutionary Algorithms and Emergent Intelligence*, Ph.D. thesis, Ohio State University, Columbus

Angeline, P.J., and Pollack, J.B., 1993., Evolutionary Module Acquisition, in *Proceedings of the Second Annual Conference on Evolutionary Programming*


Baas, N.A., 1994., Emergence, hierarchies, and hyperstructures, in Langton, C.G., (ed.), Artificial Life III, Addison-Wesley, Reading


Babović, V., 1993., Evolutionary algorithms as a theme in water resources, Scientific presentations AIO meeting '93, Delft University of Technology, pp. 21-36


Babović, V., 1995c., Evolutionary Algorithms and Their Application to Rainfall-Runoff Modelling, Advances in Water Resources, in press


Basye, K., Dean, T., and Kaelbling, L.P., 1995., Learning dynamics: system identification for perceptually challenged agents, Artificial Intelligence, Vol. 73, No. 1-2, pp. 139-172


Cavicchio, D.J., 1970., *Adaptive Search Using Simulated Evolution*, University of Michigan, Ann Arbor


Dawkins, R., 1982., The Extended Phenotype, Freeman


Deb, K., 1989., Genetic Algorithms in Multimodal Function Optimisation, (TCGA Report No. 89002 and Master of Science thesis), University of Alabama, Tuscaloosa


Deb, K., and Goldberg, D.E., 1991., mGA in C: A messy Genetic Algorithm in C (IlliGAL Report No. 92001), University of Illinois at Urbana-Champaign, Urbana


Fodor, J.A., 1975., The Language of Thought, Crowell, New York

References


Foucault, M., 1983., *This is Not a Pipe*, Univ. of California Press, Los Angeles


Hartigan, J., 1975., *Clustering Algorithms*, John Willey and Sons, New York


Harvey, I., 1992a., The SAGA cross: the mechanics of crossover for variable length genetic algorithms, in Manner, R., and Manderick, B., (eds.), in *Parallel Problem Solving from Nature 2*, Elsevier Amsterdam, pp. 269-278


Heidegger, M. 1977., The question concerning technology, and other essays, translation Lovitt W., Harper and Row, New York


Hopcroft, J.E., and Ullman, J.D., 1979., *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Reading


Holland, J.H., 1975., *Adaptation in Natural and Artificial Systems*, University of Michigan Press


Howe, A.E., and Cohen, P.R., Responding to environmental change, in *Proceedings DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*, Morgan Kaufmann, San Mateo, pp. 59-70


Korf, R.E., 1985a., Depth-first iterative deepening: An optimal admissible tree search, Artificial Intelligence, Vol. 27, pp. 97-109


Larsen, T., Frier, J.O., and Vestergraard, K., 1990., Discharge/stage relation in vegerated Danish stream, International Conference on River Flood Hydraulics


Li, M., and Vitanyi, P., 1993., *An Introduction to Kolmogorov Complexity and Its Applications*, Springer-Verlag


Mac Lane, S., 1971., *Categories for the Working Mathematician*, Springer-Verlag, New York


Mahfoud, S.W., 1992., *Crowding and preselection revisited*, IlliGAL Technical Report No. 91002, University of Illinois at Urbana-Champaign, Urbana, available through anonymous ftp as gal4.ge.uiuc.edu:/pub/papers/IlliGALs/


Mathias, K. and Whitley, D., 1992., Genetic operators, the fitness landscape and the travelling salesman problem, in Manner, R., and Manderick, B., (eds.), in *Parallel Problem Solving from Nature 2*, Elsevier Amsterdam, pp. 219-228


McCorduck, 1979., *Machines Who Think*, Freeman, New York


Newell, A., 1982., The knowledge level, Artificial Intelligence, Vol. 18., pp. 87-127


Nilsson, N, and Rumelhart, D., 1993., Approaches to Artificial Intelligence - A report of a workshop held at Santa Fe Institute, available through anonymous ftp as ftp.santafe.edu:/pub/Users/mm/approaches/approaches.ps


Ohno, S., 1971., Evolution by Gene Duplication, Springer Verlag, New York


Palmer, C.C., 1994., An Approach to a Problem in Network Design Using Genetic Algorithms, Polytechnic University, Yorktown Heights

Papadimitriou, C.H., 1995., Computational Complexity, Addison-Wesley, Reading


Tsujimoto, T., Okada, T., and Omata, A., 1993., Field measurements of turbulent flow over vegetation on flood plain of river Kakehaki, KHL Progressive Report, Hydraulic Laboratory, Kanazawa University


Whitehead, A.N., 1911., An Introduction to Mathematics


Wilson, S.W., 1995., Classifier system based on accuracy, *Evolutionary Computation*, in press


Wright S., 1932., The roles of mutation, inbreeding, crossbreeding and selection in evolution, in *Proceedings of the Sixth International Congress of Genetics*, pp. 356-366


Wilson, S.W., 1995., Classifier fitness based on accuracy, submitted to *Evolutionary Computation*


Summary

Following a brief survey in Chapter 1 of the emerging hydroinformatics paradigm, the notion of computational agency, as a societal perspective on computation, is introduced in Chapter 2 and an argument is advanced in favour of corresponding models. Natural intelligence, the realisation of which within artificial media has long been seen as the ultimate goal of computational science, and which is traditionally modelled as a centralised process, is depicted as a decentralised and distributed process in Chapter 3. By these means, a more nature- and socially-orientated approach to modelling intelligent behaviour is elaborated and analyzed. Further, in Chapter 4, the most interesting properties of such a societal, or distributed paradigm are identified as emergent. It is further argued that hydroinformatics itself can be best perceived as a genesis, as a set of emergent properties of its interacting components.

