Creating Energy System Design Options Using MGA and Bio-Inspired Metaheuristics

Application to a Large European Model

by

J.J.H. van den Berg

Student number: 4683196

To obtain the degree of Master of Science
in the Complex Systems Engineering and Management master's program
at Delft University of Technology.

First-supervisor:	Francesco Lombardi
Second-supervisor and chair:	Kateřina Staňková
Project Duration:	Sep 2024 - Mar 2025
Faculty:	Faculty of Technology, Policy and Management, Delft

TU Delft Delft University of Technology

Abstract

Planning and decision-making have become increasingly complex, especially in the energy sector. Modeling to Generate Alternatives (MGA) enables the creation of multiple alternative solutions, enhancing the decision-making process. However, when applied to large system optimization problems, the MGA method can become computationally cumbersome. Bio-inspired metaheuristics, a branch of artificial intelligence, have the potential to overcome these computational limitations by applying metaheuristic methods that can solve multiple model solutions simultaneously. Currently, a sophisticated integration of such heuristics with MGA, specifically focused on energy system optimization, has not yet been realized.

This thesis aims to answer the question: How does a combination of MGA and bio-inspired heuristics, aimed at optimizing energy system design, compare with existing deterministic MGA methods? To address this question, a metaheuristic-MGA algorithm was developed and compared with existing spatial energy system MGA results.

First, a literature review was conducted to determine the most suitable metaheuristic for this study. The review confirmed the scarcity of literature on the combination of metaheuristics and MGA in energy systems. However, relevant studies applying a metaheuristic-MGA approach to general optimization problems were identified. Based on these findings, a genetic algorithm (GA) was selected as the most appropriate method for this thesis. The mathematical formulation presented in the literature was adapted to fit the spatial optimization problem of energy systems.

The complete mathematical process of the GA-MGA algorithm was developed in Python. Next, a small energy system test model was built in Calliope to evaluate the performance of the developed GA-MGA algorithm. The algorithm's parameters were further fine-tuned using existing parameter-tuning methods, performance measurements, and assessments of the computational time required to complete the algorithmic process.

Before applying the developed GA-MGA algorithm to a large-scale model, it needed to be scaled to prevent errors or computational inefficiencies when generating results. It was determined that the desired resolution was not feasible due to the excessive computational time required for its completion. Instead, a time-masking method was applied to the resolution, preserving high-resolution characteristics while improving computational efficiency.

The GA-MGA algorithm was then applied to a large European energy system model, and the results were compared to existing MGA results. However, due to differences in resolution, the GA-MGA-generated results did not meet the standards of the existing MGA results, making direct comparisons less robust and reliable than desired. The comparison revealed a significant difference in battery capacity deployment, with the GA-MGA solutions deploying higher quantities of battery capacity. The lack of spatial distribution data for the large model was solved by comparing the GA-MGA results to a plot of the existing MGA results. While this was not an ideal comparison, it provided an opportunity to analyze spatial deployment differences between the GA-MGA and the existing MGA results. The comparison showed that the GA-MGA algorithm favored high-capacity deployment at specific locations, whereas the existing MGA results exhibited a more diverse capacity distribution.

The limitations of the results primarily stemmed from shortcomings in spatial comparison and differences in resolution between the two modeling techniques. Another key limitation was the algorithm's structure, which presents several opportunities for improvement in optimizing the GA-MGA approach. Despite these challenges, the theoretical combination of GA-MGA demonstrated promising potential. Future research should focus on enhancing the algorithm's performance and conducting more in-depth comparisons with MGA results to fully evaluate its effectiveness. Further research in this area could expand access to the MGA method for tackling large, complex problems, ultimately contributing to more effective planning and decision-making processes. This, in turn, would support efforts to address major societal challenges more efficiently.

All the code used for this thesis can be found via the following GitHub links:

- Test model: `https://github.com/Jacobvdberg228/energy-system-genetic-MGA`
- Large model: `https://github.com/Jacobvdberg228/Large-model-MGA-GA`

Contents

Nomenclature

The first table shows the abbreviations that are used in the thesis. The second table shows the symbols used to mathematically explain the algorithm.

