Outcome novelty in Exploratory Modelling and Analysis

A research into the value of novelty search for exploratory modelling

by

Martijn Ras

June 24, 2021

2020/2021 Q4

Master of Science
in **Engineering and Policy Analysis**

Faculty of Technology, Policy and Management

student number: 5189268

Thesis committee:

Dr. ir. J.H. Kwakkel,	TU Delft, Chairperson
Dr. J. Zatarain Salazar,	TU Delft, First supervisor
Prof. Dr. M.E. Warnier,	TU Delft, Second supervisor

TU Delft

Outcome novelty in Exploratory Modelling and Analysis

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Engineering and Policy Analysis

by

Martijn Ras

June 2021

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

Associated code and models are available at
https://github.com/M4rtijnR4s/NoveltySearchforEMA.

or models only at

https://github.com/quaquel/EMAworkbench.

Executive Summary

Many grand challenges, also known as societal, environmental or technical challenges with no clear answer, can be characterized as wicked problems. These are problems where one clear (optimal) solution does not exist. This is mostly due to multi-actor complexity ,where actors have different perceptions, but can also be because of the nature of the problem. Examples of wicked problems include climate change, poverty, terrorism and even the war on drugs. To tackle these problems, as best as possible, the use of computational experiments is required, this comes down to computer-aided modelling and simulation. Exploratory modelling is based on the modelling for decision support framework, which requires computational experiments. As the name might already suggest, exploration is a key concept of exploratory modelling.

Exploratory modelling is based upon the XLRM-framework. This combines the exogenous uncertainties (X) and policy Levers (L) using the Relationships (R) to gather relevant measures (M). Exploratory modelling uses sampling algorithms or optimization algorithms to find potential states of the future. These states of the future can be considered as knowledge during the decision making process. Sampling algorithms are used to identify potential states of the future to increase the understanding of the modelled system. Optimization algorithms are used to identify specific optimal solutions to the problem at hand. Although both can be considered to explore the model, it is clear that sampling algorithms are used for gaining an understanding of the system, this can be considered normal exploration. Conversely optimization algorithms focus only on very specific area's and imply that the user already has some knowledge of the system.

Within the lake problem latin hypercube sampling is the preferred sampler for model exploration. Nonetheless, novelty search presents other qualities for model exploration. Novelty search uses evolutionary algorithms in combination with novelty functions, fitness function without a tangible, static and predefined goal, but rather a goal that focuses on new results (novelty). Where exploration is about the discovery of new information, novelty search should be suitable for model exploration. Quality diversity algorithms exist to find a diverse set of optimal solutions, and are based upon novelty search. Such algorithms would belong to the optimization algorithms.

Novelty search has been applied in robotics, or neural networks, and mazes. Nonetheless novelty search does not exist for exploratory modelling. The most important part of novelty search is the novelty function, because evolutionary algorithms are already present. Defining a novelty function requires the definition of novelty in the given scenario. While different models might share the same novelty function or definition of novelties, neural networks and mazes do not. Novelty in a maze entails every place left that is not visited, whereas for models this only holds theoretically. Novelty has been defined as the place furthest away from the crowd. This has some downsides, such as leaving novel solutions that are not furthest away, as well as potentially finding contradicting novelties. These have mostly been resolved at the cost of run-time. This is an initial novelty function, with the knowledge from this paper a better novelty function could be created.

Finding the added value of novelty search for exploratory modelling, and thus comparing novelty search against latin hypercube sampling, requires some experimentation. These experiments will be based upon 5 diverse solutions, or policies. For the first two policies basic novelty search will

be compared against latin hypercube sampling. Each algorithm will run one million scenarios or functional evaluations, where the evolutionary algorithm for novelty search uses the default outcome goals. Secondly, the influence of the outcome goals will be explored in the experiments. For the third and fourth policy, the experiment will be performed with one million functional evaluations. One run maximizes all goals and the other minimizes all goals. Finally, the experiment aims to see how different number of functional evaluations impact the results.

The results from the experiments reflect the influence of the outcome goals from the evolutionary algorithm. Therefore showing characteristics of a quality diversity algorithm, which uses novelty search. With a novelty search implementation requiring more computations, the relative efficiency will be less compared to a standard evolutionary algorithm. Comparing this to latin hypercube, shows that in equal situations latin hypercube sampling is faster. While latin hypercube sampling has one million results, novelty search only returns about 25 results. With these 25 results novelty search is able to find novelties, or outcomes that had not been discovered before, and depending on the outcome goals even comparable range with a better distribution. When lowering the number of functional evaluations the results move closer to the crowd, showing that more runs are favorable for novelty.

This study shows that novelty search is able to discover outcomes that would otherwise be unattainable, showing that it has potential to replace latin hypercube sampling with an adjusted evolutionary algorithm and an improved novelty function. In the current state novelty search adds value as an addition to latin hypercube sampling, since it does find novel solutions. Latin hypercube sampling discovers most of the solutions from the output space, whereas novelty search only samples a small portion of novel solutions, that might not be representative of the real output space. .

Acknowledgements

I would like to properly show my gratitude to my committee - Dr. ir. Jan Kwakkel, Prof. Dr. Martijn Warnier, and Dr. Jazmin Zatarain Salazar - for their support and everlasting patience. If it had not been for them some parts of this research might have taken a lot longer. Finally I would like to thank Sahiti Sarva for continuously putting up with my, sometimes extremely, dumb questions.

Contents

1 Research Problem 3
1.1 Introduction . . . 3
1.1.1 Exploratory Modelling and Analysis . . . 3
1.1.2 Exploratory modelling . . . 4
1.1.3 Novelty search . . . 4
1.2 Research gap . . . 4
1.3 Research questions . . . 5
1.4 Research approach . . . 5
1.4.1 The lake problem . . . 6
1.5 Research relevance . . . 7
1.6 Chapter summary . . . 7

2 Literature Review 8
2.1 Evolutionary and Genetic Algorithms . . . 8
2.1.1 NEAT . . . 9
2.1.2 NSGA-II . . . 10
2.1.3 Epsilon NSGA-II . . . 10
2.1.4 BORG . . . 10
2.2 Novelty Search . . . 11
2.2.1 Introduction . . . 11
2.2.2 Novelty search compared to standard EA . . . 11
2.2.3 Novelty search in code . . . 11
2.2.4 Comparison of algorithms, implementations and extensions . . . 12
2.2.5 Difference simulation models and neural networks . . . 12
2.3 Exploratory Modelling . . . 13

3 Novelty Search 15
3.1 Novelty . . . 15
3.1.1 Maze . . . 15
3.1.2 Robotics . . . 16
3.2 Implementation . . . 16
3.2.1 The evolutionary algorithm . . . 16
3.2.2 The novelty function . . . 17
3.2.3 Earlier developments . . . 17

4 Experimental Setup 19
4.1 The experiment . . . 19
4.1.1 Optimal policies . . . 20
4.1.2 Exploration . . . 20
4.2 Results . . . 20
4.2.1 Optimal policies . . . 20
4.2.2 Exploration . . . 21

5 Results 22
5.1 Optimization results . . . 22
5.1.1 Conclusion . . . 25
5.2 Model exploration: The Lake problem . . . 26
5.2.1 Experiment 1 . . . 26
5.2.2 Experiment 2 . . . 28
5.2.3 Experiment 3 . . . 30

6 Discussion and Reflection 33
6.1 Optimization . . . 33
6.2 Novelty search . . . 34
6.2.1 Novelty function . . . 34
6.2.2 Experiments and results . . . 34
6.3 Assumptions . . . 36
6.4 Challenges . . . 36
6.5 Chapter summary . . . 37

7 Conclusion 38
7.1 Research questions and their conclusions . . . 38
7.2 Societal and scientific relevance . . . 41
7.3 Future work . . . 42

List of Figures 46

List of Tables 48

A Software and Packages 49
A.1 Tools . . . 49
A.2 Python Packages . . . 49

B Policy Results 50
B.1 Per Policy Exploration Results . . . 50
B.1.1 Policy 1 . . . 50
B.1.2 Policy 2 . . . 54
B.1.3 Policy 3 . . . 57
B.1.4 Policy 4 . . . 62
B.1.5 Policy 5 . . . 67

1
Research Problem

1.1. Introduction

Decision making can be extremely difficult, especially for wicked problems, societal challenges or problems that deal with deep uncertainty. Deep uncertainty focuses on inherent and multi-dimensional uncertainty. Modelling for decision support can be used to aid the decision makers to quantify the real world situation and explore potential states of the world (Davis et al., 2007; Kwakkel, 2017). One of the frameworks that can be used for this is Exploratory Modelling and analysis.

Modelling for decision support focuses on creating models for problems with deep uncertainty, such as wicked problems (Jobst and Meinel, 2014; Ritchey, 2013). These problems do not have one correct or best solution. This can be partially explained due to the different perspectives amongst stakeholders, resulting in multi-actor complexity (Ritchey, 2013). Although reproducibility is an important aspect of modelling for decision support, it also allows for testing of multiple hypotheses as well as problem formulations. Modelling for decision support allows users to easily expedite the results from different configurations. This shows that the contribution is not solely on reproducibility and replication, but on other levels as well.

With computational experiments being reproducible, modelling for decision support becomes even more useful. The experiments and results can be shared with other stakeholders to try and convince them of your opinion. The results can be replicated and so they will also be verifiable.

The deep uncertainty that is discussed in this paper follows the definitions from Lempert (2003) and Moallemi et al. (2020). Focusing on inherent and multi-dimensional uncertainty in which the uncertainties cannot reasonably be given a probability (Lempert, 2003; Moallemi et al., 2020). This can be categorized into two different types. Firstly, the uncertainties that are presently unknown, and secondly the uncertainties that no shared understanding can be agreed upon. The latter has a direct relation with multi-actor complexity.

1.1.1. Exploratory Modelling and Analysis

Modelling for decision support is based on computational experiments. The experiments can be performed using various methods that have been developed specifically for modelling for decision support. These methods include (multi-objective) robust decision making (MORDM or RDM) (Groves and Lempert, 2007; Kasprzyk et al., 2013) and dynamic adaptive policy pathways (Kwakkel et al., 2016). The EMA workbench is best described as a tool to explore and analyze deeply uncertain systems, or in other words, used for decision making under deep uncertainty (Kwakkel, 2017; Kwakkel

and Pruyt, 2013).

The EMA workbench is a functional implementation of the XLRM model framework (Kwakkel, 2017; Lempert, 2003). The XLRM framework structures system models, problems, as external input X, policy levers L, causal system relations R (which can be a black box) and metrics of interest M. This framework allows models to be defined as a mathematical function: $M = f_R(X, L)$. In this function M can be seen as output, X,L are the input and R is the model or function. The inputs of the XLRM model are mostly scalar values and the outputs are either scalar values or a set of time series values. The causal system relation R, the function itself, can be expressed in any modelling paradigm such as agent based modelling, system dynamics or discrete event simulation. The representation of the XLRM framework allows for the application of many mathematical questions to decision based questions. Thus making it perfect for model based decision support.

1.1.2. Exploratory modelling

Exploratory modelling aims to design policies that support valid conclusions or reliable insights based on a limited number of computational experiments (Bankes et al., 2013). The most common method is sampling the input space, using sampling algorithms such as Latin Hypercube and Monte Carlo (Kwakkel, 2017). Exploratory modelling is mainly used for exploration of a model, the goal of the exploration can be different each time. Using an optimization algorithm to explore a model, leads to a biased understanding towards some evaluation function. For example, the evaluation function could aim to find the "best" solution. This bias can thus be seen as a wanted and predicted bias. This shows that exploratory modelling, is not just something that should be used in the beginning of a project, to gather an understanding of the system, but can be used throughout the entire project, for finding optimal solutions.

The main goal of model exploration is to gain insights into the model. Particularly, the goal is to learn something about the system that the model represents. Model exploration could be done in the initial stage of a project, to gain an understanding of the system, but can also be used to find relevant states of the world. Model exploration can even be used to find the best possible solutions. The use-case that this thesis will focus on is exploring over a single, explicitly identified, policy. This requires the lever variables to have a combination of values, in which the state of world will be explored.

1.1.3. Novelty search

Novelty search is a state-of-the-art evolutionary approach that focuses on behavioural novelty, a technique that explicitly rewards diverging, as opposed to pursuing static objectives (Gomes et al., 2015; Lehman and Stanley, 2011b). Novelty search is relatively new, and has only seen real applications in neural networks and it has not yet been tested for exploratory modelling.

1.2. Research gap

Current research and implementations of novelty search focus on neural networks. Further research is required into the inner workings of novelty search, and requirements of novelty search for model exploration. Once this information is available a (dirty) novelty search implementation could be made to test novelty search for model exploration. Although prior research, focusing on neural networks, suggests exploration should be improved, this cannot be directly translated to the exploratory modelling. If novelty search can be implemented for exploratory modelling, the question is what would entail a benchmark for comparison? and how should performance be measured?

1.3. Research questions

Assuming that model exploration will be able to use novelty search, the main research question to be answered in this paper is:

> (M) What is the added value of novelty search for Exploratory modeling and analysis?

To answer the main research question, four sub-questions have been defined. The first two sub-questions are answered through a literature review. The last two sub-questions will need a combination of literature review, code development and validation. These sub-questions will focus on the implementation and comparison of novelty search against current algorithms. To answer the third and fourth sub-questions, we take an approach into defining novelty for the lake-problem. The guiding sub-questions are:

> (1) What are the key concepts, features, algorithms of novelty search and how do they work?

> (2) What are the relevant search strategies in exploratory modeling and analysis and what are the relevant features to compare the different algorithms?

Once the theoretical part is finished, a more practical part follows. This part focuses on the placement and implementation of novelty search:

> (3) Where would novelty search fit in exploratory modeling and analysis?

Having a functional implementation of novelty search, a comparison can be made between existing strategies and novelty search:

> (4) How does novelty search compare to the current algorithms in some model(s), under different circumstances?

1.4. Research approach

This thesis consists of 7 chapters. The first chapter is dedicated to the introduction of the research problem and the research questions that guide this thesis. The introduction contains a small literature review, and a extensive review on the model that is used throughout this case-study. The case-study is focused around the lake problem, which is explained in section 1.4, after this paragraph.The following chapter is a complete literature review on novelty search, evolutionary algorithms and exploratory modelling. The second chapter focuses solely on novelty search, its implementation (the third sub-question). The third chapter aims to finalize the answer of the second and third sub-question. After having executed some experiments the fourth chapter will focus on the setup of these experiments and a fifth and sixth chapters will answer the fourth sub-question. Finally a chapter will conclude the research, answering all questions, discussing the methods, reflecting the work and promoting topics for future research.

In the experimentation phase the Direct Policy Search (DPS) version of the lake problem by Quinn et al. (2017) will be used. The lake problem is an established benchmark problem, useful for comparison. The lake problem is initially used to develop a suitable novelty search implementation, subsequently it will also be used for the experiments. The difference between initial testing and the experiments will be the number of functional evaluations, and the depth of the analysis.

1.4.1. The lake problem

The lake problem was originally developed by Carpenter et al. (1999) to balance the economic benefits of discharging phosphorus into a lake with the environmental cost of irreversibly tipping the lake in an eutrophic state (Quinn et al., 2017). The bahaviour of this model is representative for many socio-ecological systems with tipping points. By applying a closed loop control strategy, called Direct Policy Search (DPS), the lake problem could be solved using Many objective evolutionary algorithms (MOEA) (Quinn et al., 2017).

The current lake problem can be seen as a stylized and hypothetical decision problem where the population of a city has to decide on the amount of annual pollution it will put into a lake. Once the pollution in the lake passes a threshold, the lake suffers irreversible eutrophication. The following mathematical equation 1.1 captures this problem in its entirety.

$$X_{(t+1)} = X_t + a_t + \frac{(X_t^q)}{(1+X_t^q)} - bX_t + \epsilon_t \tag{1.1}$$

In equation 1.1 X_t is the pollution at time t, a_t is the rate of anthropogenic pollution at time t, b is the lake's natural removal rate, q is the lake's natural recycling rate, ϵ_t is the rate of natural pollution at time t. The rate of anthropogenic pollution a_t is the decision variable with values between 0 and 0.1. The natural pollution ϵ_t is modeled, following Singh et al. (2015), as a log normal distribution with mean μ and standard deviation σ.

The lake problem has four outcomes of interest. The first is the average concentration of phosphor in the lake, depicted in equation 1.2. The second is the economic benefit derived from polluting the lake, depicted in equation 1.3. The third is related to the year over year change in the anthropogenic pollution rate, depicted in equation 1.4. The fourth is the fraction of years where the pollution in the lake is below the critical threshold, depicted in equation 1.5.

$$f_{phosphorus} = \frac{1}{|T|} \sum_{t \in T} X_t \tag{1.2}$$

In equation 1.2 $|T|$ is the cardinality of the set of points in time.

$$f_{economic} = \sum_{t \in T} \alpha a_t \delta^t \tag{1.3}$$

In equation 1.3 α is the utility derived from polluting and δ is the discount rate. By default, α is 0.04. Equation 1.3 is defined as the discounted benefit of pollution mines the costs of having a polluted lake, following the description of Singh et al. (2015).

$$f_{inertia} = \frac{1}{|T|-1} \sum_{t=1}^{|T|} I(|a_t - a_{t-1}| > \tau) \tag{1.4}$$

In equation 1.4 I is an indicator function that is 0 if the statement is false, and 1 if the statement is true, τ is the threshold that is deemed undesirable, and is for illustrative purposes set to 0.2. Effectively, $f_{inertia}$ is the fraction of years where the absolute value of the change in anthropogenic pollution is larger then τ.

$$f_{reliability} = \frac{1}{|T|} \sum_{t \in T} I(X_t < X_{crit}) \tag{1.5}$$

In equation 1.5 I is also an indicator function that is 0 if the statement is false, and 1 if the statement is true, X_{crit} is the critical threshold of pollution and is a function of both b and q.

The DPS lake problem is characterized by both stochastic and deep uncertainty, where the stochastic uncertainty arises from the natural inflow. To account for this uncertainty, multiple replications are performed and the average over the replication is taken. Deep uncertainty is presented by uncertainty about the mean μ and standard deviation $sigma$ of the log-normal distribution, that characterizes the natural inflow, the natural removal rate of the lake β, the natural recycling rate of the lake q, and the discount rate δ. The table 1.1 specifies the ranges for the deeply uncertain factors, description and their base values as defined by Quinn et al. (2017).

Table 1.1: Deep uncertain parameters information

Parameter	Description	Base value	Range
μ	Mean natural P inflow	0.03	$[0.01, 0.05]$
σ^2	Real-space variance of natural P inflow	10^{-5}	$[(0.001)^2, (0.005)^2]$
σ	Root of real-space variance of natural P inflow	± 0.0017	$[0.001, 0.005]$
b	Linear P loss parameter	0.42	$[0.1, 0.45]$
q	Shape parameter of sigmoid curve modelling P recycling	2.0	$[2.0, 4.5]$
δ	Discount parameter for calculating eoncomic benefits	0.98	$[0.93, 0.99]$

1.5. Research relevance

This research effort has a significant relevance for the policy analysis field, specifically to the field of model-based decision support. The focus is on reducing computational time, or improving the understanding of the model. The main application for novelty search in this research will be exploration, but it could also improve optimized algorithms. The results of this research may be applicable in future research efforts and real-life applications.

1.6. Chapter summary

Model-based decision support is a term for a variety of computational approaches using computer models. An approach of model-based decision making support is model exploration. There are many use-cases for this exploration, but the aim will be on single policy exploration. The goal is to learn something about the potential states of the world that might occur, thus learning something about the relation between input and output. Current explorations are either done using a latin hypercube sampling, or optimizations. Optimizations use static objectives to reach a so called optimum, whereas latin hypercube just samples the input space. Therefore leaving a gap for exploration, where the focus is on gathering new information as opposed to objective oriented information. This research gap is the topic of this thesis, together with the earlier defined exploration use-case.

2

Literature Review

This chapter provides an overview of the literature. The key points will be on novelty search and similar algorithms for novelty search, which is the central topic of this paper. Since novelty search requires an Evolutionary or Genetic algorithm, this chapter will start with an overview of evolutionary and genetic algorithms, how they work, what they do and what are important algorithms from this category. Subsequently, information regarding novelty search will be given, explaining the theoretical differences between current evolutionary algorithms and novelty search, as well as some comparisons amongst different novelty search algorithms and implementations. A important topic in this section is also the way to define novelty, which turns out to be very difficult. Finally, an in depth explanation on exploratory modelling will be presented.

2.1. Evolutionary and Genetic Algorithms

The basis of any optimization algorithm, including novelty search, is an evolutionary or genetic algorithm (EA or GA) (Lehman and Stanley, 2008; Najim et al., 2004; Yusoff et al., 2011). An evolutionary algorithm is an efficient heuristic search method. Evolutionary algorithms are optimization algorithms mimicking the biological mechanisms (Dufour and Neves, 2019).

The difference between genetic and evolutionary algorithms is in the usage of the problem. Decision variables and other variables and functions defined within the problem are directly accessible in an evolutionary algorithms. In genetic algorithms these are encoded in bit strings (Roetzel et al., 2020).

Evolutionary algorithms use a fitness function to determine the fittest individual (Galvan et al., 2003; Najim et al., 2004). A fitness function is an evaluation function that evaluates all individuals according to the same standards (Jin et al., 2002). A fitness function consists of one or more objectives transformed into combined into a single function that results into single scalar outcome (Murata and Ishibuchi, 1995). They can be as simple as a linear comparison function and can be made as difficult as desired, containing multiple functions that are normalized into a single value(Harman and Clark, 2004).

The most relevant applications for evolutionary algorithms are for optimization, search, and exploration. The applications are not restricted to a single field of interest, but have already proven useful in many different fields (Slowik and Kwasnicka, 2020).

The basis of each evolutionary algorithm consists of selection, crossover, mutation and evaluation. In algorithm 1 the pseudo-code for a basic evolutionary algorithm is given, as defined by

Eiben and Smith (2015). The elements of the pseudo-code are the bare minimum, each and every evolutionary algorithm will have this, however it does not mean it is the same for every algorithm. There are also possibilities of more elaborate evolutionary algorithms.

Algorithm 1 Evolutionary algorithm basis

START
Initialise population with random candidate solutions
Evaluate each candidate
for do REPEAT:
  Select parents
  Recombine pairs of parents
  Mutate the resulting offspring
  Evaluate new candidates
  Select individuals for the next generation
  UNTIL some termination condition is satisfied
STOP

Due to the many applications for this type of algorithm, many different algorithms and implementations have been developed. The most frequently used groups of algorithms from Slowik and Kwasnicka (2020) are:

- Genetic algorithms (GA),
- Genetic programming (GP),
- Differential evolution (DE),
- Evolution strategy (ES),
- Evolutionary programming (EP).

For novelty search, there a few algorithms that are expected to work well. The reason is either, because they have already proven to work exceptionally well in other situation, or because they have properties similar to those that have proven to work well (Gomes et al., 2015). A small selection of algorithms that are expected to work especially well are:

- NSGA-II
- Epsilon NSGA-II
- NEAT
- Borg

2.1.1. NEAT

NEAT is one of the best underlying algorithms for novelty search according to Lehman et al. (2013). NEAT stands for NeuroEvolution of Augmenting Topologies (Stanley and Miikkulainen, 2002). NEAT is an algorithm that focuses on neuroevolution, as can be deduced from the name. Neuroevolution is the artificial evolution of neural networks using genetic algorithms, therefore NEAT can be considered a GA (Stanley and Miikkulainen, 2002). The main ideas of NEAT are: Genetic Encoding with Historical Markings, Speciation and Minimizing Dimensionality (Stanley and Miikkulainen, 2002; TIBERMACINE and DJEDI, 2014).

Since NEAT focuses primarily on neuroevolution, it is not considered useful for exploratory modeling. Models are inherently different from neural networks, therefore it is not feasible to use NEAT without completely altering the algorithm. Due to this difference in requirements it is not relevant to put more effort into NEAT. It is important to know, that it might be possible to use NEAT, but it will require some tweaking, and changing of the algorithm to make it work with specific models.