Chapters 5 and 6 are more specifically orientated towards methodologies for individual agent design. Chapter 5 describes some more traditional search techniques, whereas Chapter 6 addresses the issues related to computational models of evolution, exemplified by a rather broad category of Evolutionary Algorithms (EAs). It is argued that the interaction of a relatively few, simple agents among each other and with their common environment results in the induction of models that are capable of intelligent performance. It is even argued that as this technology comes to maturity over the coming decade, practically all our programming work will be done with the assistance of such processes of evolution of artificial agents. It is argued even further that, because of the interaction of evolving agents with their natural and social environments, they come to incorporate parts of these environment in the models that they provide, *grounding them in these environments*, and thus preventing us from building-in our own pre-conceptions, our own, often one-sided, perceptions and our own, and again often restricted, systems of values. Thus, it is asserted that only agents that are designed in such a way that they are properly grounded in their physical and social environment can be really considered ethical, and these constitute the only category truly serving the many and varied users of hydroinformatics systems.

Chapter 7, directs the focus of interest to somewhat more complex agents, the modelling of which has been done traditionally using ordinary and partial differential equation, or in a manner that is described throughout this work as *top-down*. The work concludes with a discussion of the architectural issues of an ‘ideal’ hydroinformatics system.
Vita

Vladan Babovic was born on December 21, 1961. Graduated at the Faculty for Civil Engineering at the University of Sarajevo (major in Hydraulic Engineering) in 1986. In 1990-1991 attended postgraduate course on *Computational and Experimental Hydraulics* at the International Institute for Infrastructural, Hydraulic and Environmental Engineering (IHE) in Delft, The Netherlands and obtained the Diploma of Hydraulic Engineering (Dip HE Delft) with distinction. In April 1991 obtained Master of Science degree with distinction at IHE Delft on the basis of work carried out at the Danish Hydraulic Institute (DHI) with a dissertation entitled *Applied Hydroinformatics: A control and advisory system for real-time applications*.

From May 1991 until July 1996 worked as a Research Fellow at the Hydroinformatics group of IHE-Delft.

His primary interests are Artificial Intelligence, Artificial Life and Evolutionary Algorithms and their application in hydroinformatics and water resources in general.

List of publications


Emergence, Evolution, Intelligence; Hydroinformatics


Babovic, V., 1996, Can water resources management benefit from artificial intelligence?, in *Proceedings of the 26th Internationales Wasserbau Symposium*, Aachen, Germany


Babovic, V., 1996., Some experiences with individual-based approaches to ecological modelling, in *Proceedings of the Ecological Summit '96*, Copenhagen, Denmark

Babovic, V., and Baretta, J. 1996., Individual-based modelling of aquatic plant populations, in *Proceedings of the Second International Conference on Hydroinformatics*, Zürich, Switzerland

Books and Proceedings


The aim of the International Institute for Infrastructural, Hydraulic and Environmental Engineering, IHE, is to transfer scientific knowledge and technological know-how related to transport, water and the environment to professionals, especially from developing countries.

IHE organizes regular one-year postgraduate courses which lead to either an MSc degree or an IHE diploma. IHE also has a PhD-programme based on a research, that can be executed partly in the home country. Moreover IHE organizes short tailor-made and regular non-degree courses in The Netherlands as well as abroad and takes part in projects in various countries to develop local training and research facilities.

International Institute for Infrastructural, Hydraulic and Environmental Engineering

P.O. Box 3015
2601 DA Delft
The Netherlands

Tel.: +31 15 2151715
Fax: +31 15 2122921
E-mail: ihe @ ihe.nl
The computer-controlled operating environments of such facilities as automated factories, nuclear power plants, telecommunication centres and space stations are continually becoming more complex. The situation is similar, if not even more apparent and urgent, in the case of water. Water is not only mankind's most valuable natural resource, but one which is in increasingly limited supply. The fresh water is the vital natural resource which supports all environmental activities, that is, natural economy, and all human socio-economic activities, that is, the artificial economy. The pressure for a sustainable control and exploration of water and thus for the peaceful co-existence of human- and hydro-economies, is not only a human, socio-economic pressure, but it is the question of life and death!

**Hydroinformatics** - the nascent technology concerned with the flow of information related to the flow of fluids and all that they convey - is probably the best possible answer yet proposed to the problem of the control of the waters, the very arteries and veins of the biosphere.

This work addresses some of the central issues within hydroinformatics paradigm. It focuses on the analysis of decentralised and distributed computation, as well as the issues of design of individual computational agents using evolutionary algorithms.