Abbreviations

Abbreviation	Definition
DEAP	Distributed Evolutionary Algorithm in Python
EA	Evolutionary Algorithm
EAGA	Evolutionary Algorithm to Generate Alternatives
EC	Evolutionary Computing
ESM	Energy System Model
ESOM	Energy System Optimisation Model
GA	Genetic Algorithm
HSJ	Hop-Skip-Jump
MGA	Modeling to Generate Alternatives
MOEA	Multi-Objective Evolutionary Algorithm
MOGA	Multi-Objective Genetic Algorithm
NSGA-II	Non-Dominated Sorting Genetic Algorithm II
PSO	Partical Swarm Optimisation
RES	Renewable Energy Sources
SI	Swarm Intelligence
SPORES	Spatially Explicit Practically Optimal Results

1

Introduction & Background

1.1. Introduction

The formulation of policies and decision-making processes is inherently challenging, particularly in complex socio-technical domains such as infrastructure planning for the energy transition [1]. One of the primary sources of complexity is the involvement of multiple stakeholders at both national and supranational levels, each with their own priorities, interests, and constraints [2]. These interactions create conflicting objectives, making it difficult to determine a single optimal course of action. To address these challenges, multi-objective optimization models are widely used to assist decision-makers by quantifying trade-offs between competing goals [3]. These models generate a set of non-inferior (Pareto-optimal) solutions, where no objective can be improved without negatively affecting at least one other. However, while Pareto-optimal solutions provide valuable insights, they also introduce limitations: they do not account for real-world constraints such as political feasibility, public acceptance, or long-term adaptability [4, 5].

1.1.1. Beyond Pareto-Optimality: MGA

Relying solely on Pareto-optimal solutions may restrict decision-makers to a narrow subset of mathematically optimal solutions, potentially overlooking more practical but slightly inferior alternatives. To mitigate this issue, an alternative modeling approach known as Modeling to Generate Alternatives (MGA) has been developed [6]. Rather than searching for a single best solution, MGA systematically generates multiple diverse solutions that remain near-optimal but differ significantly in their decision variables.

MGA is particularly valuable for decision-making in energy system planning, where different stakeholders may prioritize different aspects of the transition, such as cost efficiency, technological diversity, or regional equity. By presenting multiple viable pathways, MGA allows decision-makers to compare and select strategies that align best with broader policy objectives and stakeholder preferences. This MGA approach has already been applied in various fields, including wastewater treatment and energy system optimization, demonstrating its effectiveness in highly nonlinear decision spaces [7, 8].

1.1.2. Computational Challenges of MGA in Energy System Models

Despite its advantages, MGA becomes computationally demanding when applied to large optimisation problems such as large-scale energy system models (ESMs). With increasing spatial and temporal resolution and greater technological detail, the number of variables and constraints grows exponentially [9]. Traditional MGA approaches compute near-optimal alternatives iteratively, which leads to substantial computational cost, making it impractical for complex decision problems [10, 11]. A promising solution to this challenge is the integration of bio-inspired metaheuristics, which have been argued as to be a possibility for significantly improving computational efficiency and search performance in optimization problems [12].

1.1.3. Enhancing MGA with Bio-Inspired Metaheuristics

Metaheuristic algorithms, particularly bio-inspired methods such as genetic algorithms (GA) and particle swarm optimization (PSO) have demonstrated strong performance in multi-objective optimization [10, 13]. Unlike classical MGA methods that rely on iterative perturbations, metaheuristics explore the search space adaptively, making them more efficient in handling high-dimensional, multi-objective problems.

1.1.4. Research Objective

Although bio-inspired metaheuristics have already been applied to multi-objective energy system models [13], their integration with MGA remains an under-explored area [12]. Given the computational challenges of traditional MGA, leveraging metaheuristic-driven alternative generation could significantly enhance both efficiency and scalability in complex optimization problems. This research aims to bridge that gap by exploring whether a metaheuristic-driven MGA approach can efficiently generate diverse, near-optimal solutions in energy system modeling. By integrating bio-inspired metaheuristics with MGA, this research seeks to enhance the practical usability of alternative generation methods in complex energy system planning. If successful, this approach could provide decision-makers with more computationally feasible and diverse solutions, ultimately supporting more robust and adaptable policy-making in the energy transition.

1.1.5. Research Outline

The goal of this thesis is to develop a metaheuristic-MGA combination designed to optimize spatial capacity distribution in an energy system. To achieve this, the thesis will first explore the concepts of MGA and metaheuristics in general, as well as their applications in energy systems. Once a solid understanding of these concepts has been established, the research question and sub-questions will be discussed in detail. This will be followed by the methodology section, outlining the steps required to answer each sub-question. The methods will then be implemented, and the results will be presented, leading into the discussion section, where the findings will be analyzed. Finally, each sub-question will be answered, and a conclusion will be drawn to address the main research question.