2.1.2. NSGA-II

NSGA-II closely follows NEAT as one of the best algorithms for novelty search, according to Lehman et al. (2013). NSGA-II stands for Non-dominated Sorting Genetic Algorithm (Deb et al., 2002). Therefore it is clear that it belongs to the group of genetic algorithms. NSGA-II is a MOEA algorithm that uses non-dominated sorting (Deb et al., 2002). The main ideas that make NSGA(-II) special are non-dominated sorting, which should be improved in NSGA-II, and diversity preservation.

2.1.3. Epsilon NSGA-II

Epsilon-NSGA-II or $\epsilon-$NSGA-II is a combination of epsilon-dominance archiving and the parent algorithm NSGA-II (Ferringer et al., 2009; Kollat and Reed, 2005). The key features that it significantly differs from the parent algorithm are epsilon-dominance archiving, auto-adaptive population sizing and the use of time continuation (Ferringer et al., 2009). The idea behind epsilon-dominance is that it does not require full dominance over a certain value, but can be partly dominant. The goal of this is to have epsilon dominance MOEA's reach both the convergence and diversity (Laumanns et al., 2002).

2.1.4. BORG

Borg is another MOEA that uses epsilon dominance. . Borg uses epsilon dominance archiving with auto-adaptive operators (Hadka and Reed, 2013). The algorithm detects search stagnation and automatically adjusts the operators accordingly. Furthermore it exploits randomized restart, to escape local optima, and selects recombination operators based on their success (Hadka and Reed, 2013).

2.2. Novelty Search

2.2.1. Introduction

Novelty search is a new school of thought for evolutionary and genetic algorithms where the focus is instead placed on what is new (novel) (Gomes et al., 2015; Lehman and Stanley, 2008; Risi et al., 2009). Where an evolutionary algorithm normally works using a fitness function, for novelty search the fitness function is replaced with a novelty function (Lehman and Stanley, 2011a). A requirement in any novelty search algorithm is a way to keep track of the encountered novelties, where an archive can be used to track novel solutions and update it with better solutions.

Novelty search consists of 4 important building blocks, according to Gomes et al. (2015). First of all the underlying evolutionary algorithm. Each EA impacts the outcomes of novelty search differently as shown in Gomes et al. (2015). The EA are the basis of all novelty search algorithms and implementations (Lehman and Stanley, 2008). Another important aspect of novelty search is the novelty function. The novelty function is a mathematical translation of the novelty definition used in the subject of interest (Cuccu and Gomez, 2011; Lehman and Stanley, 2008, 2011a). It should answer the question: "What is novelty?". Subsequently there is an archive that keeps track of all novel individuals, and finally an optional evaluation function that combines a novelty and a fitness function.

2.2.2. Novelty search compared to standard EA

A major difference between normal evolutionary algorithms and novelty search is the action after the evaluation of the individuals according to the novelty or fitness function. Once a solution comes closer to a goal, that needs to be exhausted even more to actually find that goal in a normal EA. For novelty search this does not hold. Once a goal is (novelty) is found the space has been explored and the algorithm will move away from this. An overly simplified reason for this is that the static objectives from the fitness function only lead to a few solutions, whereas the dynamic objectives from the fitness function should yield a dispersed set of solutions.

Another difference between a novelty search algorithm and a evolutionary algorithm is in the novelty function and the fitness function. The goal of a fitness function is to find some goal, or gene for genetic algorithms (Eiben and Smith, 2015; Galvan et al., 2003). These goals are pre-specified (static) objectives. Individuals coming close to these objectives are rewarded. Novelty functions reward individuals that are far from others, or are very different from the average. Novelty functions aim to find novelty, or new outcomes. This urge for exploration can be seen as a goal as well, albeit a more dynamic goal. This shows both fitness and novelty functions looking for goals in their system (Harman and Clark, 2004). Thus both functions are guiding the search into a specific direction, while one tries to keep it close, the other tends to go further away.

2.2.3. Novelty search in code

Algorithm 2 shows the pseudo-code for novelty search as supported by (Cuccu and Gomez, 2011; Lehman and Stanley, 2008). The termination condition can be something as simple as a number of function evaluations or as hard as determining diminishing novelty. Evaluate should refer to the evaluation of the input variables and thus execution of the individual as well as the evaluation against the novelty function. The novelty function depends on the situation, therefore it cannot be presented in a general pseudo-code for past functions.

Algorithm 2 Novelty search algorithm

START
Initialise population with random candidate solutions
Evaluate each candidate
for doREPEAT:
    Select parents
    Recombine pairs of parents
    Mutate the resulting offspring
    Evaluate new candidates
    Determine winners and losers and add to archive
    UNTIL some termination condition is satisfied
STOP

2.2.4. Comparison of algorithms, implementations and extensions

From Gomes et al. (2015) and Lehman et al. (2013) it becomes clear that the best underlying algorithm is NEAT followed closely by NSGA-II. These papers focus on novelty search in neural networks and mazes, therefore it is not directly translatable to simulation models. These papers represent a good start to learn more about novelty search, due to the limited research into other fields.

The literature shows that novelty search is great for exploration (Gomes et al., 2015; Lehman and Stanley, 2011c; Risi et al., 2009; Velez and Clune, 2014). Gomes et al. (2015) shows best practices, and pre-defined optimal variables (for maze exploration). Even though the author notes that this basis is not the same for all fields and applications, it does provide a good starting point. The major flaw in most of the literature is that there is no proper reasoning as to why something works best. The author provides a practical experiment and deduces the best practices, by interpreting the results. However, these studies do not provide a reasoning for the algorithmic behaviour.

As can be seen from the differences in NEAT and NSGA-II, NEAT is doing most things a bit different. For example, having a small difference in focus of dimensions. This is most likely the reason that NEAT works better in neural network optimization and mazes. Since NEAT is not available as evolutionary algorithm for simulation models, another choice needs to be made on the evolutionary algorithm. Algorithms such as EpsNSGA-II and BORG have more options and might be able to replicate NEAT, or show similar characteristics (Deb et al., 2002; Hadka and Reed, 2013).

Novelty search is part of many quality diversity algorithms, within the evolutionary computation algorithms (Pugh et al., 2015). Quality diversity algorithms are used to find a divers group of optimal solutions (Pugh et al., 2015). The reason for the use of novelty search in these algorithms is due to the exploratory character of novelty search. Basically, novelty search is well-suited to get a diverse set of solutions, requiring a mechanism to attain high quality solutions.

2.2.5. Difference simulation models and neural networks

When talking about neural networks in this paper we are talking about artificial neural networks. Artificial neural networks try to represent biological neural networks, where each node or artificial neuron is able to transmit a signal to the others (Hopfield, 1988; Jain et al., 1996). Neurons consist of some mathematical model, or in other words a function (Jain et al., 1996). Through training a neural network "learns" the weight for a connection between two neurons (Jain et al., 1996). This does imply that a training set exists, which consists of some input with a known output related to the input. In most cases the inner workings of a neural network can be considered to be a black-box.

A simulation model consists of several mathematical models all relating input to output. These models should be verified and validated. A simulation model does not require any training. Once verified and validated a model is ready to simulate the situation. It is important to realise, that models contain uncertainty and real world systems translations to a computer model. Therefore it should be taken into account that all models are wrong, only some can be said to be useful as quoted from Box (1976).

2.3. Exploratory Modelling

Exploratory modelling or exploratory modelling and analysis (EMA) is used to analyze complex and uncertain situations through the use of computational experiments (Bankes et al., 2013; Bankes, 1993; Kwakkel and Pruyt, 2013). Although the main focus of exploratory modelling is model-based decision making, the model-based foresight that is needed for this can also be quite useful in other situations. The focus is not only on optimizing a system to accomplish a goal or answer a certain question, but also to evaluate what-if situations (Kwakkel and Pruyt, 2013). An example of such a situation is: "when would a policy fail?" or "what would make a policy do well?". Therefore it can be said that EMA focuses on "out of the box" thinking as indicated by Kwakkel and Pruyt (2013). Another example from Bankes et al. (2013) shows how to shift the debate from the right model, to which policy would work for all models.

EMA will result into a large number of model runs, where the resulting outcomes must be analyzed and communicated to the decision makers and analysts (Bankes et al., 2013). Algorithms such as Latin hypercube sampling, Monte carlo sampling, Full factorial sampling, partial factorial sampling and a wrapper for various other sampling schemes, such as Sobol, Morris and FAST, are used for the exploratory part. Optimizations experiments are also available through various evolutionary algorithms, such as NSGA-II, EspNSGA-II, Generational Borg, SPEA and many more. When using the EMA-workbench it should be fairly simple to add new algorithms, or implementations of algorithms, to the workbench. Therefore enabling the addition of new methods such as novelty search.

These two search strategies that are used in exploratory modelling are called, open exploration and directed search. The sampling algorithms belong to open exploration and the optimization algorithms such as NSGA-II, Borg and SPEA are used in directed search. Open exploration systematically explore the plausible states of the future (Kwakkel and Pruyt, 2015). Directed search is used to find particular future states of the world that are of interest (Kwakkel and Pruyt, 2015). Although novelty search uses an evolutionary algorithm, it belongs to the open exploration strategies in this paper. Even though a quality diversity algorithm uses novelty search, and shows characteristics of novelty search, it belongs to the directed search strategies.

Latin Hypercube Sampling

Latin hypercube sampling contains many of the desirable features from random sampling and stratified sampling, and also producing more stable outcomes compared to random sampling (Helton and Davis, 2003). Latin hypercube sampling incorporates both no stratification, and stratified sampling (Helton and Davis, 2003; Stein, 1987). The estimates of latin hypercube, random sampling and stratified sampling are all unbiased, but latin hypercube is more stable than random sampling and easier to implement than stratified sampling Helton and Davis (2003).

From Kucherenko et al. (2015) latin hypercube has the edge over monte carlo, which is also available. However, explorations of the lake problem have been using latin hypercube sampling.

Since latin hypercube outperforms some of the other sampling methods, and it is used more

frequently, as default, for the lake problem, latin hypercube can be considered to be the base case. This would mean, latin hypercube will serve as a basis for the comparison of novelty search in exploratory modelling and analysis. Although latin hypercube might be the best available method it also has some downsides. Latin hypercube sampling does not excel at identifying stochastic uncertainty, extreme values are less likely to be the real extreme values and find other details (Helton and Davis, 2003).

3

Novelty Search

This chapter will go deeper into the implementation of novelty search. Explaining the implementation itself, which important decisions have been made, what assumptions have been made, what challenges occurred, the requirements and a special part about novelty. Since novelty search cannot go without novelty, an entire section will explain how this novelty was implemented, and why that was done this way. The implementation is based on the knowledge and algorithm as provided in chapter 2. However prior to the implementation a quick discussion on novelty is given.

3.1. Novelty

Novelty is something new, or something that has not been discovered yet. Looking at the definition in the Cambridge University Press (1995) dictionary it shows: "the quality of being new and unusual" or "something that has not been experienced before and so is interesting". Both of these definitions give a clear idea of what novelty search should search for. This section will elaborate more on novelty, giving examples of how it could be defined and finally an explanation of what has been decided upon in this research.

3.1.1. Maze

In a maze novelty can be easily defined. Every place that has not been reached yet is a new place, and considering the definition of novelty, that means it is novel as defined in Gomes et al. (2015). However the question is how should we reach this novel place. Given a maze, you have a start, an end and a set of moves. The set of moves consist of a combination of all moves that are taken, where the potential moves are up (forward), down (back), left and right. Depending on the maze, and the location in the maze some moves might be prohibited or unable to be made. Dividing the maze in a two-dimensional grid, gives an even better overview of the surroundings and thus what next step would be novel. For instance visiting 1,1 is easier to remember than translating a set of moves to another. An example of translating moves is: going left once, and right twice being equivalent to going forward once. This also holds for larger sequences with dummy steps in between. Therefore making it harder to determine a novel place, however sufficient computational power solves these issues.

In either case, using a grid or by translating sets of moves, it is quite simple to determine if a next step will be novel or not. A very clear and direct relation exists between input and output. Adding a move to the set of moves, means you do a step in that direction and enter a different part of the grid.

3.1.2. Robotics

In robotics it is already harder to define a novelty function. It depends on the goal of the "robots". This means there is not really a generic novelty function for these kind of situations. A frequent occurrence is using distance metrics for novelty functions. These distance metrics are not generic at all, because the values used in the distance metrics are different every time. For some robots there is need to learn how to walk, whereas for others the goal is to create clusters, find missing persons or walk through a maze.

It is clear that distance metrics are common, but the usage of the distance metrics differs in each situation. Although there is a functional relation between the input and output, it is unrealistic to call this a direct relation, because there are so many input variables and these functions are the model. The idea is to use the distance metrics as well. The idea to find optimal novelty is by trying to go as far a way from the crowd as possible. Therefore the distance metric that will be used is going to be a crowding distance.

Taking the individuals that are furthest away from the crowd should result in the most extreme results, in theory. The downside of this will be that in the worst case, combining two individuals with a high crowding distance will result in ending up in the crowd. By selecting one individual that is far away from the crowd and the other that is closer to that individual should resolve this issue of staying within the crowd. This does bring a new issue in the worst case, that is having two cliques with extreme values.

3.2. Implementation

The implementation is based on algorithm 2 by Cuccu and Gomez (2011) in chapter 2. Although this pseudo-code is not entirely sound, because it is not entirely clear what happens where. Therefore a implementation has been made so that it is based upon the algorithm, but not identical to that. We are using the EMA workbench, so the idea is to use as much as possible from the parts that are available.

Within the EMA workbench the implementations of the evolutionary algorithms and everything they require is pulled apart and divided over different functions. This makes it possible to adjust an algorithm and take the selector of another algorithm. This also means that these parts can be replaced or extended with self-developed parts. This makes it possible to create a whole new algorithm implementation, or only partially new implementation. The latter is what has been decided upon, as developing an entire new evolutionary algorithm is quite exhaustive for the developer.

3.2.1. The evolutionary algorithm

The evolutionary algorithm that will be used is not a fixed algorithm, although Borg is preferred and therefore set as default. This should make sure that novel optima are not exhausted, by adjusting the epsilon values or forcing a restart to focus on other areas. Pseudocode for the algorithm is given in algorithm 3. Most of this algorithm is from the platypus workbench, and has not seen any changes, the only change is in the selection of parents to use a novelty function.

Algorithm 3 Novelty search implementation

```
START
Initialise population with random candidate solutions
Evaluate each candidate
for Some Condition do REPEAT:
    Select most novel parents
    Recombine pairs of parents
    Mutate the resulting offspring
    Evaluate new offspring
    Add winning individuals to archive
    UNTIL some termination condition is satisfied
STOP
```

3.2.2. The novelty function

The novelty function is implemented in the tournament selector to select the most novel individuals. At first the crowding distance is calculated for all individuals. Then at first the furthest individual is selected. In the directly following selector at first a random individual is selected. Subsequently a distance is calculated to the previous individual and all individuals from the population and the closest to the previous winner is selected. A representation of this function can be found in algorithm 4. This is the final novelty function used in the implementation for the experiments.

Algorithm 4 Novelty function

```
START
Calculate crowding distance for population
Winner = Select random candidate from population
if First Parent then
    for Population do REPEAT:
        if Candidate crowding distance > Winner crowding distance then
            Winner = candidate
if Second parent then
    for Population do
        Calculate manhattan distance to first parent
        if candidate distance is smaller than winner and greater than 0 then
            Winner = Candidate
Return Winner
STOP
```

3.2.3. Earlier developments

Before novelty search, and the novelty function had been developed in their current form, other ideas had been researched, (partly) developed and evaluated. With the little research into novelty search for model exploration, the start was rough. Some research, such as Gomes et al. (2015), shows that different algorithms work better for novelty search. One of these algorithms is NEAT, which seemed very interesting. Therefore the earlier focus was on getting a good evolutionary algorithm. A few programs such as multi-NEAT had been installed to verify this, but it quickly became more and more work to get it working properly for exploratory modelling. Other implementations of NEAT were looked into, and none of them had what it would take. They all required a lot of bug fixing, or actually altering the algorithm, to get it to work with exploratory modelling

There was only one thing left to do, translating this NEAT algorithm to an algorithm for exploratory

modelling. A few ideas were present such as just remaking the whole algorithm or using a generalized setup to build parts of the algorithm. Building such an algorithm from scratch takes a lot of effort, and was not really possible, since NEAT is clearly focused on neural networks. So it would come down to making a whole new algorithm, since there are plenty of evolutionary algorithms available that is not desirable. Another reason for not progressing on this path is that, the evolutionary algorithms tend to impact novelty search, but the impact of the novelty function is bigger.

This information started a shift in focus, with evolutionary algorithms already present in exploratory modelling, all that is left is a novelty function. The first developed representation of a novelty function worked with nearest neighbours. Knowing your neighbours, a more novel individual can be selected. While this is true in theory, it does require a lot of extra computations. More importantly though, it still did not really solve the issue of getting a novel individual. You would be able to find a neighbour that is far away, but that is only in relation to a single individual. Finding a neighbour that is far from all individuals was much harder and much more computationally expensive. This transitioned into a nearest neighbour algorithm that only uses distances. This took a bit of time but as it turns out the Manhattan distance would work best for the current model. Although the Manhattan distance works best, it still was not optimal or something to be called good. The issues with these two novelty functions had to do with the focus on nearest neighbours. This made it harder to find novel solutions further away.

Since nearest neighbours was not showing potential, the focus started to shift to furthest neighbours. It sounds like a small step, but furthest neighbour algorithms do not really exist, it is usually nearest neighbour and then finding the largest distance. This is obviously inefficient, as it close to what we tried already. There is one solution that is interesting for the novelty function, that is the crowding distance. It took a bit of time to find, and to optimize, but this is what we also ended up with.

There are earlier usages of the crowding distance that differ from the current novelty function, and even other variations of the crowding distance are still possible. At first, the crowding distance was just looking for the the largest distance (infinity). The results were good, but not great. Looking back, there are two issues at hand here, either taking the same parent twice, or taking the complete opposite side. The first issue is not really bad, but having two different parents seems better, since you focus less on one novelty. The second issue can become a problem, as it might steer you into the crowd again. Which means you have come back to where you came from.

Solving the second issue was most important, so the earliest fix was only selecting one parent. This did yield better results, so there had to be another way to make sure that novelties in opposite directions were not combined as parents. There are multiple ways of doing this, and none are really efficient. With a nearest neighbour algorithm already developed, this would be useful. One parent is already known, and this is supposed to be the most novel individual, where similar novel individuals might exist, but not more novel. Therefore it would seem best to take the first parent as center and find its nearest neighbour. This can be tweaked some more, by selecting an arbitrary neighbour, however since we have a most novel individual, neighbour will be further away, therefore selecting the closest neighbour suffices.

4

Experimental Setup

This chapter will give an overview of the experimental setup. An explanation will be given on how the selection for models are made, as well as an overview of how the different search strategies will be applied and compared. One of the search strategies will be novelty search, as that is the focus of this paper. The other search strategy will be latin hypercube sampling, since that can be considered to be an established benchmark for the lake problem. The goal of this chapter is to give a clear overview of what is done to fairly compare different search strategies. All experiments will be using the cythonized DPS version of the lake problem model, which is elaborated in chapter 1.

4.1. The experiment

The experiment will consist of four different sub-experiments. The goal is to see if novelty search is a good addition for exploratory modelling and analysis. With novelty search focusing on exploration, a choice needs to be made as to what type of exploration will get the focus. This has already been defined as exploration for a specific policy, where no policy is also a policy. To get a decent amount of results 5 different policies of the lake problem will be used. Randomly selecting a policy seems arbitrary, and does not fit this research of a specific policy. Therefore the first experiment can be defined as finding optimal policies for the lake problem. That includes running 5 optimizations with a million functional evaluations and each a different seed, and after that selecting the five best policies using cliques.

With the 5 policies available the experiments using novelty search can begin. The 5 policies will serve as basis to run novelty search and latin hypercube sampling. The mean search strategies will be latin hypercube sampling and novelty search. Using the multiple policies different questions should be answered. How novelty search behaves under different policies, how novelty search compares against latin hypercube sampling and, how novelty search behaves under different parameters.

The experiments involving the search strategies will use at least 1000 functional evaluations and maximally 1000000 functional evaluations. The 1000 functional evaluations will only be used as comparison between the effectiveness of novelty search over other search strategies, if applicable. In general experiments with 100000 will be most common. These estimates will be explained more in-depth in the following sections explaining each experiment in detail. Since we do not know what the policies will be exactly, they will identified by a simple number 1 up to 5.

4.1.1. Optimal policies

The optimization experiment will consist of five different runs all with one million functional evaluations. The goal is to get five optimal and diverse results, which are sensible policies for further experiments. Each run will have a different seed for the random number generator that is used. The first will have no seed, so the default value, the others will be 1, 2, 3 and 4. The results of each run will be stored separately and combined into a single dataframe. Once all the results are in, the most diverse solutions will be identified and used as policies for further experiments. No extensive comparison or testing will be done on these results, the only goal is to get five diverse and sensible policies. The explanation on how the five policies are calculated can be found in section 4.2.1

4.1.2. Exploration

The exploration experiments will consist of three experiments, and will also use the lake problem. The first two experiments will be most important and therefore use 2 policies, whereas the final experiment only uses one policy, as it is not of great importance. Experiment 1 will be about exploration, and thus the comparison between novelty search and latin hypercube sampling. Experiment 2 will be about the influence of the optimization algorithm in place, or rather the influence of the outcome goals on novelty search. Finally, the third experiment will focus on the impact of the number of functional evaluations and the number of scenarios. Starting at only 1000 functional evaluations and scenarios multiplied with a factor ten each iteration up to one million. Table 4.1 gives a concise overview of the the experiments per policy.

Table 4.1: Experiments lake problem

	nfe	Search strategies	Outcome goals
Policy 1	1.000.000	LHC and NS	Default goals
Policy 2	1.000.000	LHC and NS	Default goals
Policy 3	1.000.000	LHC and NS	Default, all min and all max goals
Policy 4	1.000.000	LHC and NS	all min and all max goals
Policy 5	1.000 - 1.000.000	LHC and NS	Default goals

4.2. Results

This section is divided into the two main parts of this research. First of all, acquiring the optimal policies that are to be used in the novelty search experiments. Secondly the exploration experiments, that are about experimenting with novelty search. Since there are differences in the usage of the results a clear difference is made between these two. Another reason for this is the dependency between the two, since exploration cannot be started if the optimal policies are not available.

4.2.1. Optimal policies

The results of the optimal policy testing will be saved as a csv. Little to no statistics will be used, as this is not the point of interest. The only goal is to get 5 good policies. The expectation is to get approximately 25 policies per seed, thus with 5 seeds getting 125 policies in total. Therefore there should be sufficient room to get 5 policies.

With the existence of more than five policies an approach to identify diverse solutions is required to narrow down to five policies. A common practice is the diversity maximization approach by Carlsen et al. (2016) and Eker and Kwakkel (2018), but this would take too much time. Therefore a diverse solution identification approach that is based upon the diversity maximization approach

of Carlsen et al. (2016) and Eker and Kwakkel (2018) is used. Machine learning is used to create five different cliques from the available results, each clique representing one policy. Consider two cliques, or circles, then the center (or mean) of that clique always has a fixed distance to the other center. This fixed distance makes it perfect to generalize as a diverse solution, since the there will always be the distance of its distance to the edge. This prevents unnecessary computations, that is checking each solution against each solution. It is possible for the center of a clique to have no representation, in that case it would suffice to take the nearest neighbour of the center. The result of this diverse solution identification approach is 5 different and diverse policies in a matter of seconds.

4.2.2. Exploration

The results of novelty search will be saved as a csv, containing all uncertainties and outcomes as separate columns. The results of latin hypercube sampling will be saved as a compressed folder containing multiple csv files. One file containsn the experiment data, including the policy levers, uncertainties and the model. Four other interesting files are the outcomes of the experiments, max_P, inertia, utility and reliability. Furthermore some metadata files are present, but these are not of any significant importance. Finally a txt file is constantly updated with the start and end times of the experiment that was executed. The experiments contain the following information:

- Novelty search or Latin hypercube sampling (LHC);
- The number of functional evaluations;
- The policy number;
- The configuration of the outcome goals, or nothing for the default configuration.

The analysis of different policies is done using separate jupyter notebooks. The main features of these notebooks are combining the latin hypercube data, plotting the results, and checking validity. The main plots for displaying the outcomes will be parallel plots and scatter plotss. The validity check is done by generating the scenarios and re-running these on the model and comparing the results.

The parallel plots will not get a predetermined scale, however they will be normalized . Latin hypercube sampling results will be displayed in hairlines, allowing for multiple solutions to generate a clearly visible line. Novelty search lines will have the default line width. This also enables the combination of the results and therefore only displaying a single plot, instead of multiple plots for different runs. These plots should give a good overview of the distribution, dispersion and range of the data. Having multiple results in a single plot improves comparability, but can in some cases reduce readability. However as comparing is the more important, a slightly reduced readability is preferential when gaining a lot more comparability.

The final plot to convey the results is a scatter plot across different combinations of output variables. These plots are used to give a proper idea of the distribution across the output spaces. The results will be plot in one figure where the different search strategies are depicted using different colors. Latin hypercube sampling will be limited in the number of solutions, because it severely hinders readability if all results are displayed.

5

Results

This chapter will start with the optimization results, and finish with the main results of the novelty experiments. The optimization results are needed to acquire the policies that are to be used in the novelty search experiments. All experiments have been run using the the lake problem, that includes optimization experiments and novelty search experiments. The explanation of the model that is used for the lake problem can be found in chapter 1.4.

5.1. Optimization results

These results are generated using the default optimization algorithm, therefore all of the results generated can be considered to be on the Pareto front. There are two issues two tackle in displaying different results. First of all, there are many, so that reduces the quality of the picture, and secondly, it consists of 4 dimensions. The results consist of optimization runs with 5 different seeds, and had a size of 167 individual results.

The results could be split in two parts, the policy levers and their respective outcomes. The policy levers are needed for the further experiments. In figure 5.1 shows the spread of all outcomes, whereas 5.2 shows the spread of the lever values. These figures are quite chaotic, because they represent 167 different items. They can however already give an idea of the different results. In figure 5.1 the preferences of the objectives are aimed upwards, thus clearly showing max_P is minimized and others are maximized.

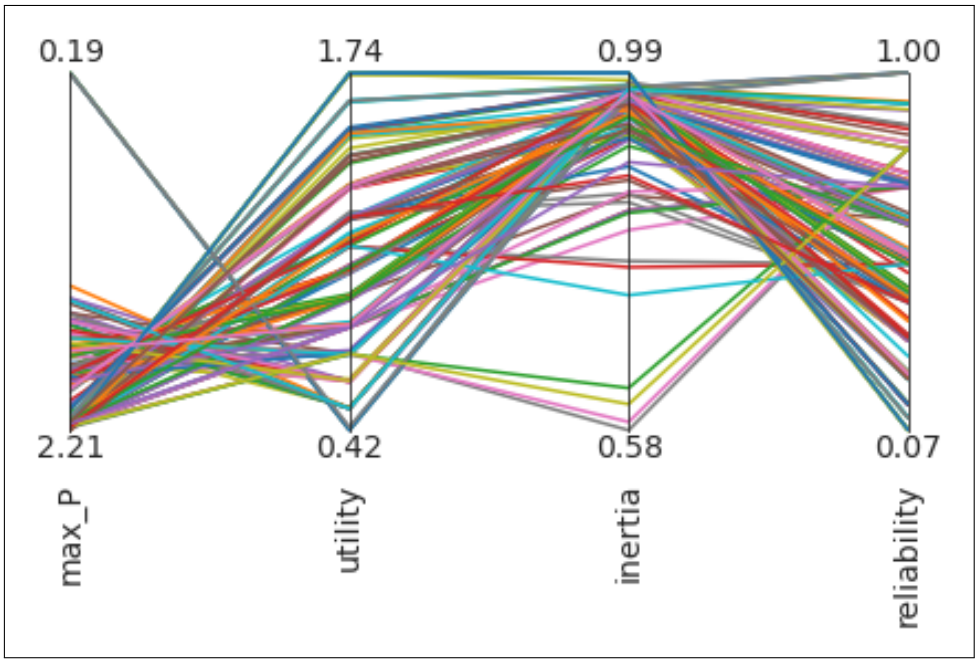

Figure 5.1: Comparison of individual observations for the output values of the optimized results

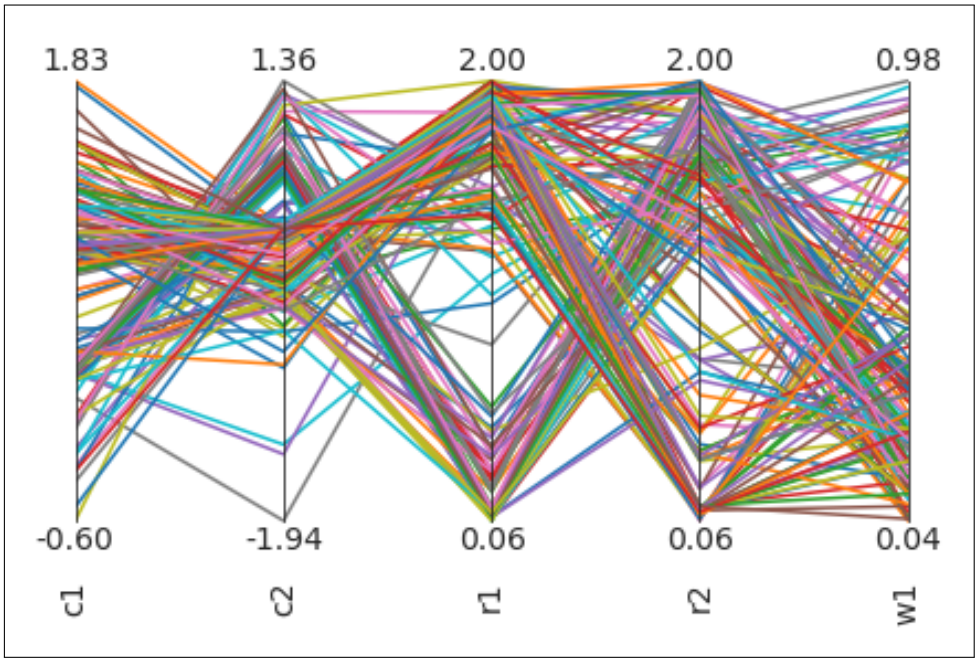

Figure 5.2: Comparison of individual observations for the input values of the optimized results

To reduce the results to only five solutions, let us divide the current results into five different cliques. Figure 5.3 gives an overview of these cliques, where each color represents a clique. The sub-figures show the different combinations of outcomes, and all containing the same cliques. Some of the sub-figures show a clear difference between the cliques, in other sub-figures the cliques are harder to detect. The cliques are generated using the Gaussian mean mixture model and all outcomes. The usage of all outcomes explains the overlap of cliques in some of the sub-figures and the clear difference in others. Taking the means of these clusters should give good results that are maximized in diversity.

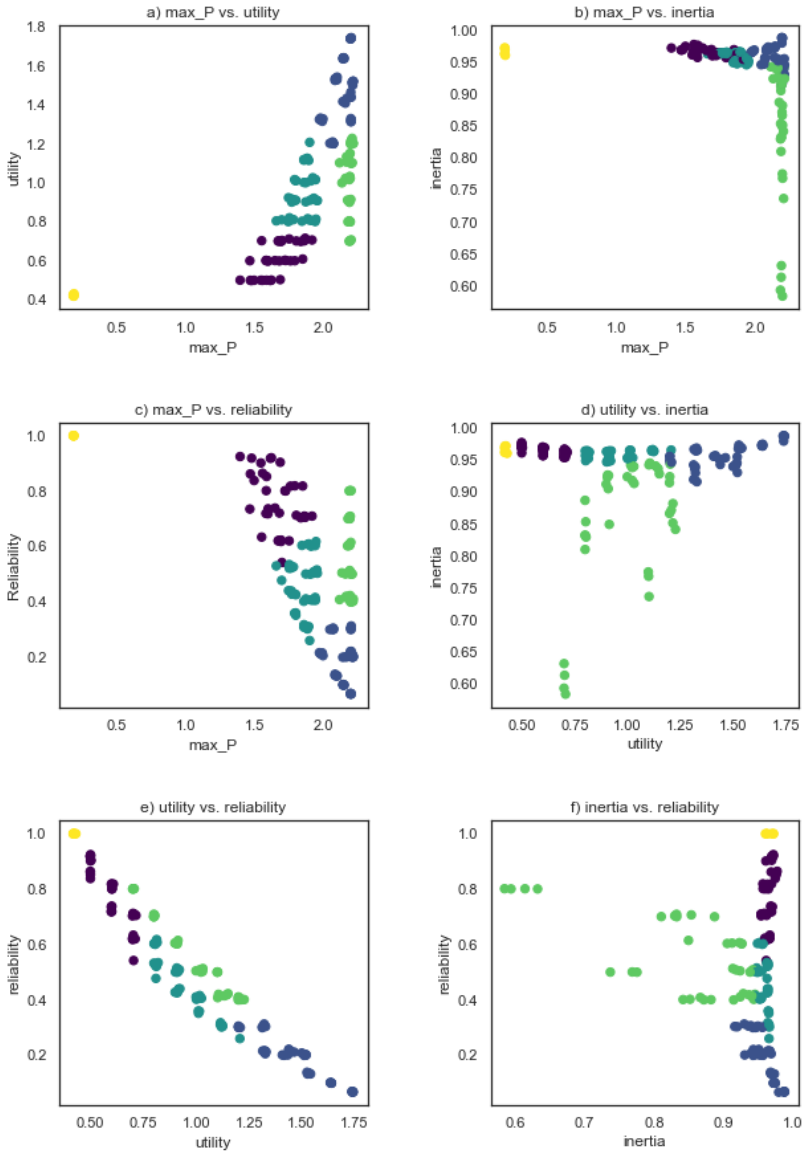

Figure 5.3: Colored optimized outcomes representing different cliques

Taking the mean from each of the five cliques gives the results as portrayed in table 5.1. These values are not present in the results, therefore a nearest neighbour search is needed to find the nearest neighbour of the mean. Those values, that can be said to represent the mean, have an associated policy lever which is needed for the real experiments. These values can be found in table 5.2, where the corresponding policies are also present.

Table 5.1: Mean values for all optimized cliques

max_P	Utility	Inertia	Reliability
2.131467	1.529885	0.965194	0.155472
1.916168	1.018227	0.955609	0.406914
0.193955	0.424356	0.966650	1.000000
2.193811	1.040205	0.857782	0.522767
1.704649	0.655923	0.963968	0.720546

Table 5.2: nearest existing neighbour of the mean of the cliques

c1	c2	r1	r2	w1	max_P	utility	inertia	reliability
-0.324375	0.966297	1.634259	1.536966	0.858481	2.098175	1.542225	0.968889	0.1333
0.677892	0.116634	1.497320	0.445654	0.770312	1.921762	1.026671	0.953737	0.4063
0.317884	0.005661	0.457124	0.820411	0.646500	0.191392	0.423406	0.963232	1.0000
0.835592	0.258739	1.855603	0.102020	0.532751	2.188747	1.033012	0.913636	0.5136
1.057577	-0.061549	1.948418	1.939679	0.269329	1.680635	0.600393	0.968990	0.7209

5.1.1. Conclusion

The values for the policy levers, **c1**, **c2**, **r1**, **r2** and **w1** found in table 5.2, will be used to create policy references that will be used as basis for the upcoming experiments around novelty search. An example of this would be: Policy('reference', c1=-0.324375, c2=0.966297, r1=1.634259, r2=1.536966, w1=0.858481).

5.2. Model exploration: The Lake problem

Through different experiments, that are based on different policies, the goal is to see if novelty search can have added value for exploratory modelling. The current, proven, way of doing this is through latin hypercube sampling. Therefore the ground truth, known as latin hypercube sampling, is present in each experiment. All experiments use one million functional evaluations, or scenarios unless another amount is explicitly mentioned. Each experiment will present the clearest figures, to support the cause. Detailed results for all policies, can be found in appendix B.

5.2.1. Experiment 1

The first experiment is about the comparison between novelty search and latin hypercube sampling. In figure 5.4 the results for novelty search and partly for latin hypercube sampling are presented in a parallel plot. The figure clearly shows that latin hypercube has 1 over sampled space and furthermore results for almost everywhere. Only for inertia and reliability gaps can be found in the results. These gaps are also present in the novelty search results, although they seem smaller for reliability and larger for inertia. The novelty search results for inertia pop out a lot, having 2 outliers and all others centered around the same point.

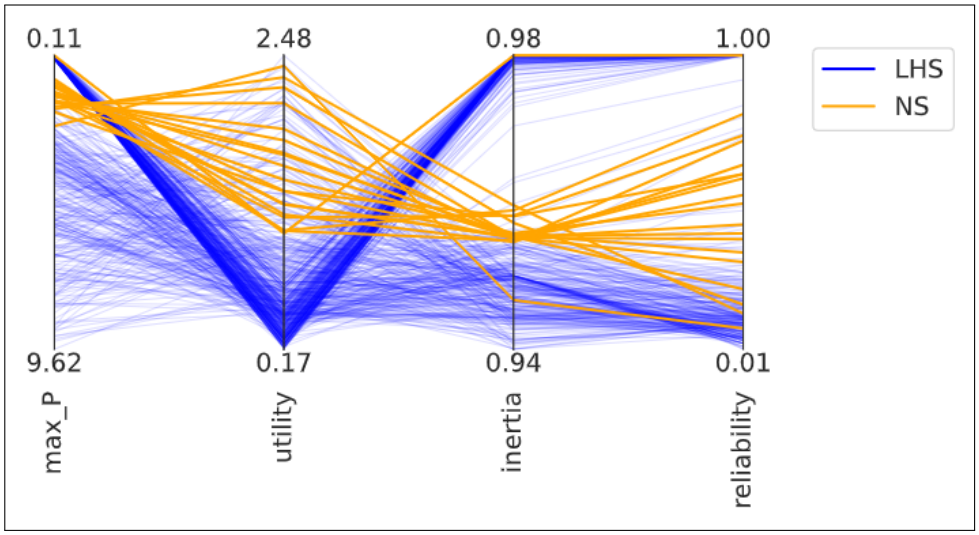

Figure 5.4: Parallel plot of the results for the second policy and first experiment

The second figure is used to visualize the results in a scatter plot, for all relevant variables. In this situation those are: max_P, utility, inertia and reliability. Figure B.5 is such a figure, containing 12 different scatter plots. Figure 5.5 is used to give an idea of the location of the results. Both novelty search and latin hypercube have been included in this figure, where orange again represent novelty search results and blue represents latin hypercube sampling results. The most interesting part to look at in the comparison of these scatter plots, is to see if different area's have been explored. These scatter plot figures contain the individual scatter plots twice, however in a different rotation. Finally scatter plots give a good overview of what areas have been explored and what not. Although there are many places of overlap, there are also plots with less overlap. This suggests that different results have been found. Especially looking at utility with reliability and max_P with utility there are clear differences in the plot. Whereas other plots such as inertia with reliability show a lot of overlap. These finding are reinforced by figure 5.6, showing the density of the results using a contour plot.

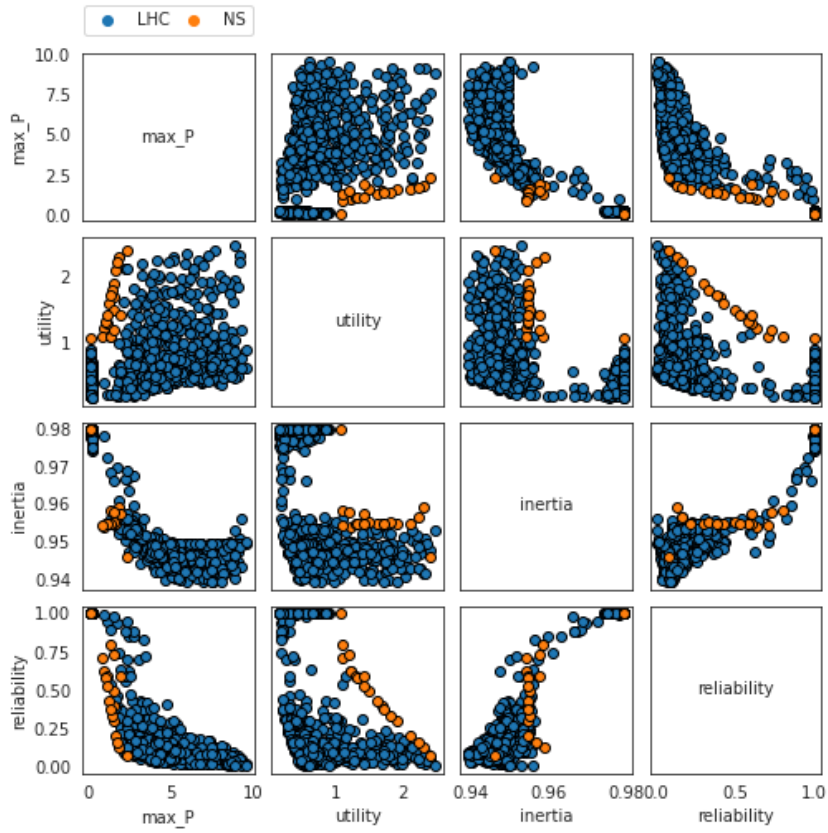

Figure 5.5: Scatter plot of the results of the second policy and first experiment; showing overlap and differences in results

The contour plot in figure 5.6 presents the contours of the results, or gives a visual representation of the density of the results. This supports the findings from 5.5 that different solutions have been identified, since it clearly shows contours that do not completely overlap. Indeed most of the novelty search contours have some overlap with those of latin hypercube sampling, but they also show clearly that there are places that do not overlap. With no overlap it means that the results do not present the same areas. Therefore novelty search has found areas that latin hypercube did not.

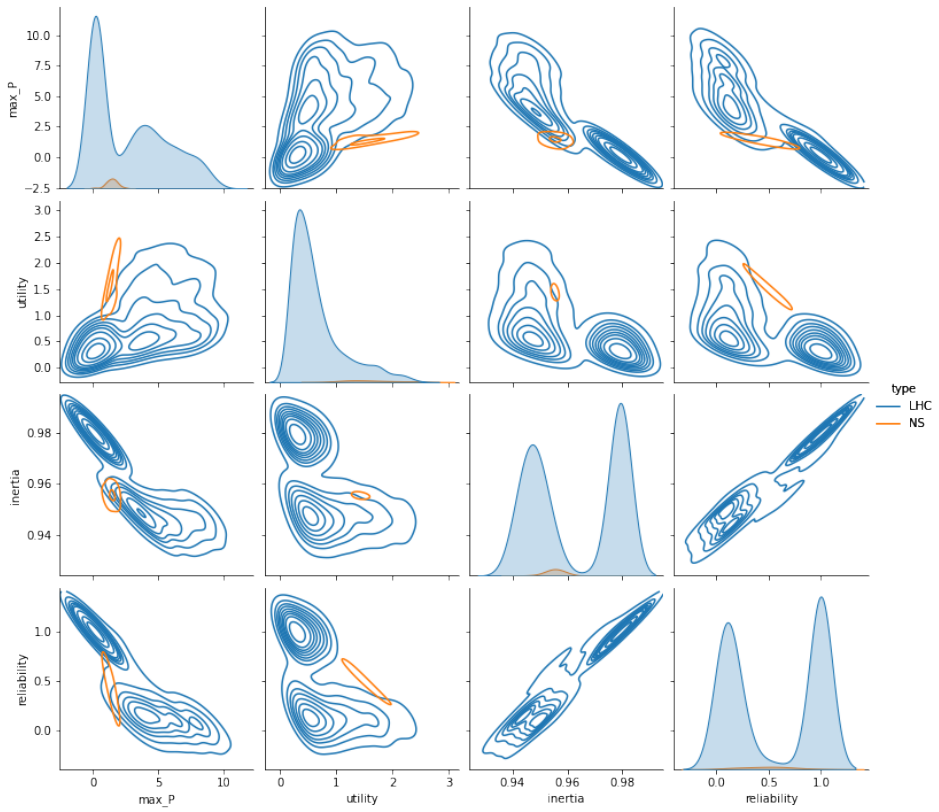

Figure 5.6: Contour plot of the second solution and first experiment; showing exploration of different areas

5.2.2. Experiment 2

The second experiment is about the effects of the outcome objectives on the results. Figure 5.7 present a parallel plot for latin hypercube sampling, novelty search and novelty search with all minimized and maximized objectives. The minimized results are hardly visible, since they are hidden behind some other results. The figure is quite chaotic, but it shows that there are clear differences between the maximized objectives and default objectives.

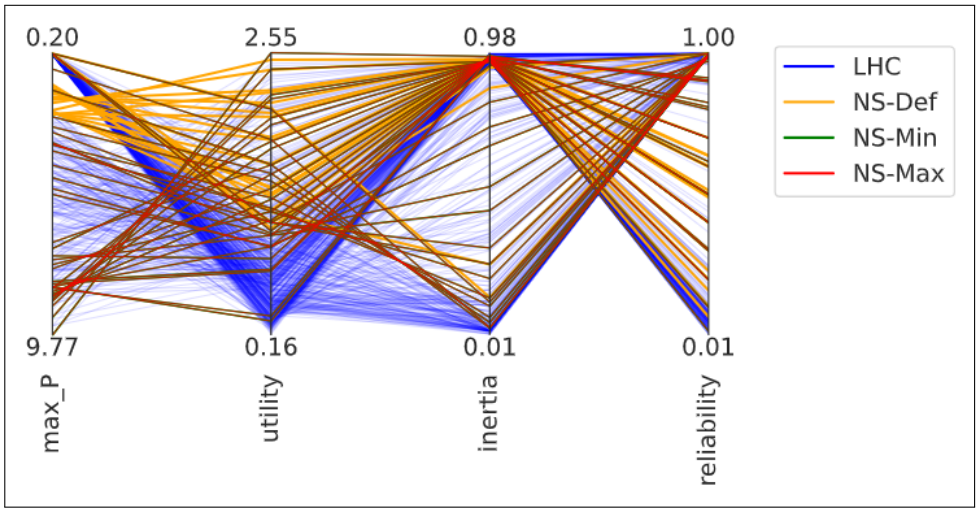

Figure 5.7: parallel plot of the results of the fourth policy and second experiment; showing different outcomes with different objectives

The scatter plot of the fourth policy and second experiment in figure 5.8 shows the explored areas of the different search strategies. Including, novelty search with default, minimized and maximized objectives, as well as latin hypercube sampling. In comparisson to the previous scatter plot, in figure 5.5, the results are different but in a similar manner. What is unexpected is the difference in results for different objectives. Novelty search with different objectives shows a clear difference in most individual plots. The only plot that is hard to read is the combination of reliability and inertia, which seemingly has all results clustered around the same points.

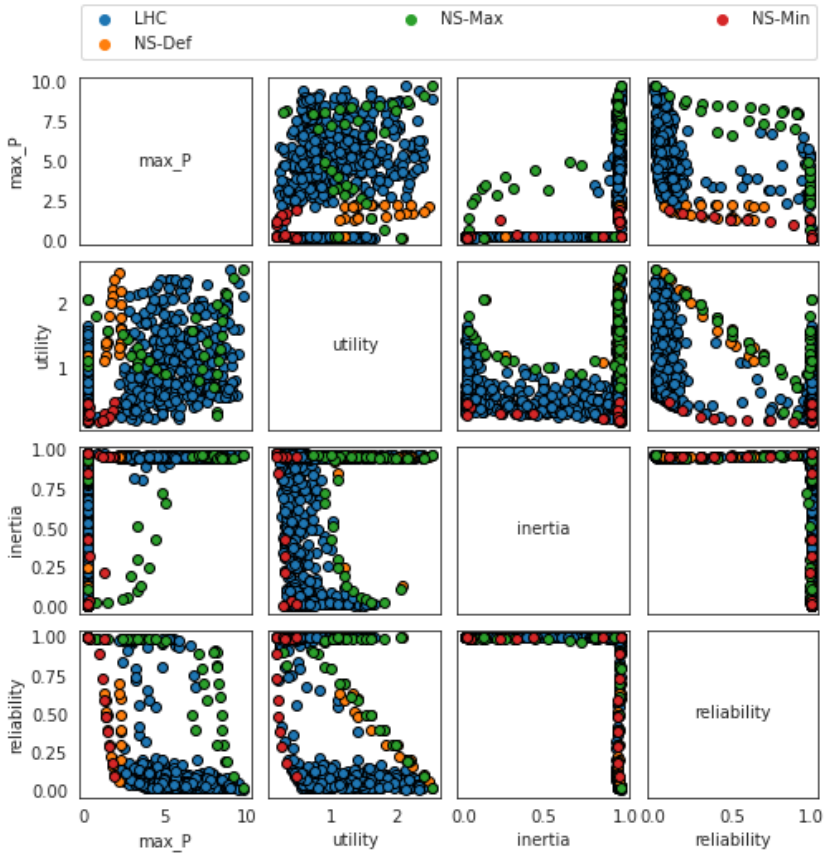

Figure 5.8: Scatter plot of the results of the fourth policy and second experiment; showing overlap and differences in results

The contour plot in figure 5.9 is used to reinforce the findings of the scatterplot. It does show more overlap, but still some parts that can be considered new for latin hypercube sampling. What it does show much more clearer, is the difference between the different objectives. Especially minimizing and maximizing objectives have their densities far from each other. The default objective seems to have some overlap with maximizing objectives. This seems reasonable with the default objective consisting of maximizing for three of the four outcomes. Something noteworthy is the combination of inertia and reliability, which seems to be missing the minimization objective. Looking at that combination in figure 5.8 the results are very dispersed. Which could mean that there is no density due to the distances. In other words, the relation for the density is nonexistent, which might be a direct cause of the spread of the results and the limited amount of results.