The methodology will consist of a literature review and a model-based approach. The literature review will help determine the most suitable metaheuristic to combine with the MGA method for this thesis. The model-based approach will involve designing a small-scale test model to assess whether a metaheuristic-MGA combination is feasible and, if so, how well it performs. If successful, the developed algorithm will be refined and adapted for application to an energy system model that already incorporates MGA, as established in previous research. Finally, the enhanced algorithm will be applied to the larger model, and its results will be compared with existing MGA outputs to evaluate its effectiveness.

1.1.6. Link to CoSEM Master Program

This research applies mathematical optimization and modeling to analyze and optimize the spatial energy production of a complex energy system. As such, it aligns well with the Complex System Engineering and Management (CoSEM) master's program, which extensively focuses on complex system analysis and optimization. Expanding knowledge in energy system optimization can contribute to better managerial decision-making and support the integration of renewable technologies into complex energy networks.

1.2. Key concepts

1.2.1. Modeling to generate alternatives (MGA)

Modeling to Generate Alternatives (MGA) was first introduced by Brill [4] as a response to the limitations of traditional optimization methods in addressing complex decision-making problems, particularly those involving political and societal trade-offs. Unlike conventional optimization, which seeks a single best solution, MGA generates multiple, diverse solutions that are near-optimal, allowing decision-makers to explore alternative strategies that reflect different priorities, constraints, or perspectives. As Liebman [14] emphasized, optimization models should support decision-making by providing insight rather than dictating rigid solutions.

MGA is particularly relevant for problems involving multiple stakeholders with conflicting objectives, where a single optimal solution may fail to capture the full range of trade-offs necessary for effective decision-making. To formalize this approach, Brill developed the Hop, Skip, and Jump (HSJ) method, which systematically generates alternative solutions that remain close to optimal but vary significantly in their decision variables [6]. The HSJ method begins by computing an initial near-optimal solution, after which it generates a maximally different alternative while ensuring that the new solution remains within acceptable bounds. This process is iterated until a diverse set of alternatives is obtained.

Over time, MGA has been widely applied in various fields, including environmental management [15] and energy system planning [8], where flexibility in decision-making is essential. Unlike multi-objective optimization, which primarily focuses on exploring the Pareto front, MGA emphasizes generating solutions that are as distinct as possible within the decision space, providing decision-makers with broader options beyond strict mathematical optimality.

In real-world decision-making, numerical optimization alone is often insufficient, as many practical considerations extend beyond what can be captured in a mathematical model. Decision-makers must take into account qualitative aspects such as political feasibility, social acceptance, and environmental justice [16]. A key illustration of this is provided by Zechman [17], who demonstrated that a model optimizing only a single objective, such as cost efficiency, may produce an optimal mathematical solution. However, when additional real-world criteria such as social feasibility are introduced, the preferred decision may shift to a different alternative.

This is illustrated in Figure 1.1, where a model is maximizing a single objective Z_1, leading to the optimal mathematical solution Z^*. However, when considering a second, qualitative objective Z_2, such as political or societal feasibility, the best choice shifts to X^C. While Z^C might seem like an inferior solution when viewed through the lens of Z_1 alone, it may in fact be the optimal decision in practice. This observation highlights the importance of generating multiple solutions, as focusing solely on the mathematically optimal outcome can overlook alternative solutions that better align with real-world considerations.

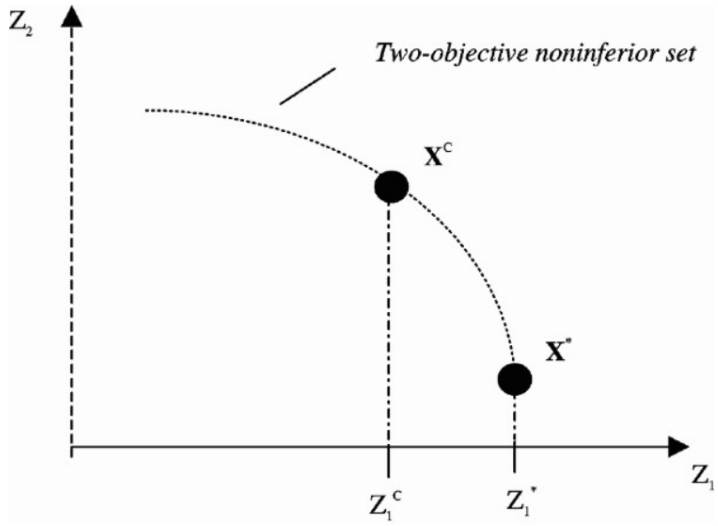


Figure 1.1: Objective space of Z_1 and Z_2. Source: Zechman [17].