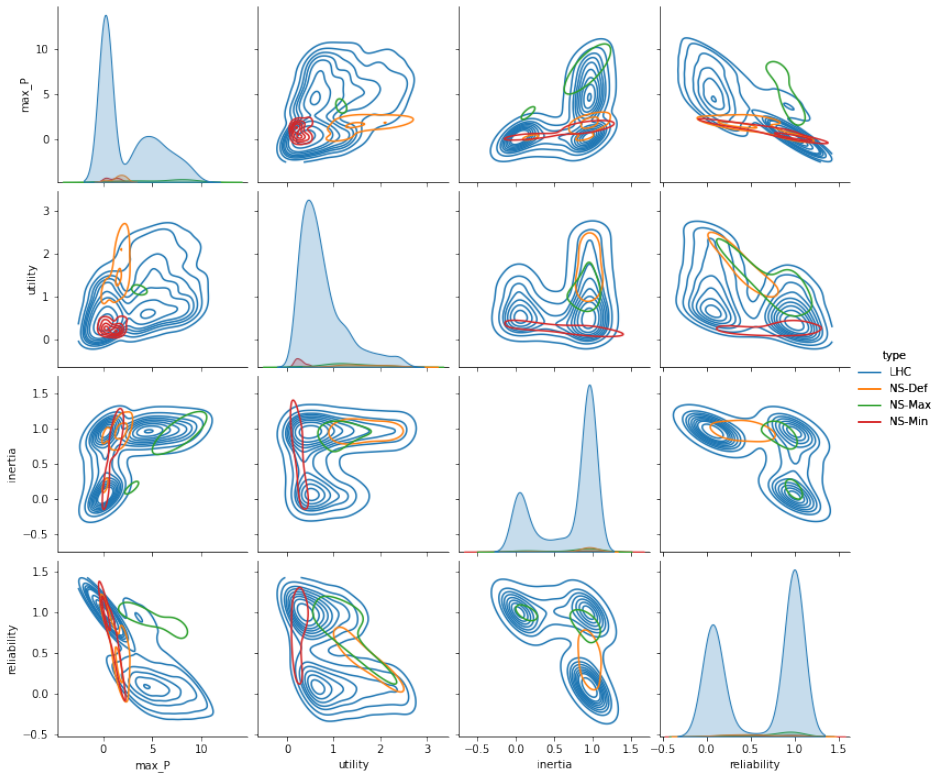

Figure 5.9: contour plot of the results of the fourth policy and second experiment; reinforcing the scatterplot

5.2.3. Experiment 3

The third experiment is to see the impact of the number of functional evaluations or scenarios. Since the latin hypercube sampling is not the point interest and not changing drastically, because the results are already reduced to be presented, these will only be displayed with one million scenarios. Figure 5.10 presents the parallel plot for the final experiment. The run with 10000 functional evaluations is left out, because it does not improve the figure. Figure 5.10 shows that the novelty search with different number of functional evaluations are similar. The colors that are most present are orange, red and blue, suggesting that these present most of the differences. With these 1000 functional evaluations the solutions are already similar and only take about half a minute to be executed.

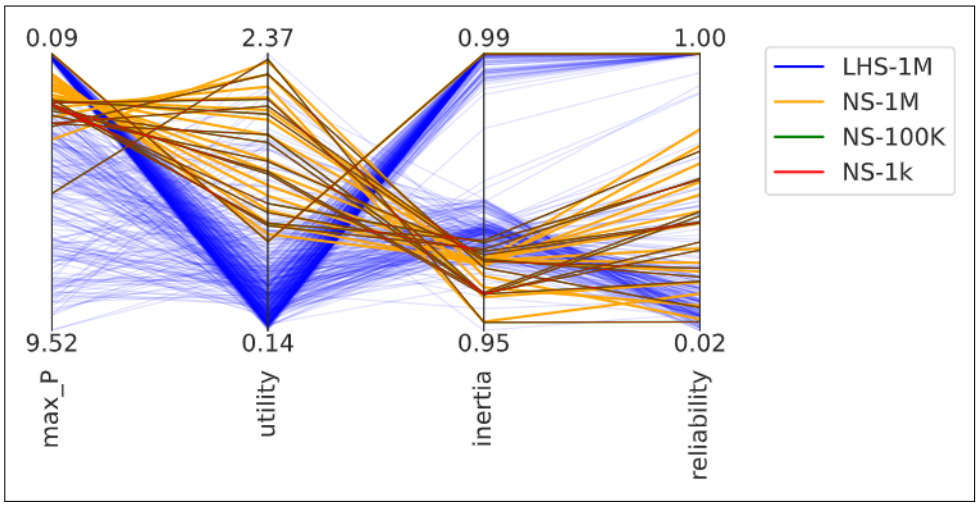

Figure 5.10: Parallel plot of the results for the final policy and experiment

Figure 5.11 displays the scatter plot containing all different runs for novelty search. It is surprising to see that the results can improve so much, as it is clear that results tend to go more towards the objectives when the number of functional evaluations increases. That also implies that the longer it runs, the more novel solutions will be identified. Logically speaking there will be an end to this, because there is a point that either all input has been tested or that no more novel solutions can be found. Either way it seems logical that there will be a turning point. Considering the solutions of 100000 functional evaluations are not visible, it is likely that one million functional evaluations does not gain anything new.

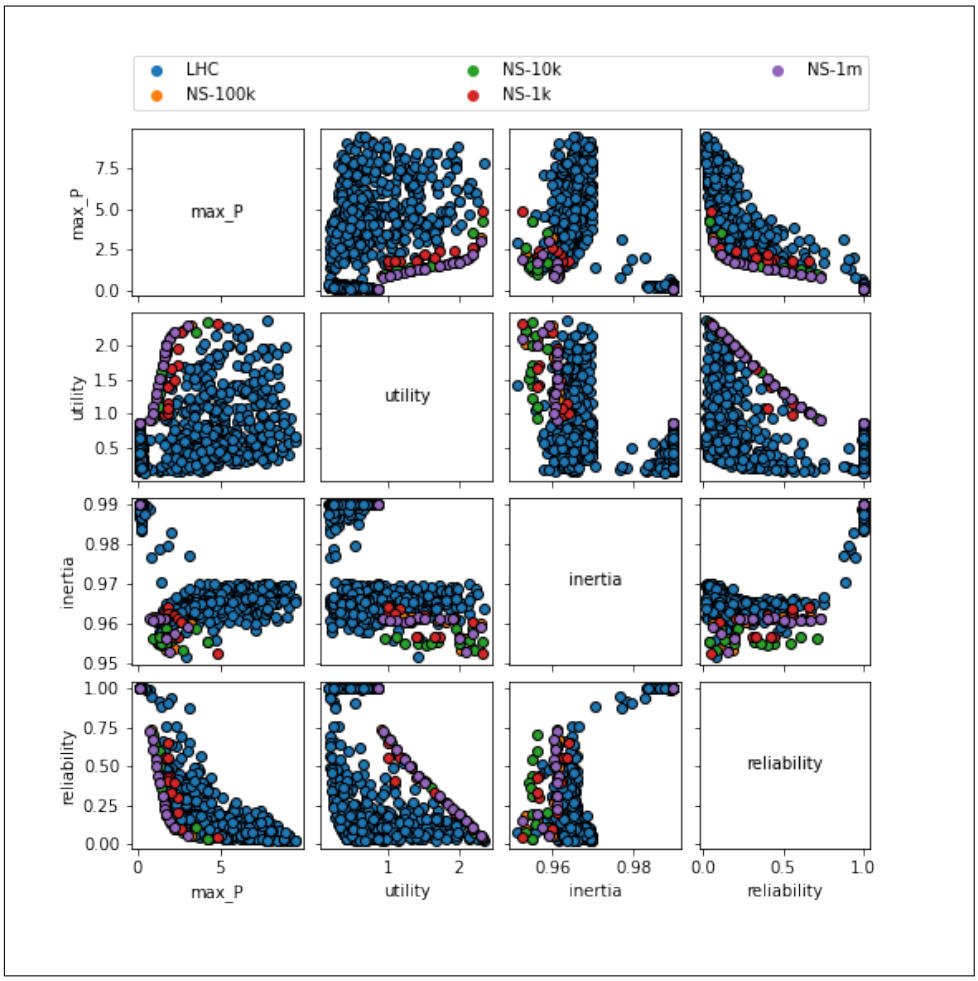

Figure 5.11: Scatter plot of the results for the final policy and experiment; showing the similarities between different nfe

Figure 5.12 shows the contour plot that is related to the scatter plot from figure 5.11. What becomes clear is that 100000 functional evaluations are not equal to one million, but have similar results in some plots. The reason for 100000 functional evaluations missing in the scatter plot of figure 5.11 is that it overlaps with some of the other results, but not always one million functional evaluations. Furthermore there is not much to say, as this is a similar as the scatter plot and perfectly supports figure 5.11.

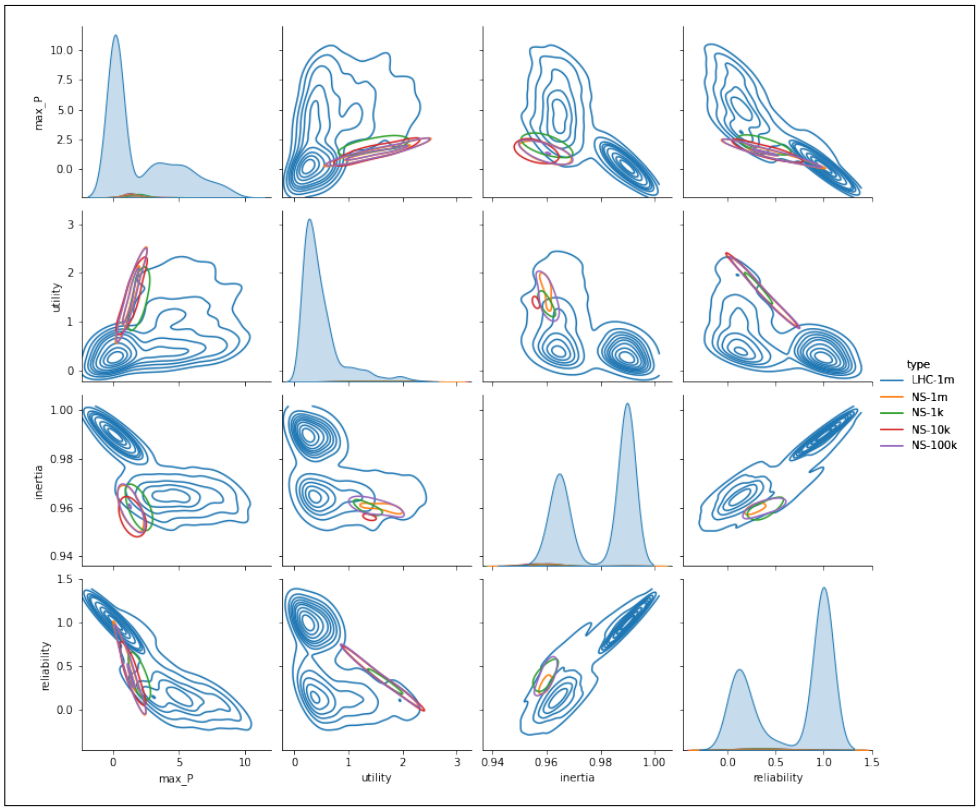

Figure 5.12: Contour plot of the results for the final policy and experiment; supporting the scatter plot

6

Discussion and Reflection

This chapter will discuss the results from chapter 5. The results on optimization will be discussed in short, only mentioning the important parts. The main focus will be on the novelty search and latin hypercube results, which is also the topic of this research. The discussion will be explaining, the difference in time, the difference in results, the challenges, assumptions and finally the interpretation of the results.

6.1. Optimization

The optimization part can be divided into two steps. First of all, running the optimization algorithm on the lake model, and secondly identifying the five best results. During the first step there is little to do, and little to go wrong, considering the EMA workbench can be trusted. This means the main story of this section will be focused on the second step, the identification of five diverse results. This step, as explained in chapter 4, requires more analysis and as a result is more prone to errors. The results of this can be found in chapter 5.

First of all, it is important to know that the results of the lake problem are 4-dimensional, meaning there are 4 output variables. Since 4-dimensional results cannot be displayed it is best to create 6 2-dimensional plots, where all variables are put next to each other. This can be seen in figure 5.3. Getting the best most diverse solutions is possible, but it would require individual checking of all results. In the worst case you might have to do this as many times as the factorial of the size of the results. To get an approximate on the diversity of the results, it is best to define as many cliques as diverse results. These cliques represent clear clusters of results, and therefore also clear differences in results. The mean or center of a clique should in general be furthest away from other centers. The mistake here is that the center of the cliques does not exist in the results. To solve this problem, the nearest neighbour of each of the clique center should be identified and used as solution. Although this does not identify the most diverse set of results, it is a very diverse set. It can be more diverse, but diversity maximization with 167 results would take hours or days to be identified.

There are two reasons for doing this identification of the five diverse results. First of all, it is not the main topic of this research and doing actual diversity maximization would take too long. This time would not be comparable to the time of novelty search, which would only take about 55 minutes for one million functional evaluations. Secondly, this type of identification already uses graphs and plots. These graphs and plots give good visual representation of the results and can therefore already be used in this paper. Figure 5.3, showcasing the different cliques, improves the reasoning for selecting the results due to the visual confirmation that is possible.

6.2. Novelty search

To properly discuss novelty search and reflect on that, it is best to separate this section into two parts. The first part will be about the novelty function, which is a huge part in novelty search. The second part will be about the experiments, results and what potential conclusions can be drawn from those.

6.2.1. Novelty function

The main requirement for a novelty function is to know what novelty is in the available context. Within this context everything is relevant, every piece of information is another hint at getting a better novelty function. The reason for saying this is, novelty is focused on exploration. The question is not only about what system you explore, but also in what way do you want to explore this system. Are you only there to find the extremes of the outcomes, is there only one relevant outcome or is the goal to get a well distributed and dispersed set of outcomes? These are relevant questions when thinking about novelty and exploration.

In the initial understanding of exploration, the idea was to get more extreme values for the outcomes. This resulted in creating a novelty function that focuses on going as far as possible from the crowd. The theoretical downside of this is locating two crowds far away from each other, or constantly ending up in the same places. Due to the assumption of only two parents there will also only be two crowds. In reality an evolutionary algorithm, and therefore novelty search as well, can function with more than two parents. In that case this novelty function is not likely to perform well, but this has not been tested. An option for that could be to work with cliques and use that to determine the most novel individuals to use as parents. The novelty function alone does not determine the result of novelty search, the evolutionary algorithm is important too.

The evolutionary algorithm that was used was Borg, which might be reason for the theoretical downsides not showing up in the results. Another reason might be the implementation of the evolutionary algorithms in the EMA workbench. The evolutionary algorithms require a outcome objectives, which results in the algorithm always pushing in that direction. These algorithms can be identified as optimization algorithms. From the third and fourth experiments it was clearly visible that these goals have a huge impact on the results. Figure 5.8 shows a distinction between the novelty results, which is the result of the impact of the objectives.

While the implementation was developed to create a novelty search implementation, it does match some of the characteristics of a quality diversity algorithm. Quality diversity algorithms use a combination of a novelty function and fitness function, and aim to find a diverse set of optimal solutions. This makes it normal for the implementation of novelty search to have characteristics of quality diversity, since quality diversity is based upon novelty search.

The implementation of the novelty function is inefficient. With each new generation, the parents need to be identified. To do this first, the most novel parent is found, and secondly a novel parent relatively close to other earlier parent is located. These two parents create a new individual. To find the first parent a crowding distance is calculated, and to find the second we use the crowding distance and nearest neighbour algorithm. The worst case run time, without any of the crowding distance or nearest neighbour calculations, then equals $O(n^2)$, where n equals the size of the population. This explains the inefficiency of the novelty function, and thus the increased run time.

6.2.2. Experiments and results

The experiments and results leave a lot of room for discussion, due to variation in results and the selection of figures. For the scatter plot and parallel plot figures it is impossible and undesirable to

display a lot of data. The more data those figures represent the less readable it becomes. Therefore the parallel plots only use 500 latin hypercube results, which include all minimum and maximum results of the outcomes and a random sample. The scatter plots have been capped at 1000 latin hypercube results, also including the minimum and maximum of the outcomes and a random sample. The reason for this is partially readability and partially because of the increased time to generate the graph with more results. Adding more samples to the generation does not make it significantly better, whereas reducing the samples makes it worse. This means the choice for 1000 samples is correct. For each of the graphs an attempt had been made to run it with all possible samples, which is one million for latin hypercube sampling, and those figures never generated completely after 24 hours. This means that the figure was not done yet, and having one figure take over a day to generate is too much. The figures for novelty search contain all novelty search results, because the maximum amount of novelty search results is 35. The average number of results is somewhere around 27.

Comparing novelty search with latin hypercube sampling is not really difficult. Novelty search has a lot less results, and to properly show the results of latin hypercube sampling a lot of results have to be discarded. However the fact remains that novelty search finds different solutions. Comparing the figures it looks like novelty search present different solutions, especially when looking at the density figures, 5.6, 5.9 and 5.12. Which show that novelty search has different densities than latin hypercube sampling. In fact, any version of novelty search that has been used in the experiments was able to find different results. Thus it can be concluded that novelty search has found novelty.

Although novelty has been identified, the question remains if this novelty is useful for exploration. None of the results are similar to those of latin hypercube sampling, although some do share similar variables. The results of novelty search have been rerun with the generated uncertainties and yield identical results. Therefore it can be concluded that these results are real results, and are already in the desired direction, however they might not be representative for the model. Given the limited amount of results it might present a poor overview of the potential outcomes, especially since one million latin hypercube samples do not give similar results to any of the tested novelty search strategies. The novelty search with minimized and maximized goals could be used to generate an idea of the border of potential outcomes. This does mean running novelty search twice. To reduce the time of experiments it is possible to reduce the number of functional evaluations. As depicted in the third novelty search experiment, lower number of functional evaluations still find novelties, and different results from latin hypercube sampling. However novelty search with a higher number of functional evaluations does give results further from the crowd and also further from latin hypercube sampling.

With the identification of novel outcomes, the algorithm does explore another part of the model. Since this has only been tested on the lake model, there is no data on how this would behave in different models. The results could have greatly improved by verifying these results in a different model.

No comparisons have been made with any of the optimization algorithms that are already present, since the goal of this research has been exploration. Seeing the results, which focus on uncertainty, it might be interesting to see how this implementation of novelty search compares to actual optimization algorithms, that focus on the policy levers. Whenever such explorations follow the same line as these have done, this implementation might provide more dispersed sets of optimized results. Whether these are improvements is currently unknown, and these are only speculations. These are interesting follow-up questions.

6.3. Assumptions

This section will clarify the important assumptions that have been used throughout the research. The assumptions will be listed in table 6.1. The first and second assumption were used to develop the initial novelty function. The third novelty function is used to make the transition between cython and python seemingless. Depending on the settings, or definitions of the variables, it has already been confirmed that cython and python variants yield the same results. The fourth and fifth assumption, are based on the research that was done for the literature review. Since the results show that the outcome goals have such a high influence, it implies that the outcome goals are the fitness functions. Before the results it was not clear how the algorithm would react to these goals and novelty function. The assumption is better described as not expecting so much influence rather than not knowing it is a fitness function. The research did show that any evolutionary algorithm is supposed to work for novelty search, however some algorithms do work better. Exactly this was the reason for choosing Borg.

Table 6.1: Assumptions

index	Assumption
1	The evolutionary algorithm only uses two parents
2	Initially the idea to find novelty or explore was by finding more extremes.
3	The cython variant of the lake problem is equal to the normal python variant.
4	Any evolutionary algorithm will work for novelty search.
5	The implemented evolutionary algorithms (in EMA) do not have (influential) fitness functions.

6.4. Challenges

In retrospect the current novelty function is by far complete. Especially in combination with the current algorithm, which already minimizes and maximizes certain outcomes, another novelty function could and should have been used. Although it is hard to say what kind of novelty function would have worked, since novelty functions are not really something that is readily available. Novelty functions cannot be generalised, as they are only meant for the topic they are in. This is made clear in the maze example, where any place not visited is considered to be a novelty. Similar problems can use similar novelty functions, so the current novelty function should in theory also work for similar models. The main similarity is the outcomes, that should all be Real numbers. Models with time based outcomes might not work as those results should be handled differently.

The reason for the previous challenge is most likely due to the lack of knowledge about novelty and exploration. The definition of both words are clear, however that is not something that can be easily adjusted to some model. The initial idea for exploration was focused on extremes. Although that is not specifically wrong, it is also not entirely correct. There is a common theme with this challenge and the lake problem. There is not a single correct solution. Novelty can be found in the edges of the solution space, but also in the centre. The focus is on finding solutions that have not been explored before. A perfect novelty function would be able to find all those solutions, whereas the current focus more on edges, as it is far away from the crowd.

Another challenge had to do with developing a new, implementing a different, evolutionary algorithm. Understanding evolutionary algorithms is doable, but creating such an algorithm, without proper pseudo-code, is extremely difficult. In the end it was best to give up on the idea of implementing that algorithm, because it would take too much time. Also, the current evolutionary algorithms have been optimized a lot, and have been made by someone with a lot more experience. Therefore it would have been an illusion to think that a self-made algorithm could even slightly compare to the

already available algorithms.

Although not specifically relevant to this research topic, the pandemic had a bit of influence over this research. With everyone working from home, the ability to promptly interact is reduced. You still have the opportunity to ask questions to your supervisor(s) via mail, and they do reply quite quickly. In some situations you are not really having an issue that can be easily conveyed. The only thing that is needed is someone to have a conversation with, obviously about a specific part of your study. You might have forgotten what word you need in the situation, or you are looking for a good implementation of some algorithm. Luckily search engines, such as google and duckduckgo, exist. Although these can be of help in many situations, a little spar with someone about the topic of interest can be more helpful.

Finally there were a bunch of decision based challenges, such as what figures to use for presenting the results, how much functional evaluations will be used, what would be the best experiments and how best to distribute the experiments. These questions are relatively simple, and cannot compare to the previous challenges. The first two challenges are definitely very important, and those definitions are taken for granted. Prior to this thesis, I would have never thought that these two tiny words would cause most of my problems.

6.5. Chapter summary

In summary, this chapter tells us about the challenges that occurred during the research, where novelty and exploration are prominently available. Some assumptions are given, and most importantly there is a discussion about the results. These show that the results as depicted in chapter 5 are fair, but not perfect. The reason for them not being perfect has to do with novelty and exploration, but also the implementation being inefficient. Finally the results have been discussed, realizing that novelty has been discovered, but the question remains if this novelty is useful.