By moving beyond a purely optimization-driven approach, MGA enhances decision-making by highlighting trade-offs and increasing solution diversity, ensuring that potential constraints and shifting priorities are not overlooked.

Despite its advantages, MGA faces significant computational challenges when applied to large-scale energy system models. Traditional MGA approaches require solving multiple optimization problems, systematically modifying constraints to generate alternative solutions. As spatial and temporal resolution in energy system models increases, this process becomes computationally costly [9, 10, 11].

To address these computational limitations, Lombardi et al. [12] propose a fundamental shift in approach by integrating bio-inspired metaheuristics with MGA. Unlike traditional MGA methods that recompute solutions from scratch, metaheuristic algorithms can efficiently generate multiple alternatives in a single computational run [18]. By altering selection mechanisms and search operators in response to the performance landscape, adaptive search techniques enable efficient exploration of the decision space

This integration allows for a more balanced trade-off between computational feasibility and solution diversity. Instead of relying on repeated full-scale optimizations, metaheuristics can iteratively refine existing solutions, uncovering diverse alternatives without excessive computational overhead. By merging MGA with bio-inspired algorithms, decision-makers can generate alternative solutions more efficiently, enabling the application of MGA to large-scale energy system models in ways that were previously computationally impractical. This approach represents a significant advancement in the way alternative solutions are generated and evaluated in complex optimization problems.

1.2.2. Bio-Inspired Metaheuristics

Nature has long demonstrated intelligence through self-learning, resilience, and efficiency, allowing organisms and natural processes to navigate and solve complex challenges. These principles have inspired metaheuristic algorithms, which use advanced search techniques to solve optimization problems beyond traditional methods like linearization [19, 20]. Advancements in computational power have enabled exploration of multi-modal and multi-objective optimization, where metaheuristics have proven competitive with classical methods [21, 22]. One major category is Evolutionary Computation (EC), encompassing algorithms such as Genetic Algorithms and Evolutionary Programming [23, 24]. These algorithms follow a common computational process:

- Generate a population of feasible solutions
- Evaluate their properties
- Select the best candidates
- Apply genetic operators to create a new generations
- Repeat until an optimal solution is found

Another significant offshoot is Swarm Intelligence (SI), which models collective behaviors observed in nature, such as ant colonies or bird flocks [25]. Unlike individual agents, swarms exhibit intelligence through cooperation, making them adaptable, robust, and efficient [26]. SI has a lot of resemblance with EC such as population initialization and fitness, but does not incorporate mutation and crossover.

Metaheuristics, including EC and SI, have found widespread application in solving complex optimization problems. Their ability to efficiently explore vast solution spaces makes them particularly valuable in stochastic optimization and multi-objective optimization. Stochastic optimization involves solving problems under uncertainty. Unlike deterministic methods, stochastic optimization integrates randomness into the search process, enabling solvers to handle dynamic and uncertain environments. Bio-inspired heuristics excel in this domain by adapting to fluctuations in the fitness landscape, allowing them to continuously generate optimal solutions even as conditions change [27, 28]. Another critical research area is multi-objective optimization, where bio-inspired metaheuristics are particularly effective in generating Pareto-optimal solutions.

Many of these algorithms take an evolutionary approach based on EC, forming for example a multi-objective evolutionary algorithm (MOEA) [29]. They incorporate elitism, ensuring that top-performing solutions are preserved across generations [30]. The goal is to find a Pareto front, a set of solutions where no improvement in one objective can be made without sacrificing performance in another. Many types of MOEAs exist, which are extensively discussed by Bechikh et al. [31]. Two of the most popular methods are the Pareto-based approach, which

includes the Non-Dominated Sorting Genetic Algorithm II (NSGA-II), and the decomposition-based approach, which consists of the MOEA/D algorithm.

But there are also SI algorithms that are focused on multi-objective optimisation. The most well known is the particle swarm optimisation (PSO) technique. Simply put, PSO mimics swarm intelligence principles. By leveraging personal and global best positions, it efficiently finds optimal solutions in complex search spaces [32]. By adapting PSO for multi-objective optimization, the algorithm can efficiently handle conflicting objectives and identify Pareto-optimal fronts.