7

Conclusion

This chapter provides the answers to the research questions and a wrap up of the research and this paper. As will be presented, novelty search does bring added value to exploratory modelling. The new information could be used to persuade opposing actors or contribute in forming alliances. Even though the algorithm in this research is far from complete, it did discover novelty. Quality diversity algorithms contain traces of novelty search and default optimization algorithms, which is why novelty search looks like a quality diversity algorithm. Although quality diversity algorithms are not the centre of this paper, it is an interesting algorithm for exploratory modelling. In situations where single solutions do not exist, quality diversity algorithms, with the focus on finding a diverse set of optimal solutions, are useful. This shows there is much more research to be done for exploratory modelling on novelty and novelty (search) based algorithms.

This chapter will start with the answers to the research questions, other relevant information and the conclusions that can be drawn from that. Following that is the societal and scientific relevance, which is briefly discussed in chapter 1, but will be more elaborate here. Subsequently, the last section of this chapter, and arguably this paper, will be about potential directions of future endeavors.

7.1. Research questions and their conclusions

This section will give an answer to the research questions, as defined in chapter 1, and also repeat insights from the research that are considered to be important. To give a complete overview of the answers the question will be repeated, so that it is clear where the answer belongs. The sub research questions will be answered first, after which main research question will be answered to end this section.

> (1) What are the key concepts, features, algorithms of novelty search and how do they work?

First of all novelty search is an algorithm in itself, so the only variations in existence are variations in implementation. These variations mostly differ in setup, so having a different novelty function, archive, parameters and/or different evolutionary algorithms. General pseudocode for novelty search can be found in algorithm 2, in chapter 2. With the development of novelty search, there have also been efforts into combinations of novelty search and normal evolutionary algorithms. Those algorithms focus on the combination of a novelty function and a fitness function. One prominent example is quality diversity algorithms, aiming to find a diversity of optimal solutions.

Novelty search only has a single key concept, where all other concepts are not key to novelty search. Examples of concepts that are not key to novelty search are the archive and the evolutionary algorithm, with the corresponding parameters. The key concept is the novelty function. The aim of this concept is to look for novelty, and therefore prioritizing exploration over optimization. Finally there is only a single novelty search algorithm, where the novelty function is dependent on the context. Different implementations exist, such as for neural networks or mazes, which highlights the context dependency of the implementations.

(2) What are the relevant search strategies in exploratory modeling and analysis and what are the relevant features to compare the different algorithms?

Exploratory modelling, and in our case the EMA workbench, has multiple search strategies and allows you to implement your own as well, without altering any of the already existing code. The search strategies can be divided into two types, optimization and exploration. The optimization type search strategies use evolutionary algorithms with outcome goals to find optimal solutions according to the outcome goals. Within the purpose of exploratory modelling the evolutionary algorithms are actually Many-Objective Evolutionary Algorithms (MOEA). Algorithms such as NSGA-II, EpsNSGA-II, SPEA2, Borg, and many more. It is also possible to either use your own algorithm, or base your algorithm on one of the available algorithms.

Exploration type search strategies use sampling techniques to sample the input find the solutions. In reality this means sampling x number of scenarios from the input space and running these samples through the model and returning the results. This implies that the result also contains x number of solutions. Available search strategies are: Monte Carlo sampling, Latin Hypercube Sampling, Partial factorial sampling, full factorial sampling and many more. For the lake problem model, which is used in this research, the latin hypercube sampling is the best sampling method to use. Not only is better than most of the other sampling techniques, it is also trusted for this particular model. It can therefore be considered as an established sampling technique for this model.

With novelty search aiming to explore, the logical choice is to compare novelty search with another exploration type search strategy. In this case Latin hypercube sampling will be the best search strategy to compare against. There are a few characteristics that are of interest in this comparison. First of all, the time it takes to run one exploration of one million scenarios or functional evaluations. Secondly, the range of the data and the distribution of the data. It is also interesting to see how the number of scenarios or functional evaluations changes the results and finally are there any differences in the data.

(3) Where would novelty search fit in exploratory modeling and analysis?

The implementation of novelty search fits best near the evolutionary algorithms, that is next to the optimization functions. Novelty search is very similar to the optimization functions and also requires an evolutionary algorithm. This implementation however is not the interesting part. The idea of novelty search fits best with exploration. However with a minor adjustment into a quality diversity algorithm, it might also work as a optimization algorithm. Since this has not been the point of interest during this research, this can only be briefly described and for further analysis a further research would be required. The fact remains that the current implementation did find novelty, because it had results that differ from latin hypercube sampling results.

Using novelty search with all outcome goals maximized even gives a well distributed and dispersed set of outcomes, with a range that is comparable to that of latin hypercube sampling. This

could be useful for exploration, however this will be met with a trade-off. The range is of values is decent and the distribution is good, but the run time is longer as well in similar circumstances.

Latin hypercube sampling is faster and has a larger set of results. Although this is not immediately better, this can be considered more useful for exploration. The results of novelty search, can be useful as well, but not standalone. Novelty search as a standalone exploration algorithm, in its current implementation, does not give a good overview of the output space. Novelty search next to latin hypercube sampling can be considered useful, since you get the wide range of results from latin hypercube sampling and the novel solutions from novelty search.

In summary, the current novelty search implementation is not the most efficient, but could work perfectly fine in addition to latin hypercube sampling. This implies that novelty search fits into the exploration part. With a few additions the novelty search algorithm can be transformed into a quality diversity algorithm and fit with the optimization algorithms. This latter has not been explored yet, but is a serious option.

> (4) How does novelty search compare to latin hypercube sampling in the lake problem model, under different circumstances?

Comparing novelty search to latin hypercube sampling is not easy. The only easy comparison can be made on the runtime, where novelty search is clearly slower than latin hypercube sampling. Novelty search is 2.75 times slower than latin hypercube sampling (55 minutes compared to 20). This is not that surprising, because the novelty function is inefficient, and a plain evolutionary algorithm already takes about 38 minutes.

Looking at the results there are multiple variations of novelty search to take into account. First of all, novelty search with the default outcome goals (minimizing max_P and maximizing all other outcomes). The second novelty search minimizes all outcome goals, and the last variation maximizes all outcome goals. Each variation generates different results, and therefore each variation should be taken into account, although maximizing all goals and the default goals do have similarities. This can be explained due to the many similarities in goals.

When looking at the range or dispersion of the results Latin hypercube has the widest range, although novelty search with all goals maximized comes really close or finds the same range, depending on the policy. The other novelty variations do not even come close to the range of novelty with maximized goals or latin hypercube sampling. A combination of novelty search that maximizes the goals and that minimizes the goals would result in a slightly better range. This margin is so small that it is not worth the extra time of another novelty computation. In general Latin hypercube sampling does give a better range, however novelty search with maximizing outcomes is a close second.

The distribution of the results is a difficult comparison, with the difference in range of values, and the missing results for latin hypercube sampling. These issues are hard to overcome, which makes it best to make these conclusions on the basis of the scatter plots and keep the difficulties in mind. The scatter plots give a good overview of the results, other than a few outliers the results follow a clear and well distributed pattern. This is the case for all novelty search strategies, but depending on the policy the outliers can be further away or closer. The available latin hypercube results give a more randomized impression. The area's that are easily reachable have a high density of results, whereas other areas are not explored or have a lower density of results. This clearly shows that with the limited amount of results, novelty search manages to discover new areas and has a better distribution of the results. Within the different novelty search strategies, the strategy that maximizes all outcomes has

the best distribution. The minimization and default strategy both present well distributed results, but some outliers exist.

(M) What is the added value of novelty search for Exploratory modeling and analysis?

Novelty search is able to find solutions in spaces that the current exploration strategies cannot, in this instance latin hypercube sampling. This implies that novelty search has found novelty. Looking at the results and their densities in figures 5.5, 5.6, 5.8, 5.9, 5.11 and 5.12, it is clear that the results are completely different. There is some overlap, but only in very small areas. The figures 5.4, 5.7 and 5.10 show that there are different relations in results. Although there seems to be overlap in some figures, this shows that this overlap is only for some figures and not all.

With the limited results and the limited dispersion of the results, novelty search will not present a representative set of results. In other words, only using novelty search will give a biased set of results in terms of the outcome goals. Therefore the current implementation of novelty search will not bring any added value as a replacement of the current exploration algorithms. The results are too limited and too biased to be replacing current exploration algorithms.

Since novelty search discovered new areas, and therefore novelty, it generates new information. This new information can be seen as added value for the analysts, or actors in the corresponding system. The extra information can be used to gain a better understanding, but also to bring actors together. Therefore the added value of novelty search in the exploration field is besides current exploration algorithms. This would mean that first a latin hypercube run would be done on the system and after that a novelty search algorithm would be used to find novelties in the system.

Considering figures 5.9, it can be seen that minimizing and maximizing outcome goals bounds the latin hypercube sampling results somewhat, but not perfectly. Knowing and the extra time it would take to run both of these, these configurations are not really useful. It does present us with options to define novelty in another way, or using other evolutionary algorithms to see how that would impact the results. There is much more to gain from novelty search, or novelty, than just novelty search. The algorithm can be slightly altered to become a quality diversity algorithm, or another novelty function could be defined. The current implementation therefore also adds value for further research into this topic.

7.2. Societal and scientific relevance

This thesis provides a clear scientific relevance, by using new algorithms to explore the output space. This resulted into new insights into the model and into the algorithm. With novelty function of novelty search is hard to define, and requires, in its current state, tweaking to work optimally. While this function is inefficient, it has already provided outcomes that were previously undiscovered. The result is that there is more information available on the algorithm and the model, but also shows that there is still a lot more to gain in the algorithm and thus potentially in the model as well.

The societal relevance is a bit harder to describe, as these models are used in deep uncertain situation, that usually contain multi-actor complexity. This basically implies that the actors cannot reach an agreement, due to the difference in their perception of the issue. These actors need to discuss the issues resolve them or convince the opposing parties that your solution is best. Due to the uncertainty, and difference, in these situations it is impossible to give a definitive explanation, but a reasoning can be given. With the new information on the problem at hand, the actors should have a bigger understanding of the problem and its potential solutions. This information can be

used to show the immediate need of a policy to solve the problem, or to improve the discussion. The novel solutions might present new ways of reaching a consensus that were not previously available. In the best possible scenario all actors unite, but in a more likely scenario some actors find ways to co-operate. The result is that opposing parties are more likely to find middle ground, and thus stimulate discussion and collaboration. While in the worst case scenario there is only increased information and no difference towards any consensus. With that also leaving the option to further develop the algorithm and improving the understanding of the issue even more.

7.3. Future work

This thesis leaves room for a lot further research into this specific topic. A few of those avenues will be discussed in more depth.

This entire paper is focused around a single model. This leaves a lot of room for discussions on the results and answers to the research questions. A simple way to resolve these could be by redoing this entire research with one or more models. This could already be a relatively simple line of future work. However a better way would be to redo this research with a better novelty function or evolutionary algorithm. The required outcome goals , belonging to the evolutionary algorithm, can be omitted in another algorithm to test the effects of no goals. Conversely, the novelty function has been defined as inefficient and determined to be sub-optimal. This would mean better novelty functions, or novelty function configurations, could be developed.

Knowing the outcome goals can be changed or left out in another evolutionary algorithm, there are two important paths that can be taken. Firstly, improving the novelty function and treating the algorithm as a quality diversity algorithm. There will be many differences with the current research, since the focus is on both exploration and optimization. Secondly, an attempt could be made at using a evolutionary algorithm without outcome goals, and thus treating the algorithm as a novelty search algorithm. Although the novelty search implementation is sub-optimal, the use of this would increase uniformity and therefore comparability of the results. Updating the novelty search implementation and keeping the comparability is possible by holding on to the current philosophy of the implementation. The current philosophy is focused on expanding the range of the outcomes and finding unexplored areas (by moving as far away from the crowd as possible). Although comparability is not equal to that of keeping the novelty search algorithm, it is still comparable. This last line of research might be a too big of a step forward, as it might also be necessary to know what a good quality diversity algorithm consists of.

Considering novelty search does add value to exploratory modelling, another future endeavor could focus on the novelty function(s). What makes a good novelty function considering some model. This could also use the lake problem, but other models should do fine as well. The focus should be on making the best possible novelty function for exploratory modelling purposes, or determining if such a novelty function can be made.

Finally, a future avenue should focus on the relation between novelty and exploration in the field of exploratory modelling. Although this an idea for a future endeavor and not a proposition, a main research question could focus on what new results mean to the understanding of a problem. The reason for this is, if novelty is discovered, then you get more information. Although more information is usually preferential, as it clears up a certain problem, it can also hugely impact the clarity. With the discovery of novel solutions far away from the crowd, the information can be considered weird or surprising, do they clarify the situation or make it less understandable.

Bibliography

Bankes, S., Walker, W. E., & Kwakkel, J. H. (2013). Exploratory modeling and analysis. Encyclopedia of operations research and management science, 532–537.

Bankes, S. (1993). Exploratory modeling for policy analysis. Operations research, 41(3), 435–449.

Box, G. E. (1976). Science and statistics. Journal of the American Statistical Association, 71(356), 791–799.

Cambridge University Press. (1995). https://dictionary.cambridge.org/dictionary/english/novelty

Carlsen, H., Lempert, R., Wikman-Svahn, P., & Schweizer, V. (2016). Choosing small sets of policy-relevant scenarios by combining vulnerability and diversity approaches. Environmental Modelling & Software, 84, 155–164.

Carpenter, S. R., Ludwig, D., & Brock, W. A. (1999). Management of eutrophication for lakes subject to potentially irreversible change. Ecological applications, 9(3), 751–771.

Cuccu, G., & Gomez, F. (2011). When novelty is not enough. European Conference on the Applications of Evolutionary Computation, 234–243.

Davis, P. K., Bankes, S. C., & Egner, M. (2007). Enhancing strategic planning with massive scenario generation: Theory and experiments (Vol. 392). Rand Corporation.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation, 6(2), 182–197.

Dufour, J.-M., & Neves, J. (2019). Chapter 1 - finite-sample inference and nonstandard asymptotics with monte carlo tests and r. In H. D. Vinod & C. Rao (Eds.), Conceptual econometrics using r (pp. 3–31). Elsevier. https://doi.org/https://doi.org/10.1016/bs.host.2019.05.001

Eiben, A. E., & Smith, J. E. (2015). What is an evolutionary algorithm? Introduction to evolutionary computing (pp. 25–48). Springer.

Eker, S., & Kwakkel, J. H. (2018). Including robustness considerations in the search phase of many-objective robust decision making. Environmental Modelling & Software, 105, 201–216.

Ferringer, M. P., Spencer, D. B., & Reed, P. (2009). Many-objective reconfiguration of operational satellite constellations with the large-cluster epsilon non-dominated sorting genetic algorithm-ii. 2009 IEEE Congress on Evolutionary Computation, 340–349.

Galvan, B., D., G., J., P., Sefrioui, M., & Winter, G. (2003). Parallel evolutionary computation for solving complex cfd optimization problems : A review and some nozzle applications. In K. Matsuno, A. Ecer, N. Satofuka, J. Periaux, & P. Fox (Eds.), Parallel computational fluid dynamics 2002 (pp. 573–604). North-Holland. https://doi.org/https://doi.org/10.1016/B978-044450680-1/50072-3

Gomes, J., Mariano, P., & Christensen, A. L. (2015). Devising effective novelty search algorithms: A comprehensive empirical study. Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, 943–950.

Groves, D. G., & Lempert, R. J. (2007). A new analytic method for finding policy-relevant scenarios. Global Environmental Change, 17(1), 73–85.

Hadka, D., & Reed, P. (2013). Borg: An auto-adaptive many-objective evolutionary computing framework. Evolutionary computation, 21(2), 231–259.

Harman, M., & Clark, J. (2004). Metrics are fitness functions too. 10th International Symposium on Software Metrics, 2004. Proceedings., 58–69.

Helton, J. C., & Davis, F. J. (2003). Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. Reliability Engineering & System Safety, 81(1), 23–69.

Hopfield, J. J. (1988). Artificial neural networks. IEEE Circuits and Devices Magazine, 4(5), 3–10.

Jain, A. K., Mao, J., & Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. Computer, 29(3), 31–44.

Jin, Y., Olhofer, M., & Sendhoff, B. (2002). A framework for evolutionary optimization with approximate fitness functions. IEEE Transactions on evolutionary computation, 6(5), 481–494.

Jobst, B., & Meinel, C. (2014). How prototyping helps to solve wicked problems. Design thinking research (pp. 105–113). Springer.

Kasprzyk, J. R., Nataraj, S., Reed, P. M., & Lempert, R. J. (2013). Many objective robust decision making for complex environmental systems undergoing change. Environmental Modelling & Software, 42, 55–71.

Kollat, J. B., & Reed, P. M. (2005). The value of online adaptive search: A performance comparison of nsgaii, ε-nsgaii and εmoea. International Conference on Evolutionary Multi-Criterion Optimization, 386–398.

Kucherenko, S., Albrecht, D., & Saltelli, A. (2015). Exploring multi-dimensional spaces: A comparison of latin hypercube and quasi monte carlo sampling techniques. arXiv preprint arXiv:1505.02350.

Kwakkel, J. H. (2017). The exploratory modeling workbench: An open source toolkit for exploratory modeling, scenario discovery, and (multi-objective) robust decision making. Environmental Modelling & Software, 96, 239–250.

Kwakkel, J. H., Haasnoot, M., & Walker, W. E. (2016). Comparing robust decision-making and dynamic adaptive policy pathways for model-based decision support under deep uncertainty. Environmental Modelling & Software, 86, 168–183.

Kwakkel, J. H., & Pruyt, E. (2013). Exploratory modeling and analysis, an approach for model-based foresight under deep uncertainty. Technological Forecasting and Social Change, 80(3), 419–431.

Kwakkel, J. H., & Pruyt, E. (2015). Using system dynamics for grand challenges: The esdma approach. Systems Research and Behavioral Science, 32(3), 358–375.

Laumanns, M., Thiele, L., Deb, K., & Zitzler, E. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. Evolutionary computation, 10(3), 263–282.

Lehman, J., & Stanley, K. O. (2008). Exploiting open-endedness to solve problems through the search for novelty. ALIFE, 329–336.

Lehman, J., & Stanley, K. O. (2011a). Abandoning objectives: Evolution through the search for novelty alone. Evolutionary computation, 19(2), 189–223.

Lehman, J., & Stanley, K. O. (2011b). Evolving a diversity of virtual creatures through novelty search and local competition. Proceedings of the 13th annual conference on Genetic and evolutionary computation, 211–218.

Lehman, J., & Stanley, K. O. (2011c). Novelty search and the problem with objectives. Genetic programming theory and practice ix (pp. 37–56). Springer.

Lehman, J., Stanley, K. O., & Miikkulainen, R. (2013). Effective diversity maintenance in deceptive domains. Proceedings of the 15th annual conference on Genetic and evolutionary computation, 215–222.

Lempert, R. J. (2003). Shaping the next one hundred years: New methods for quantitative, long-term policy analysis.

Moallemi, E. A., Kwakkel, J., de Haan, F. J., & Bryan, B. A. (2020). Exploratory modeling for analyzing coupled human-natural systems under uncertainty. Global Environmental Change, 65, 102186.

Murata, T., & Ishibuchi, H. (1995). Moga: Multi-objective genetic algorithms. IEEE international conference on evolutionary computation, 1, 289–294.

Najim, K., Ikonen, E., & Daoud, A.-K. (Eds.). (2004). Chapter 3 - optimization techniques. In Stochastic processes (pp. 167–221). Kogan Page Science. https://doi.org/https://doi.org/10.1016/B978-190399655-3/50012-8

Pugh, J. K., Soros, L. B., Szerlip, P. A., & Stanley, K. O. (2015). Confronting the challenge of quality diversity. Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, 967–974.

Quinn, J. D., Reed, P. M., & Keller, K. (2017). Direct policy search for robust multi-objective management of deeply uncertain socio-ecological tipping points. Environmental modelling & software, 92, 125–141.

Risi, S., Vanderbleek, S. D., Hughes, C. E., & Stanley, K. O. (2009). How novelty search escapes the deceptive trap of learning to learn. Proceedings of the 11th Annual conference on Genetic and evolutionary computation, 153–160.

Ritchey, T. (2013). Wicked problems. Acta morphologica generalis, 2(1).

Roetzel, W., Luo, X., & Chen, D. (2020). Chapter 6 - optimal design of heat exchanger networks. In W. Roetzel, X. Luo, & D. Chen (Eds.), Design and operation of heat exchangers and their networks (pp. 231–317). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-817894-2.00006-6

Singh, R., Reed, P. M., & Keller, K. (2015). Many-objective robust decision making for managing an ecosystem with a deeply uncertain threshold response. Ecology and Society, 20(3).

Slowik, A., & Kwasnicka, H. (2020). Evolutionary algorithms and their applications to engineering problems. Neural Computing and Applications, 1–17.

Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. Evolutionary computation, 10(2), 99–127.

Stein, M. (1987). Large sample properties of simulations using latin hypercube sampling. Technometrics, 29(2), 143–151.

TIBERMACINE, A., & DJEDI, N. (2014). Neat neural networks to control and simulate virtual creature's locomotion. International Conference on Multimedia Computing and Systems -Proceedings, 0. https://doi.org/10.1109/ICMCS.2014.6911392

Velez, R., & Clune, J. (2014). Novelty search creates robots with general skills for exploration. Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, 737–744.

Yusoff, Y., Ngadiman, M. S., & Zain, A. M. (2011). Overview of nsga-ii for optimizing machining process parameters. Procedia Engineering, 15, 3978–3983.

List of Figures

5.1 Comparison of individual observations for the output values of the optimized results 23
5.2 Comparison of individual observations for the input values of the optimized results . 23
5.3 Colored optimized outcomes representing different cliques 24
5.4 Parallel plot of the results for the second policy and first experiment 26
5.5 Scatter plot of the results of the second policy and first experiment; showing overlap and differences in results 27
5.6 Contour plot of the second solution and first experiment; showing exploration of different areas 28
5.7 parallel plot of the results of the fourth policy and second experiment; showing different outcomes with different objectives 28
5.8 Scatter plot of the results of the fourth policy and second experiment; showing overlap and differences in results 29
5.9 contour plot of the results of the fourth policy and second experiment; reinforcing the scatterplot 30
5.10 Parallel plot of the results for the final policy and experiment 30
5.11 Scatter plot of the results for the final policy and experiment; showing the similarities between different nfe 31
5.12 Contour plot of the results for the final policy and experiment; supporting the scatter plot 32

B.1 First policy parallel plot of latin hypercube results 51
B.2 First policy feature scoring plot of latin hypercube results 51
B.3 First policy parallel plot of novelty search results 52
B.4 First policy feature plot of novelty search results 52
B.5 First policy scatter plot 53
B.6 Second policy parallel plot of latin hypercube results 54
B.7 Second policy feature scoring plot of latin hypercube results 55
B.8 Second policy parallel plot of novelty search results 55
B.9 Second policy feature scoring plot of novelty search results 56
B.10 Second policy scatter plot of all results 57
B.11 Third policy parallel plot of latin hypercube results 58
B.12 Third policy feature scoring plot of latin hypercube results 58
B.13 Third policy parallel plot of novelty search default goals results 59
B.14 Third policy parallel plot of novelty search minimized goals results 59
B.15 Third policy parallel plot of novelty search maximized goals results 59
B.16 Third policy feature scoring plot of novelty search default results 60
B.17 Third policy feature scoring plot of novelty search minimized results 60
B.18 Third policy feature scoring plot of novelty search maximized results 61
B.19 Third policy scatter plot 62
B.20 Fourth policy parallel plot of latin hypercube results 63
B.21 Fourth policy feature scoring plot of latin hypercube results 63
B.22 Fourth policy parallel plot of novelty search default goals results 64
B.23 Fourth policy parallel plot of novelty search minimized goals results 64
B.24 Fourth policy parallel plot of novelty search maximized goals results 64

B.25 Fourth policy feature scoring plot of novelty search default results 65
B.26 Fourth policy feature scoring plot of novelty search minimum results 65
B.27 Fourth policy feature scoring plot of novelty search maximum results 66
B.28 Fourth policy scatter plot 67
B.29 Fifth policy parallel plot of latin hypercube results 68
B.30 Fifth policy feature scoring plot of latin hypercube results 68
B.31 Fifth policy parallel plot of 1000 novelty search results 69
B.32 Fifth policy parallel plot of 10000 novelty search results 69
B.33 Fifth policy parallel plot of 100000 novelty search results 69
B.34 Fifth policy parallel plot of 1 million novelty search results 70
B.35 Fifth policy feature scoring plot of 1000 novelty search results 70
B.36 Fifth policy feature scoring plot of 10000 novelty search results 71
B.37 Fifth policy feature scoring plot of 100000 novelty search results 71
B.38 Fifth policy feature scoring plot of 1 million novelty search results 72
B.39 Fifth policy scatter plot with 1000 latin hypercube results 73
B.40 Fifth policy scatter plot with one million latin hypercube results 74

List of Tables

1.1 Deep uncertain parameters information 7

4.1 Experiments lake problem 20

5.1 Mean values for all optimized cliques 25
5.2 nearest existing neighbour of the mean of the cliques 25

6.1 Assumptions 36

A.1 Tools 49
A.2 Python packages 49

A

Software and Packages

This research has required the usage of a variety of computational experiments, and other computer based tools. Since updates to the used tools happen quite frequently, and potentially altering important segments of the tools, the versions of the software will be provided. These versions should aid the reproducibility, because it worked for me and I can only guarantee that these version provide the output that they do. The only items that will be listed are the main tools, since these tools also have a lot of dependencies. Table A.1 presents an overview of the important tools that have been used. Table A.2 presents the important python packages that have been used, excluding their dependencies. The code snippets that are used can be found on my github repository once you have been invited to that repository, because it is still in private mode. All experiments have been run on a Linux 20 LTS 64 bit system. Although I have also tested the experiments on Windows and with the proper setup that runs as well.

A.1. Tools

Table A.1: Tools

Tool	Version	Usage
Python	3.8.2	General programming language
Cython	0.29.23	Optimized python for model exploration

A.2. Python Packages

Table A.2: Python packages

Package	Version	Usage
ema-workbench	2.0.8	Exploratory modelling and analysis tool
jupyter-client	6.1.12	Data science and analysis (easily available and executable code)
pandas	1.2.3	Storing and reading data
numpy	1.20.2	array computations
matplotlib	3.4.0	Data science and analysis (plots)
scipy	1.6.2	data science and analysis (analysis mainly)
sklearn	0.0	data science and analysis (nearest neighbour algorithm calculation)
scikit-learn	0.24.2	real version of sklearn
platypus-opt	1.0.4	provides optimization algorithms

B

Policy Results

This appendix will present the results per policy. It will not include contour plots as these are useful, but they are more a small extension of the scatter plots and therefore not useful to show again. This appendix will also present feature scoring matrices, which in itself say nothing, but when comparing with the other matrices they say something about the solutions that have been identified.

B.1. Per Policy Exploration Results

The explanation of the model that is used for the lake problem can be found in chapter 1.4. Through different experiments, that are based on different policies, the goal is to see if novelty search can have added value for exploratory modelling. The current, proven, way of doing this is through latin hypercube sampling. Therefore the ground truth, known as latin hypercube sampling, is present in each experiment. Due to the availability of a cythonized model, the speed of execution is significantly increased. All experiments use one million functional evaluations, or scenarios unless another amount is explicitly mentioned. The lake problem as defined in chapter 1 is used, which is the cythonized DPS version. Each run of novelty search takes about 55 minutes for one million functional avaluations, whereas latin hypercube sampling takes about 20 minutes for one million scenarios.

B.1.1. Policy 1

The values for the policy lever in this policy are: $c1 = -0.324375$, $c2 = 0.966297$, $r1 = 1.634259$, $r2 = 1.536966$ and $w1 = 0.858481$.

Latin hypercube

It is important to note that these outcomes, portrayed as figures, are the base for this research. These can be considered the truth, as they have proven to work correctly, although not always most efficiently. In figure B.1 it is most important to look at the minimum and maximum values. The goals for each outcome are displayed in the top, therefore the max_P column is inverted and shows lower values on top. The goal is to minimize max_P in the model. Since the other outcomes should be maximized, the higher values are displayed in the top of the figure.

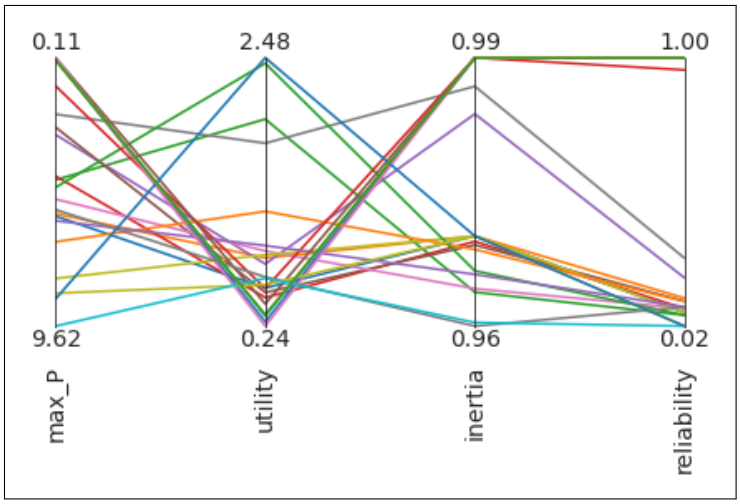

Figure B.1: First policy parallel plot of latin hypercube results

Figure B.2 shows a feature scoring matrix, focusing on the influence of an uncertainty over an outcome. It clearly shows that uncertainty b is mostly influencing the max_P in the model. The lower the value in the matrix, the lower the influence. These values have been generated using the minimum and maximum values and a 1000 other random samples from the results. Therefore these values might give an generalized representation, and will not always hold. It is possible that another uncertainty is more influential in a single case, it might also be possible for the policies to be the more influential factor, however that cannot be determined due to the the lack of policies. It is interesting to see how this will compare to the same matrix only generated with the very few results from novelty search. Having more samples to generate this plot is favorable, since more values give a better result. Another interesting comparison is to other feature scoring matrices from other policies that have been explored using latin hypercube.

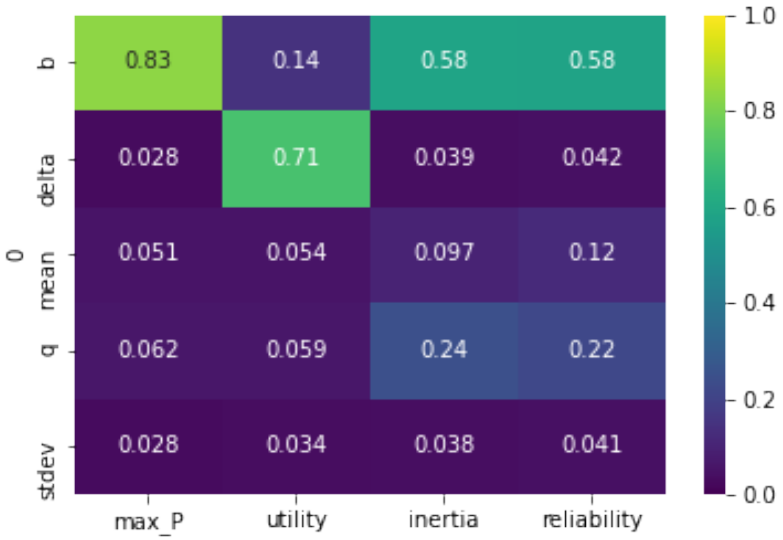

Figure B.2: First policy feature scoring plot of latin hypercube results

Novelty search

Figure B.3 gives a normalized overview of the results for novelty search. The minimum and maximum value are from the results acquired through latin hypercube sampling. Figure B.3 in itself gives a well distributed overview, but due to the comparison with latin hypercube sampling, most parts have been explored according to the previously set outcome goals. That is max_P is very low in general, and utility and inertia are quite high.

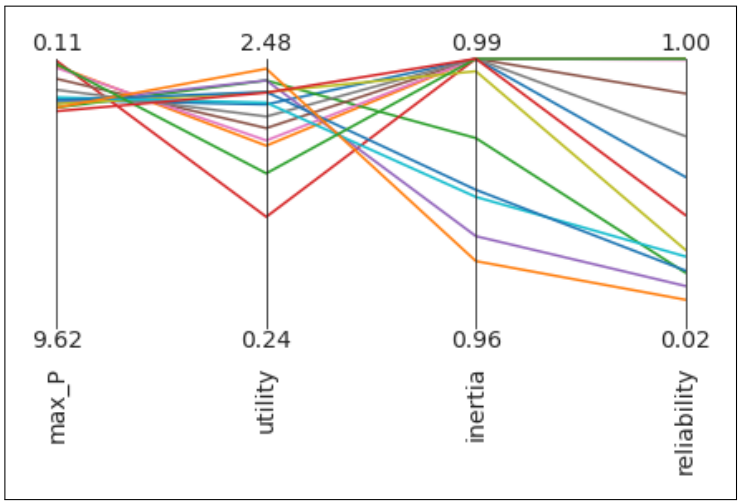

Figure B.3: First policy parallel plot of novelty search results

Figure B.4 shows the feature scoring matrix for novelty search. The most prominent difference to be identified is the influence of the variable q. For novelty search based results, q has a lot more influence over the results. Another point of interest is the influence of delta over utility, which is almost none for novelty search and close to 0.71 for latin hypercube in figure B.2.

Figure B.4: First policy feature plot of novelty search results

The final figure is used to visualize the results in a scatter plot, for all relevant variables. In this situation those are: max_P, utility, inertia and reliability. Figure B.5 is such a figure, containing 12

different scatter plots. Figure B.5 is used to give an idea of the spread of the results. Both novelty search and latin hypercube have been included in this figure, where orange dots represent novelty search results and the blue dots represent latin hypercube results. The most interesting part to look at in the comparison of these scatter plots, is to see if different area's have been explored. These scatter plot figures contain the individual scatter plots twice, however in a different rotation. Finally in a figure like figure B.5 gives a good overview of what has been explored by each search strategy. Although there are many places of overlap, there are also plots with less overlap. This suggests that different results have been found.

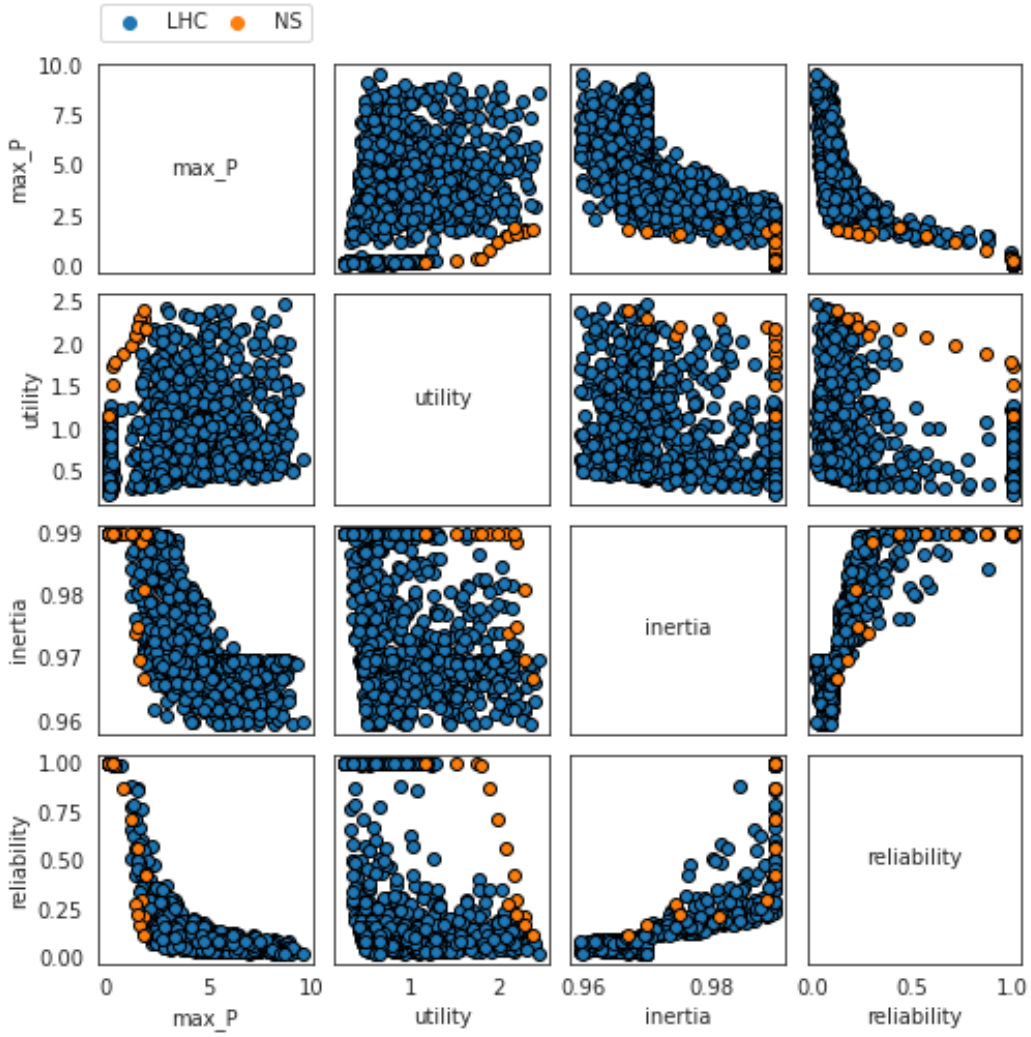

Figure B.5: First policy scatter plot

B.1.2. Policy 2

The values for the policy levers in this policy are: $c1 = 0.677892$, $c2 = 0.116634$, $r1 = 1.497320$, $r2 = 0.445654$ and $w1 = 0.770312$. The experiment from this policy will be the same as from the first policy. That is we will see how novelty search in default compares to latin hypercube sampling.

Latin hypercube

Latin hypercube sampling is considered to be the base of these experiments. Figure B.6 shows similar a similar plot to that of figure B.1. The distribution of the results is not equally distributed. This is is easily visible for inertia and reliability, where large gaps in between results exist. The figures clearly show some parts that are explored deeply and others that could use a bit more exploration.

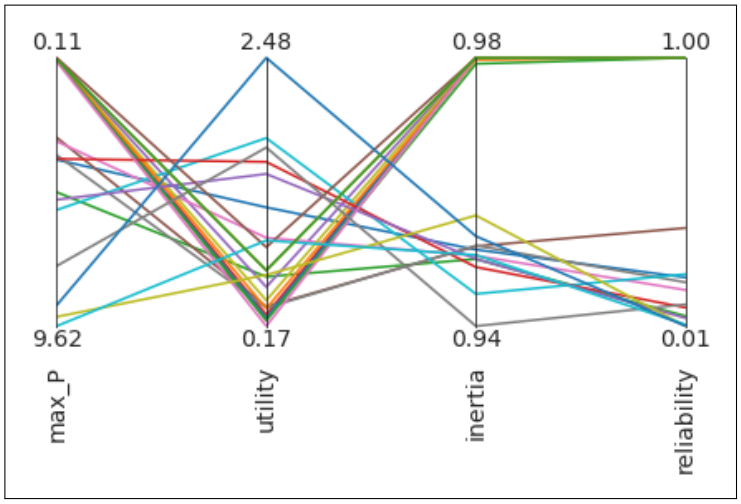

Figure B.6: Second policy parallel plot of latin hypercube results

The feature scoring matrix from figure B.7 does not show anything out of the ordinary. It is not identical to figure B.2, but there are similarities. The biggest difference is the influence of delta over utility.

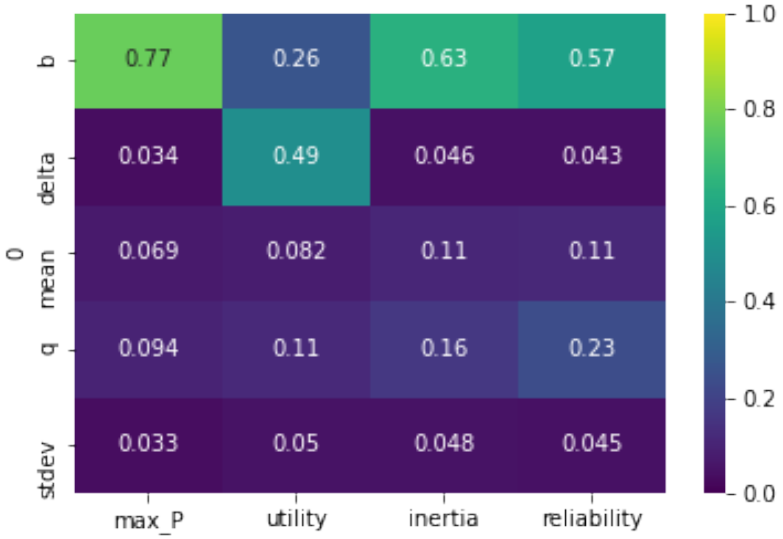

Figure B.7: Second policy feature scoring plot of latin hypercube results

Novelty search

The novelty search results for this problem are differently distributed compared to the latin hypercube distribution of results. This is something that also stood out in the first policy. Looking at figure B.8 it seems that more results have been generated compared to figure B.3. In the first policy the results were rather nicely distributed, whereas right now, there are some larger gaps visible in inertia and reliability. Comparing figure B.8 to figure B.6 these gaps are smaller for figure B.8. This leads to the same conclusion as for the first policy, that novelty search seems to be better at distributing.

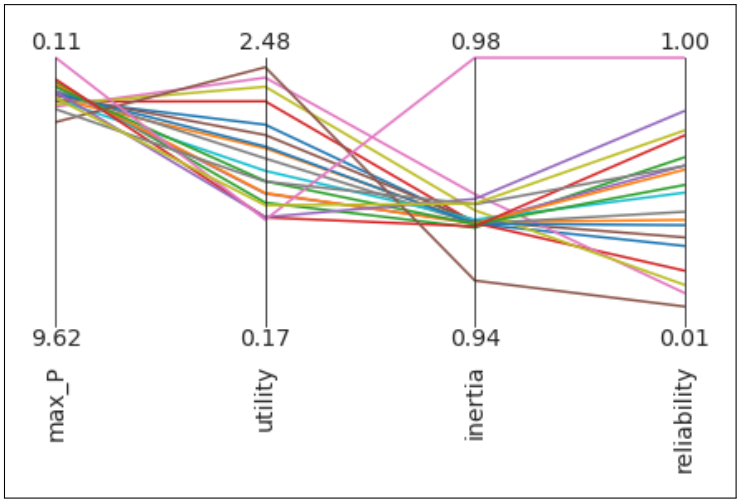

Figure B.8: Second policy parallel plot of novelty search results

Figure B.9 has a few eye-catchers. First of all, the mean has a significant influence over max_P. This is something that does not show in any of the previous feature matrices such as figures B.2, B.7, or B.4. Secondly, besides that mean and the delta row, the influences are quite close together. Other than the delta row, there are little similarities to be found in between figure B.9 and B.7.

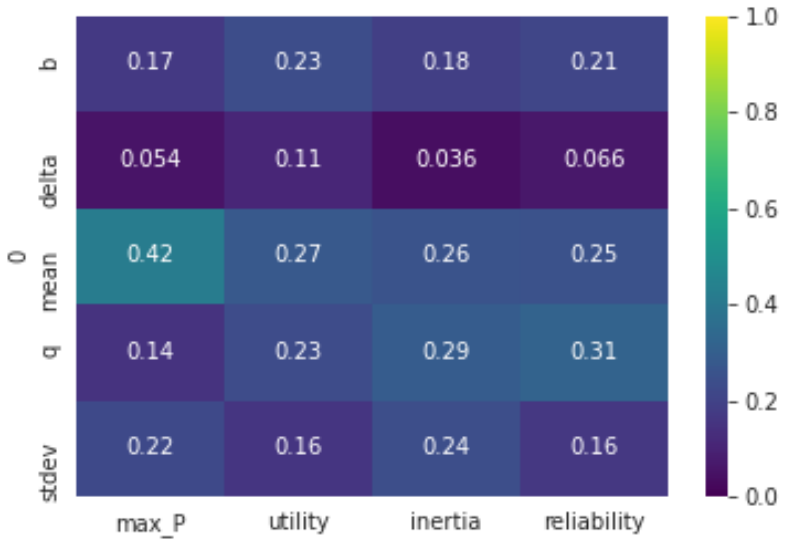

Figure B.9: Second policy feature scoring plot of novelty search results

In figure B.10 we can see the scatter plots for the different outcome variables of both search strategies. All rows containing utility seem to have a different results for novelty search in comparison to the latin hypercube samples. On the other hand, the combinations of inertia with max_P and reliability do give results that fall within the results of latin hypercube.

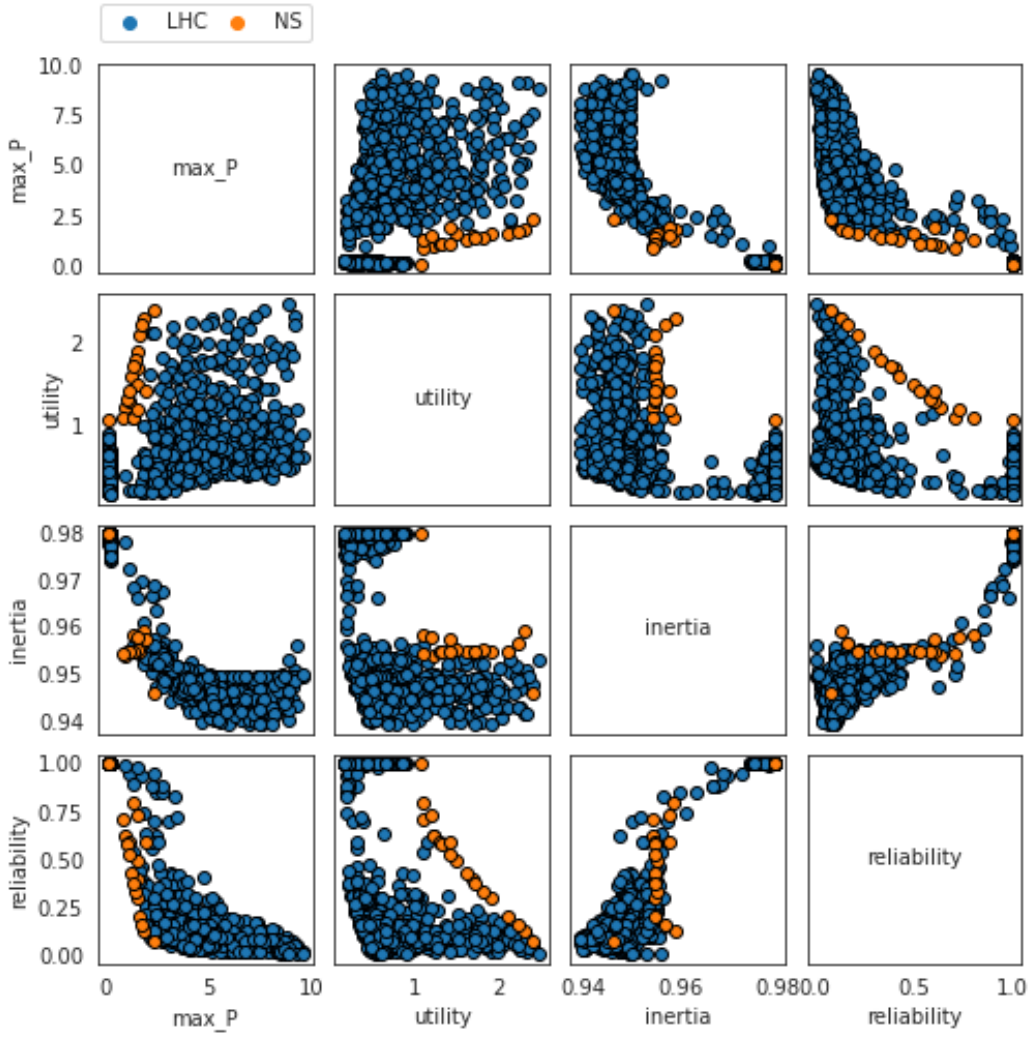

Figure B.10: Second policy scatter plot of all results

B.1.3. Policy 3

The values for the policy lever in this policy are: $c1 = 0.317884$, $c2 = 0.005661$, $r1 = 0.457124$, $r2 = 0.820411$ and $w1 = 0.646500$. This experiment will focus on the differences after changing the outcome goals for the evolutionary algorithm. The default: (min, max, max, max) for (max_P, utility, inertia and reliability) respectively will be used as well as minimizing and maximizing all outcomes.