Despite the success of metaheuristics, research often emphasizes developing new and innovative algorithms over conducting thorough theoretical analysis to justify their effectiveness. As a result, many algorithms are introduced without deep theoretical validation, making it difficult to distinguish genuinely impactful advancements from those that are simply novel [33, 34]. While nature offers countless heuristic mechanisms, their adaptation must be grounded in solid theoretical foundations to ensure effective and justified design choices.

Bio-inspired metaheuristics are increasingly being applied to energy systems modeling (ESM) to address challenges such as electricity demand, energy storage, and transmission limitations. Given the multiple objectives involved in ESM, algorithms like NSGA-II have already proven successful in this field [13]. Despite these successes, the use of metaheuristics in combination with ESM remains a promising research direction with significant potential for impact [35].

1.2.3. Energy System Models

Energy system models (ESMs) provide different perspectives and approaches to understanding the evolution of energy systems, guiding decisions in areas such as policy, infrastructure, and market dynamics. These models can be broadly categorized into four paradigms: (1) energy system optimization models (ESOMs), (2) energy system simulation models, (3) electricity and market power system models, and (4) qualitative and mixed-method scenarios [36]. Each paradigm serves a distinct role in addressing complex energy system challenges. Among them, ESOMs are the most widely used, employed by major organizations at all levels of the energy system [37].

ESOMs are generally multi-objective, as they must balance competing goals such as cost, emissions, system reliability, and policy constraints. Due to this complex characteristic, metaheuristic algorithms and MGA have already been applied to multi-objective energy optimization problems [38, 13]. MGA enables the exploration of near-optimal solutions, revealing trade-offs and uncertainties in energy planning, while metaheuristics efficiently navigate large and complex optimization spaces.

Although both approaches have been independently applied to energy system models, a combined application of MGA and metaheuristics within an ESM has not yet been explored. As discussed earlier, studies suggest that such an integration could improve the computational efficiency and flexibility of energy system optimization [**imanrad**, 12, 18].

1.3. Research Question

Energy system designers must navigate complex decision-making processes, balancing multiple objectives such as cost, emissions, and geopolitical factors. One of the key challenges they face is deploying specific technologies at optimal locations, a task complicated by the need to balance economic constraints with technological feasibility, regulatory policies, and evolving public sentiment. Additionally, the rapid evolution of technology and shifting public policies introduce further uncertainty, making the range of possible decisions vast and dynamic. Geographical constraints and specific technological requirements further complicate system design choices.

Traditional optimization methods often struggle to accommodate the vast uncertainties and conflicting objectives present in energy systems. To address this, MGA have emerged as a powerful approach for generating diverse and practical solutions. MGA has been successfully applied to energy system optimization [39, 40] and has also proven effective for problems involving the spatial distribution of technologies [38, 12].

Unlike fields where mathematical models can precisely predict outcomes, energy systems involve numerous unmodeled uncertainties, making absolute optimization impractical. Instead, decision-makers benefit more from a diverse set of near-optimal solutions that capture different trade-offs. Which is precisely what MGA offers. By generating multiple feasible alternatives rather than a single "best" solution, MGA provides policymakers and engineers with greater flexibility in long-term energy planning. However, MGA is not without its limitations, particularly in terms of high computational costs, which can make large-scale optimization impractical. Additionally, efficiently exploring the solution space remains a challenge, as choice of MGA methods can lead to premature convergence or excessive runtime. Future research should focus on enhancing computational efficiency and adaptive exploration mechanisms to make MGA a more scalable and accessible tool for energy system optimization.

Metaheuristics present a promising approach to addressing computational complexity while maintaining the ability to generate valuable MGA solutions. But the combination of MGA and metaheuristics applied to ESMs stays an unexplored field of research. MGA methods trying to optimize spatial distribution of energy could highly benefit from a successful integration with metaheuristics. This thesis will therefor try to answer the following research question:

How does a combination of modeling to generate alternatives (MGA) and bio-inspired metaheuristics, aimed at the optimisation of spatial energy distribution, compare with existing deterministic MGA methods?

1.4. Research Approach

This research follows a two-phase approach: a systematic literature review to establish a foundation and a model-based methodology to develop and test a metaheuristic MGA algorithm.

First, a literature review will identify current advancements in combining MGAs with bio-inspired metaheuristics, particularly in energy system optimization. The review will not only clarify ongoing research efforts and available resources but also determine which type of algorithm is most suitable for this study. This insight will directly shape the algorithm's design, ensuring its relevance and effectiveness.