Latin hypercube

Figure B.11 gives an overview of the results from the latin hypercube search strategy. Looking at the distribution of the results, there is nothing new. Some parts are overly explored, whereas other parts of the outcomes space are not explored or unable to be reached.

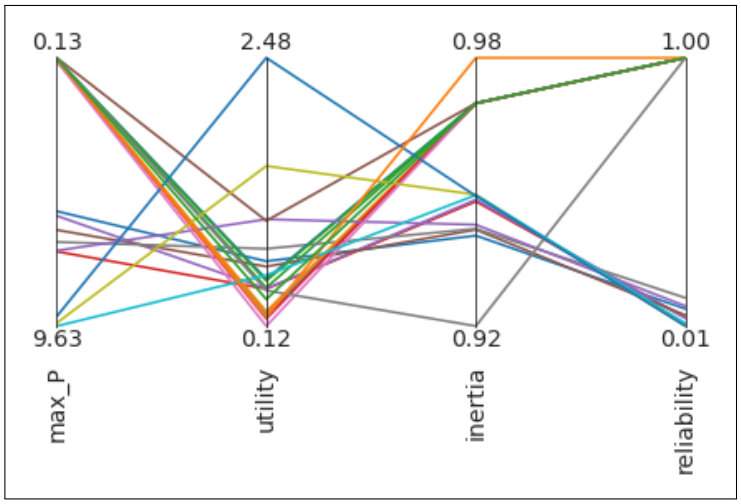

Figure B.11: Third policy parallel plot of latin hypercube results

The feature scoring matrix in figure B.12 shows similar data to that of figure B.7. Although the numbers are not identical, they are similar and only differ a few tenths. It will be interesting to see how this compares to the different novelty search matrices.

Figure B.12: Third policy feature scoring plot of latin hypercube results

Novelty search

In figure B.13 the results from the novelty search algorithm with default outcome goals are presented, looking very similar to fiure B.8. Figure B.14 presents the results where the outcome goals have been minimized and figure B.15 presents the results with maximized outcome goals. Figure B.15 seems to have the most dispersed set of results. The only outcome with a poor distribution is max_P, which contains a few gaps in the results. Figure B.14 has a very good distribution for the reliability outcome, but all others are centered around one point. Comparing figure B.13 to the others shows it's weakness in dispersion but strength in guidance into the right direction.

Comparing these figures to figure B.11 then especially figure B.15 give a better distributed overview. With one outlier figure B.13 has a lower dispersion, but improved distribution amongst that which is available. Figure B.14 is not interesting to look at, since maximizing the outcomes gives almost similar results. Making minimizing the outcome goals the worst option from this experiment.

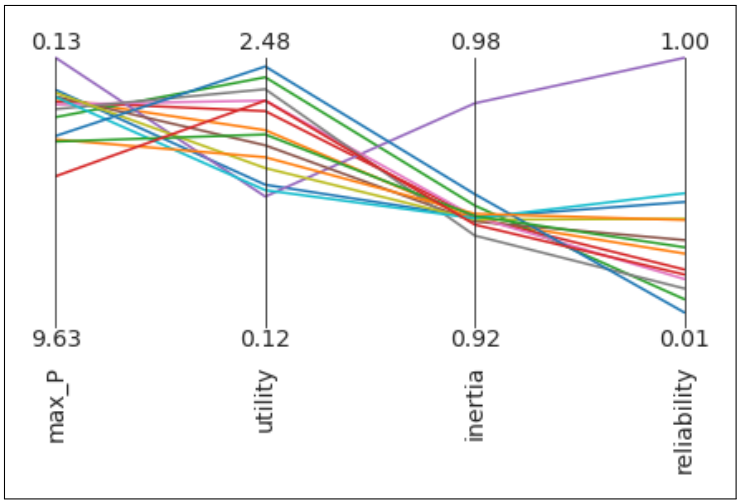

Figure B.13: Third policy parallel plot of novelty search default goals results

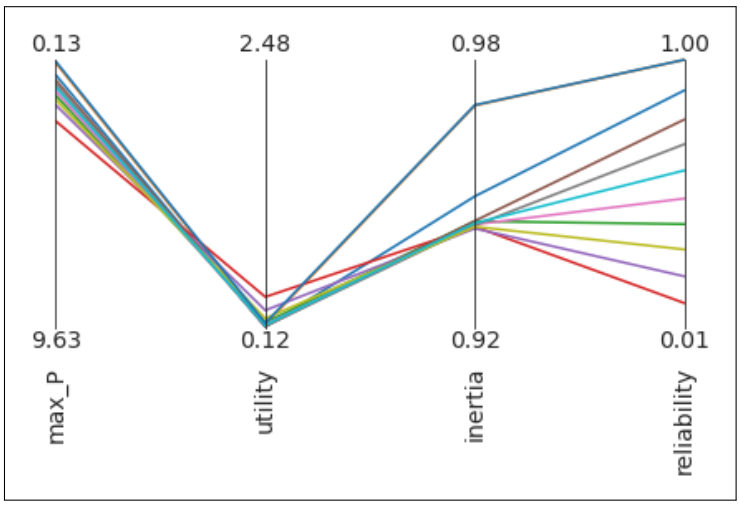

Figure B.14: Third policy parallel plot of novelty search minimized goals results

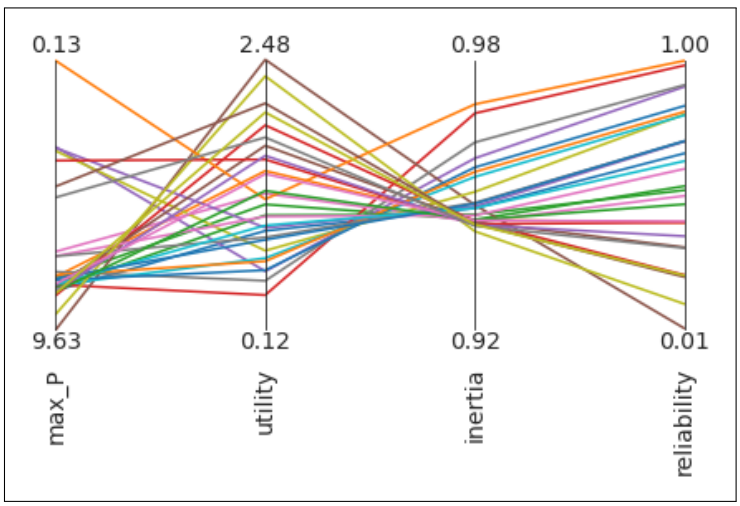

Figure B.15: Third policy parallel plot of novelty search maximized goals results

Figures B.16, B.17 and B.18 give the feature scoring matrices for the default, minimized and maximized outcome goals respectively. In figure B.16 the lack of influence from delta over any of the outcomes is surprising. The other feature scoring matrices also showed delta to be low in some cases, but never zero. Another, less prominent, eye-catcher is the relatively high influence of the mean. Figure B.17 presents us with a mix between figure B.2 and B.4. There are no better ways to describe this, the q has a relatively high influence over the outcomes inertia and max_P. That being said, other eye-catchers are not really present. Figure B.18 is quite dispersed, just like figure B.15. Relatively speaking it has some outliers, such as b and delta for max_P. Not all numbers are similar, but the difference in between numbers is not huge.

Figure B.16: Third policy feature scoring plot of novelty search default results

Figure B.17: Third policy feature scoring plot of novelty search minimized results

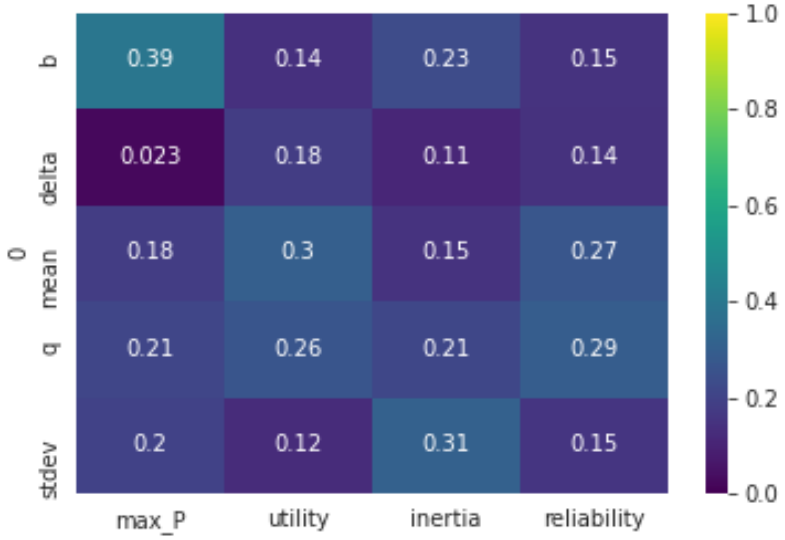

Figure B.18: Third policy feature scoring plot of novelty search maximized results

Figure B.19 gives an overview of the results of all used search strategies. Blue still represent latin hypercube sampling and orange still represents the default novelty search algorithm. Green represent the novelty search algorithm with all outcome goals maximized and red represents the all outcome goals minimized. For the plots containing reliability and inertia a lot of overlap exists between different search strategies, red on top of green, blue and orange. So much even that most of the blue results are not visible anymore. The figures containing reliability and max_P or utility clearly show the minimized and maximized novelty strategies boxing in latin hypercube. From the looks of it, the strategy with all goals maximized, and minimized both have some results that no other has explored. This is especially visible in the plot for max_P and reliability. The minimized goals also show some new results in the plot with reliability and utility. The default novelty search shows some new results in the plot with max_P and utility. The other plots show no new results from either of these. In some of the plots it is harder to see the default novelty search, due to it also being included in either the maximized or minimized version.

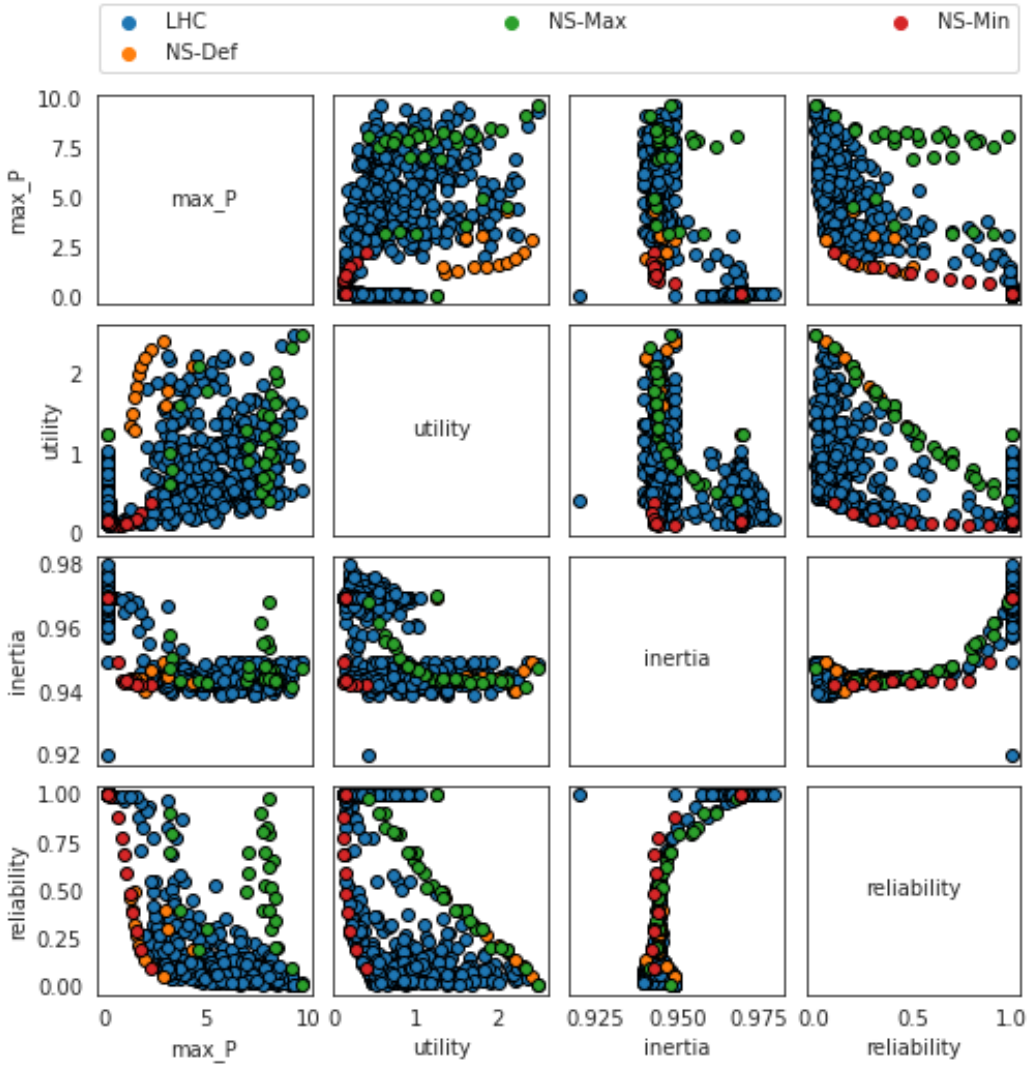

Figure B.19: Third policy scatter plot

B.1.4. Policy 4

The values for the policy lever in this policy are: $c1 = 0.835592$, $c2 = 0.258739$, $r1 = 1.855603$, $r2 = 0.102020$ and $w1 = 0.532751$. This experiment is identical to that of the third policy and will focus on the differences after changing the outcome goals for the evolutionary algorithm. The default: (min, max, max, max) for (max_P, utility, inertia and reliability) respectively will be used as well as minimizing and maximizing all outcomes. The only variation between this and policy 3 is the policy and therefore the values of the policy levers.

Latin hypercube

Figure B.20 shows the same parallel plot as always. For this policy it is remarkable to see that there is a decent dispersion, but the distribution is very poor. The only outcome that has a somewhat decent distribution is utility. The results of the remaining outcomes are focused around two or three points, which does not show any nice distribution.

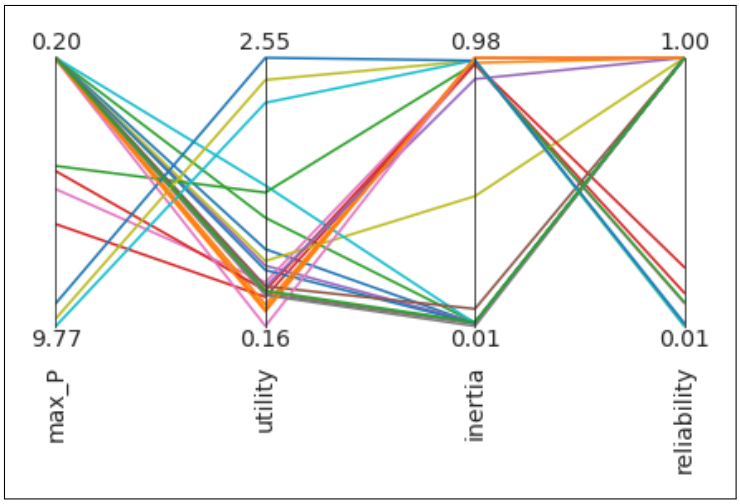

Figure B.20: Fourth policy parallel plot of latin hypercube results

Figure B.21 does not present us with any unexpected results. The numbers are different, but any feature matrix that has been generated using latin hypercube sampling is like this. The same variables hold a similar influence over the outcomes.

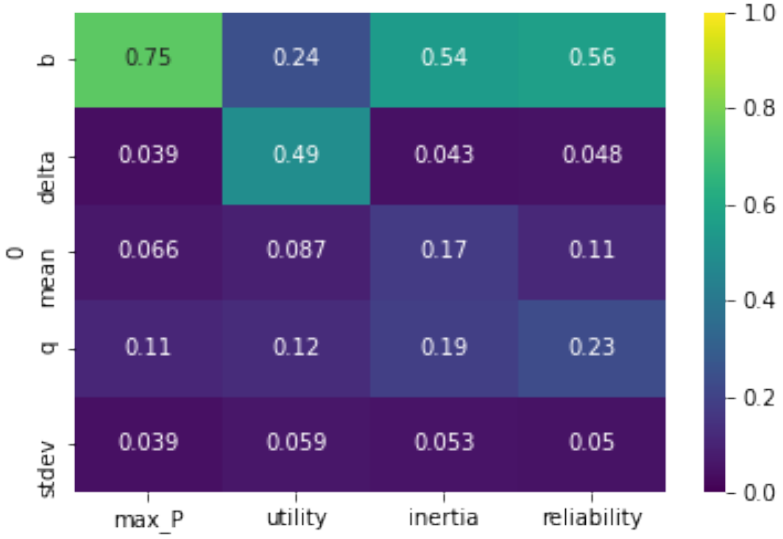

Figure B.21: Fourth policy feature scoring plot of latin hypercube results

Novelty search

The following figures B.22, B.23 and B.24 are similar to those of the third policy, figures B.13, B.14 and B.15. Figure B.24 with the outcome goals maximized shows a great dispersion and good distribution. With minimized goals, figure B.23, shows a great distribution for reliability. The default goals in figure B.22 present a good distribution over the dispersion that it has, except for the two outliers. For max_P and utility the dispersion is low in compared to novelty search with maximized goals or latin hypercube.

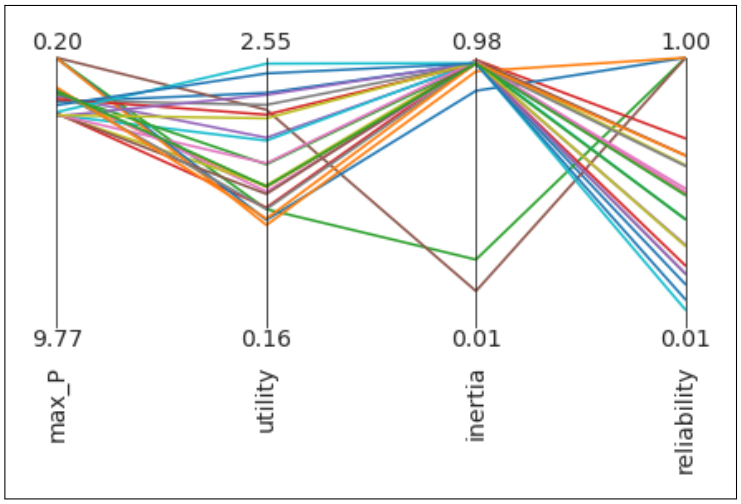

Figure B.22: Fourth policy parallel plot of novelty search default goals results

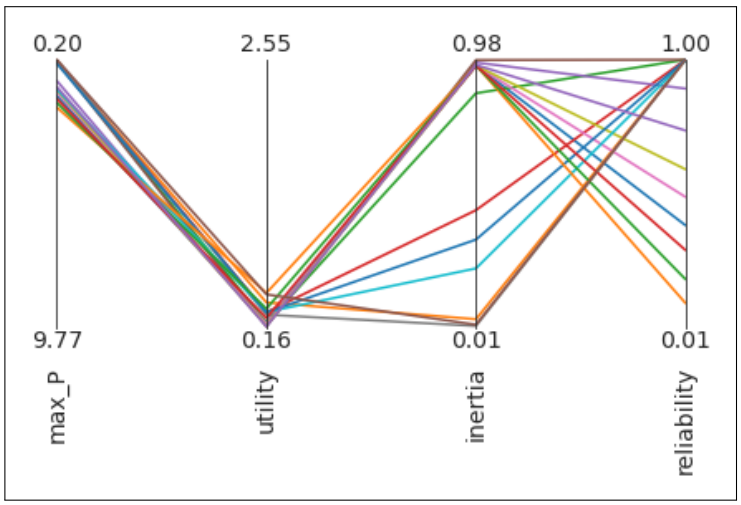

Figure B.23: Fourth policy parallel plot of novelty search minimized goals results

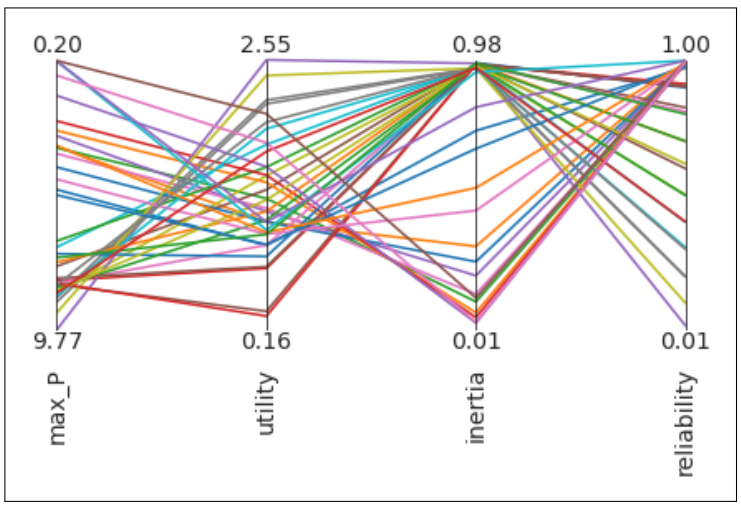

Figure B.24: Fourth policy parallel plot of novelty search maximized goals results

Figures B.25, B.26 and B.27 give the feature scoring matrices for the default, minimized and maximized outcome goals respectively. Just as for the third policy, there are too many differences to make a useful observation and comparison. There might be some similarities, such uncertainty b having a high influence on max_P inertia and reliability in figure B.27 and B.21. Other than that there are just too many differences.

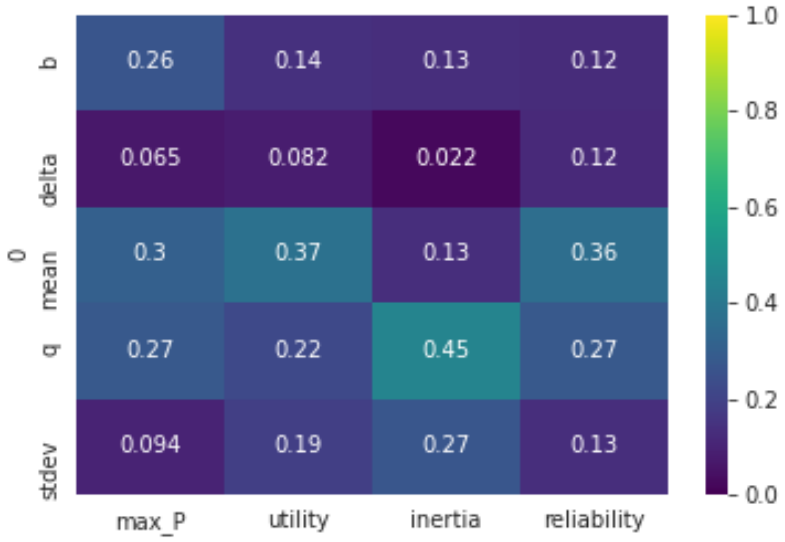

Figure B.25: Fourth policy feature scoring plot of novelty search default results

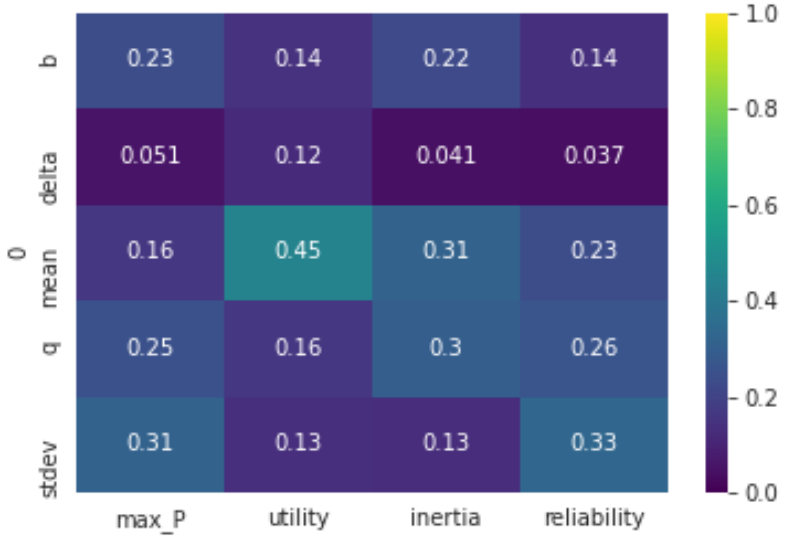

Figure B.26: Fourth policy feature scoring plot of novelty search minimum results

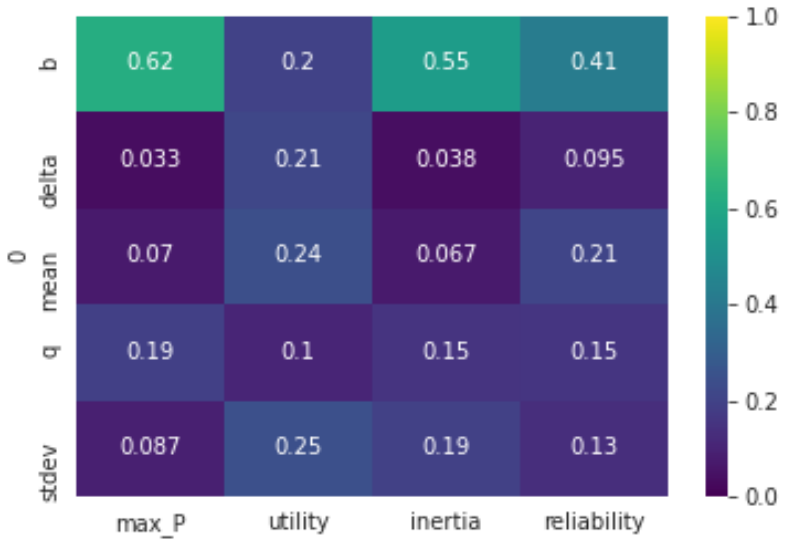

Figure B.27: Fourth policy feature scoring plot of novelty search maximum results

The scatter plot in figure B.28 presents some interesting plots to actually compare the different strategies. The plots with inertia and reliability stand out a lot, due to all results being on the same places. Every used strategy found those results, although not identical the results are very similar. The plot with max_P and inertia also stands out, due to the almost all results being on the same edges. It is interesting to see that the maximized outcomes show different areas in this graph. This suggests that the other strategies found similar results, whereas novelty search with maximized outcomes found new results. The other plots show similar plots to those in the scatter plot figure B.19 from the third policy.

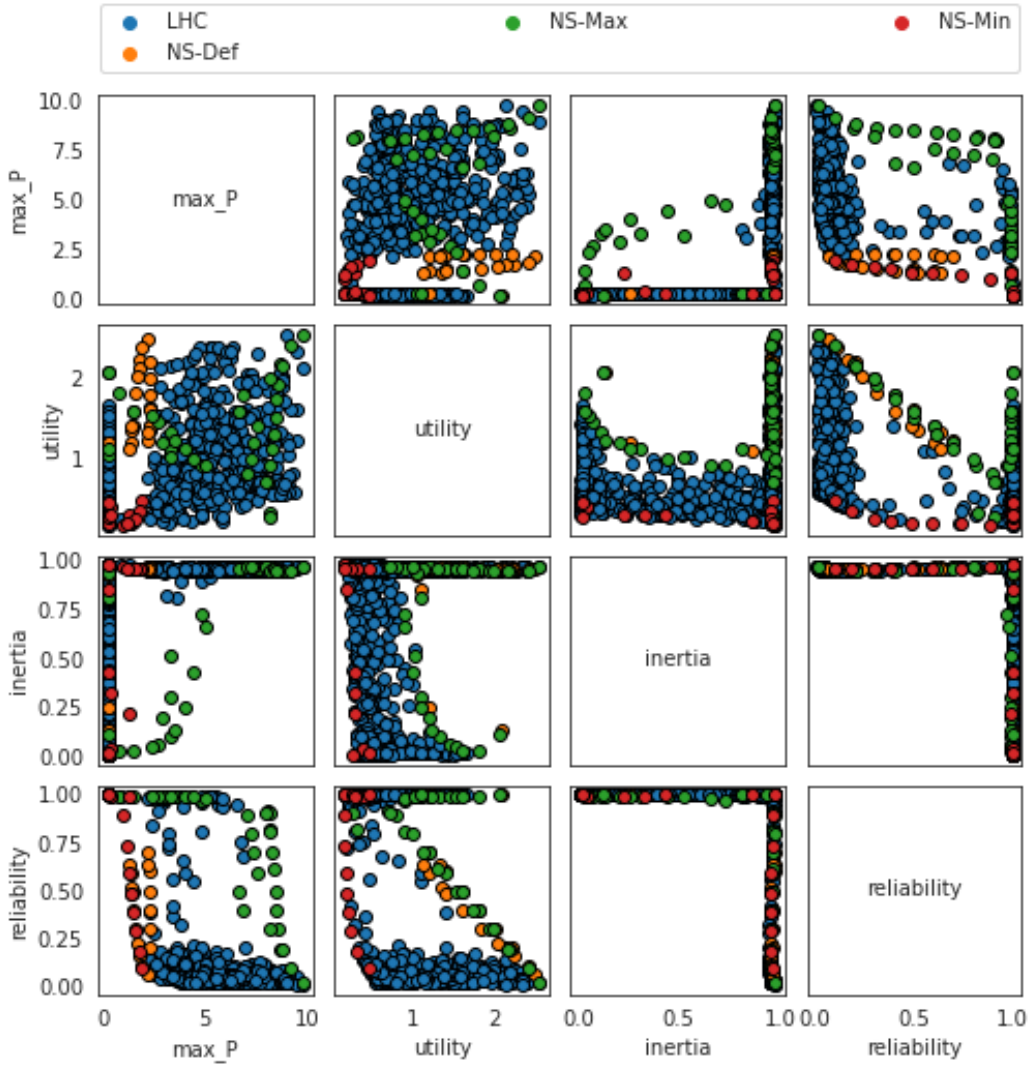

Figure B.28: Fourth policy scatter plot

B.1.5. Policy 5

The values for the policy lever in this policy are: $c1 = 1.057577$, $c2 = -0.061549$, $r1 = 1.948418$, $r2 = 1.939679$ and $w1 = 0.269329$. This is the final policy and also the final experiments, and therefore the final results as well. The experiment aims to see if the number of scenarios or functional evaluations are of influence on the results.

Latin hypercube

The results that are generated using latin hypercube, ranging from 1000 scenarios up to 1 million, are absolutely identical. They are so identical, that displaying these figures would mean, having the same figure 4 times. Although repetition is key to success, the only latin hypercube results that will be displayed are from 1000 scenarios. There are differences for the feature scoring matrix, as more results will lead to different influences, however that difference negligible. The only figure that will be displayed twice with 1000 scenarios and 1 million scenarios is the scatter plot. There are two reasons

for this. First of all, the differences are more like to be visible and secondly, it makes the comparison against previous scatter plots with minimal differences, since the focus should remain novelty search.

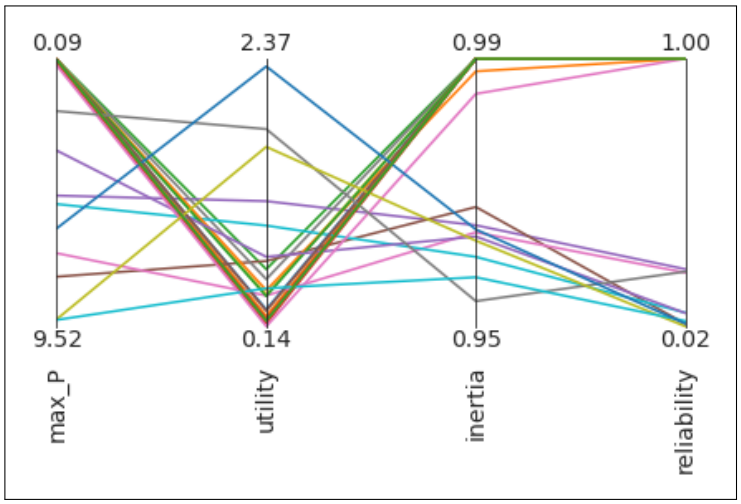

Figure B.29: Fifth policy parallel plot of latin hypercube results

Figre B.30 shows the same feature scoring matrix as the other policies. It is not identical, but it is so similar that it is not worth to look into it any more then this.

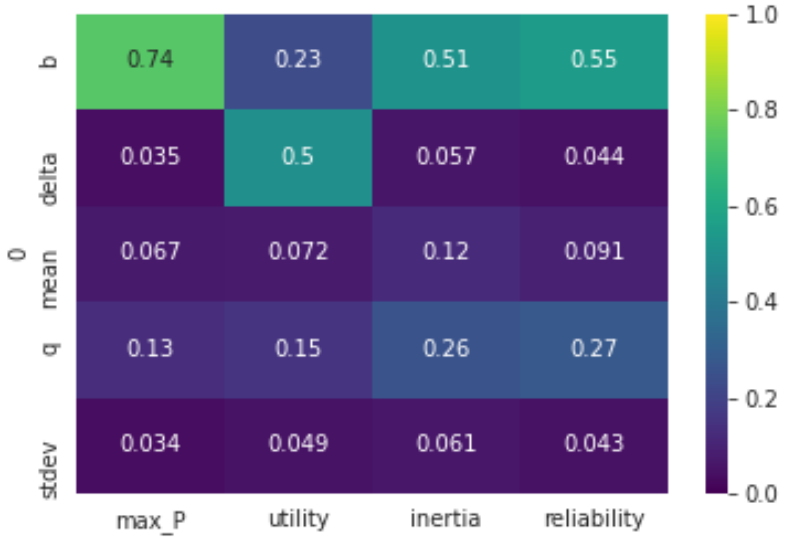

Figure B.30: Fifth policy feature scoring plot of latin hypercube results

Novelty search

For novelty search all figures will be plotted, as it shows some interesting developments. Looking at figures B.31, B.32, B.33 and B.34 the parallel plots for 1000, 10000, 1000000, and one million function al evaluations are presented. Figure B.31 with 1000 functional evaluations has a larger range, but correspondingly a worse distribution. Gradually improving up to figure B.34 with one million functional evaluations, the outcomes stabilize towards the goal and result in a improved distribution.

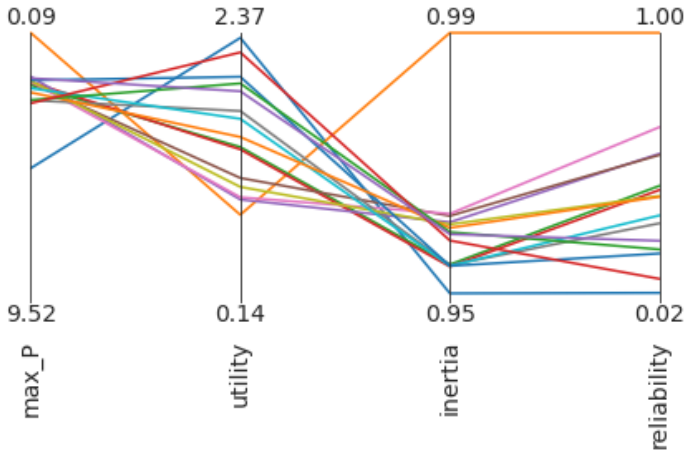

Figure B.31: Fifth policy parallel plot of 1000 novelty search results

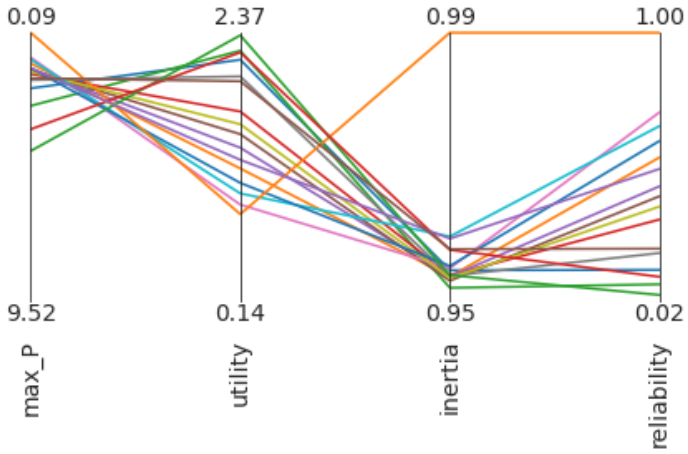

Figure B.32: Fifth policy parallel plot of 10000 novelty search results

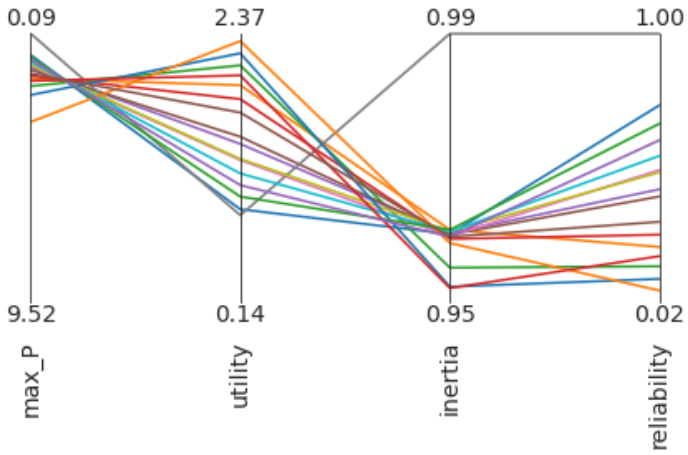

Figure B.33: Fifth policy parallel plot of 100000 novelty search results

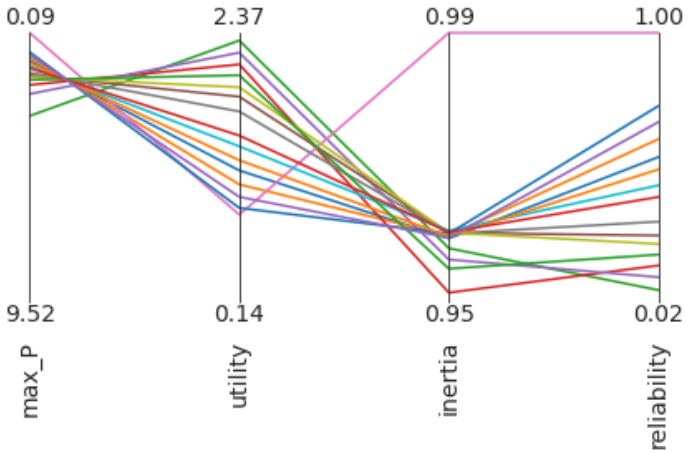

Figure B.34: Fifth policy parallel plot of 1 million novelty search results

Figures B.35, B.36, B.37 and B.38 show the feature scoring matrices for novelty search ranging from 1000 to one million functional evaluations. Although the parallel plots from figures B.31, B.32, B.33 and B.34 suggest that novelty search comes closer to the goals, it does so while changing the input variables in such a way that the differences between the feature scoring matrices are bigger than expected. This suggest that the outcomes, although closely related, have very different input parameters. This could be the reason that latin hypercube sampling is unable to identify these outcomes.

Figure B.35: Fifth policy feature scoring plot of 1000 novelty search results

Figure B.36: Fifth policy feature scoring plot of 10000 novelty search results

Figure B.37: Fifth policy feature scoring plot of 100000 novelty search results

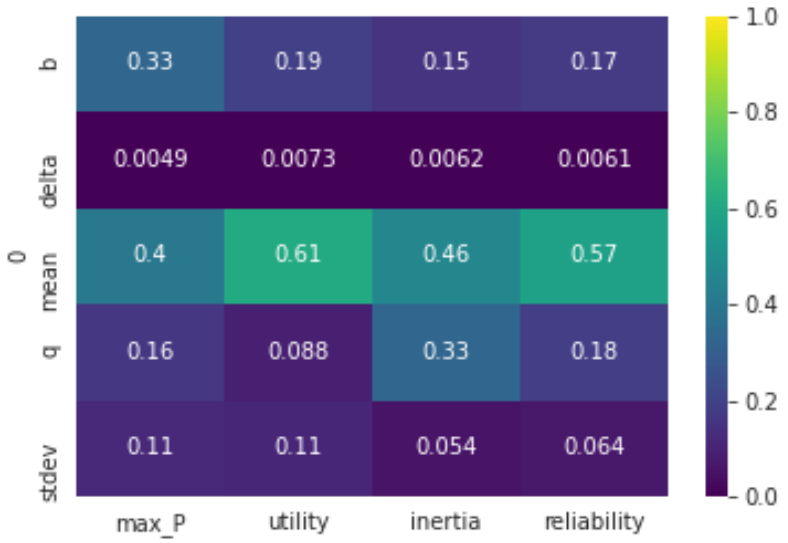

Figure B.38: Fifth policy feature scoring plot of 1 million novelty search results

Figures B.39 and B.40 present the scatter plots, with 1000 and one million functional evaluations respectively. These figures clearly show the similarities between 1000 functional evaluations and one million. With some of the results being barely visible, it shows the similarities between the different number of functional evaluations for novelty search. In most cases the one million experiment is furthest towards the goal, however the orange results are nowhere to be seen. This suggests that the other experiments contain most of the results of the experiment with 100000 functional evaluations.

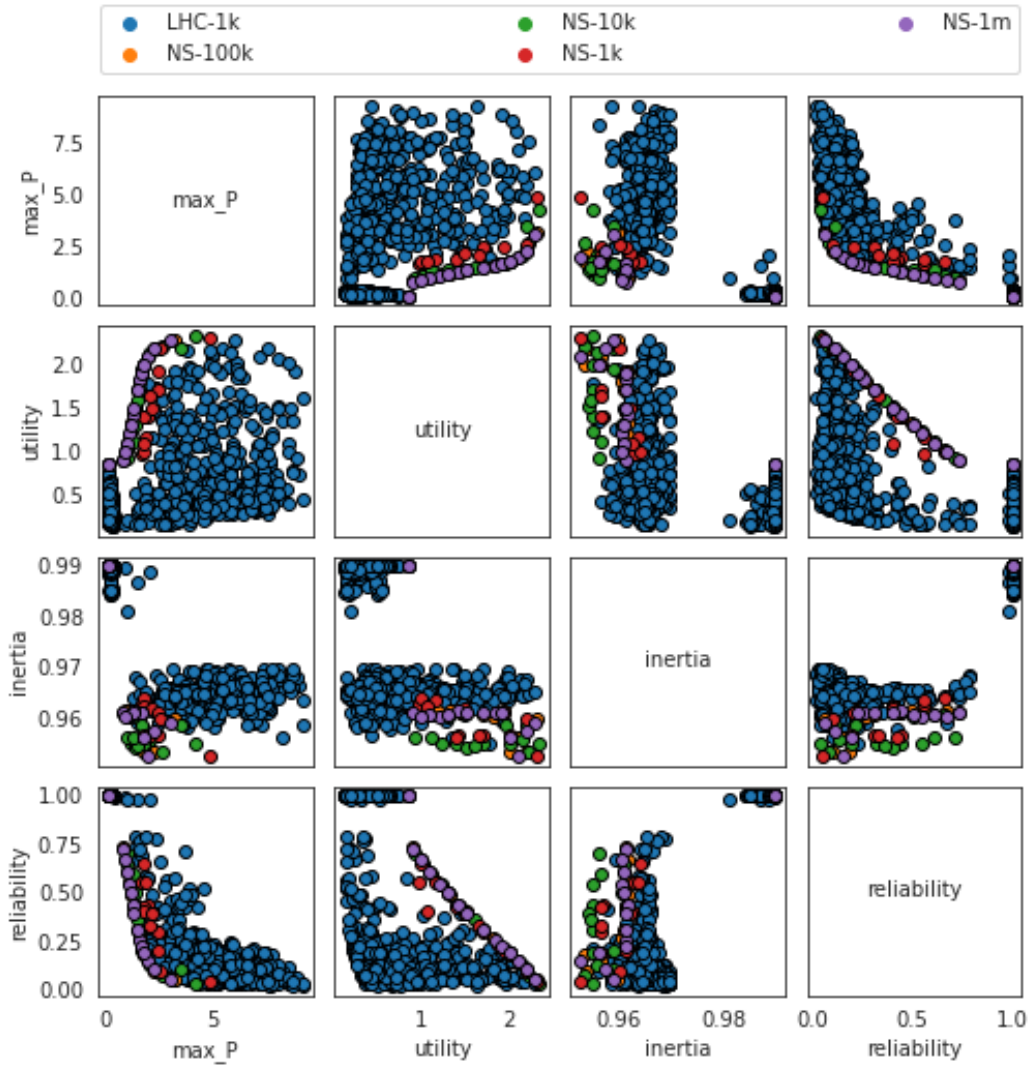

Figure B.39: Fifth policy scatter plot with 1000 latin hypercube results

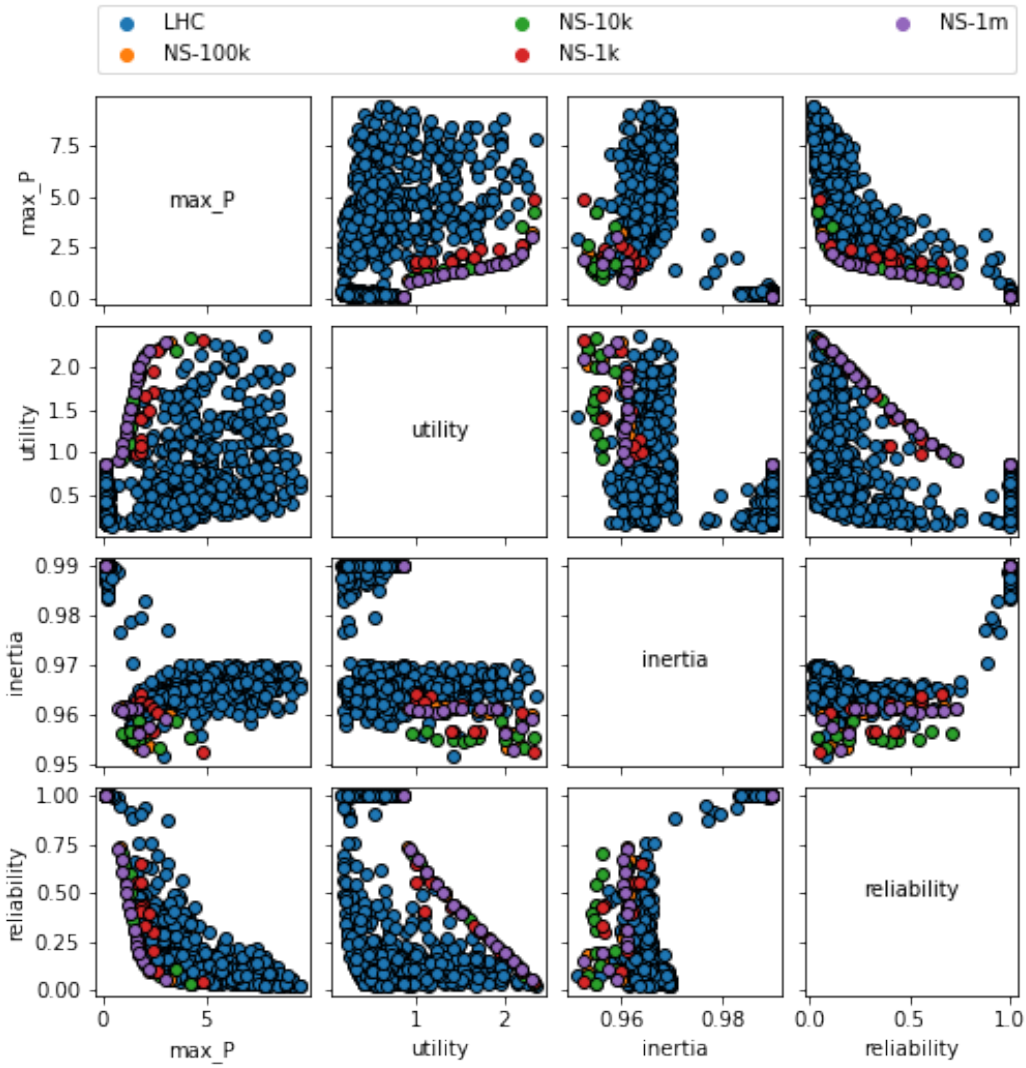

Figure B.40: Fifth policy scatter plot with one million latin hypercube results