Second, a modeling approach will be employed to develop, test, and validate the algorithm. Initially, a prototype combining an MGA with bio-inspired heuristics will be created using Python and the energy system will be build with Calliope. Calliope is an open-source modeling framework making it easy to build energy system models at varying scales. This algorithm prototype will be applied to a small-scale energy system to assess its feasibility and refine its parameters. Subsequently, the optimized algorithm will be integrated into a large-scale Calliope model, evaluating its scalability and effectiveness in complex energy systems. Finally, simulation results will be compared against existing benchmarks using Python and Excel to assess performance and reliability.

This structured approach ensures a strong theoretical foundation and rigorous model validation, addressing the research question systematically.

1.5. Sub Questions

To systematically address the research question, the following sub-questions guide the study. These questions break down the research into key focus areas, ensuring a structured investigation of algorithm selection, configuration, scalability, evaluation, and practical applicability.

1. Which bio-inspired metaheuristic algorithm is best suited for the method being used?
2. How can the tuning of algorithmic parameters optimize the performance of the selected metaheuristic for implementing a successful MGA optimization technique?
3. How will metrics be used to evaluate the algorithm's performance, and how do these comparisons with existing methods inform its applicability?

2

Methods

This section outlines the methods used to answer the sub-questions. It begins with a discussion of the methodological framework, providing a brief explanation of the chosen methods and their relevance to the research question. Following this, each proposed method is explained in detail, including how it will be implemented.

2.1. Methodological Framework

To address the research question, "How does a combination of MGA and bio-inspired metaheuristics, aimed at optimizing energy system design, compare with existing deterministic MGA methods?", a new algorithm that integrates an MGA approach with a metaheuristic must be developed. Several sub-questions guide this process:

2.1.1. Selection of a Suitable Bio-Inspired Metaheuristic

The first step in designing the algorithm is to identify the most suitable metaheuristic for integration with MGA in this study. Given the variety of available metaheuristics it is essential to determine which approach is best suited for the specific requirements of this thesis. To achieve this, a systematic literature review will be conducted to examine studies on the combination of MGA and metaheuristics, with a particular focus on their applications in optimization problems, especially in energy system optimization. The objective is to select the most relevant and practically applicable method based on existing research, acknowledging that while other potentially superior methods may exist, the choice will be guided by the best available knowledge at the time of study.

This step directly addresses Sub-Question 1: *"Which bio-inspired metaheuristic algorithm is best suited for the method being used?"*

By systematically reviewing the existing literature and selecting the most applicable metaheuristic within the scope of this research, this step ensures a well-founded basis for the algorithm's development.

2.1.2. Algorithm Configuration and Integration

Once a metaheuristic is selected, an algorithm must be developed that effectively integrates the MGA process with the chosen metaheuristic, ensuring its successful application to energy system spatial optimization.. This requires defining key algorithmic configurations, such as recombination operators, mutation strategies, selection mechanisms, and parameter tuning for both the heuristic and MGA. A development approach will be adopted, where different configurations are tested to determine their impact on the solution. Initially, a small energy system will created to test and evaluate the created algorithm. The refinement process will ensure that the algorithm produces a diverse set of solutions.

This methodology provides the necessary insights to answer Sub-Question 2: *"How can the tuning of algorithmic parameters optimize the performance of the selected heuristic for implementing a successful MGA optimization technique?*

2.1.3. Scalability, Evaluation and Benchmarking

Finally, the developed algorithm's effectiveness will be assessed by comparing its performance against an existing MGA approach. Key performance metrics will include:

- Computational efficiency: How long does the algorithm take to find near-optimal solutions?
- Solution diversity: How well does the algorithm explore the solution space?
- Accuracy and robustness: How do the generated solutions compare with deterministic MGA methods in terms of quality and feasibility?

A crucial aspect of this evaluation is determining whether the algorithm remains efficient and reliable as the scale of the energy system increases. Before the final results are created, iterative testing will be done to check if the algorithm operates as it should. This approach will assess whether the method can handle greater complexity without excessive computational burden or just failing to generate solutions in general.

This evaluation phase is essential for addressing Sub-Question 3: *"How will metrics be used to evaluate the algorithm's performance, and how do these comparisons with existing methods inform its applicability?"*

2.2. Systematic Literature Review Methodology

A systematic literature review is conducted to identify relevant research on the integration of MGA and bio-inspired heuristics in energy system optimization. The review begins by searching for studies that combine MGA with bio-inspired metaheuristics, assessing whether such approaches have been previously explored and identifying the types of metaheuristics applied. If no directly relevant studies are found, the search scope will be expanded to more general applications of MGA, as insights from other contexts may still inform its applicability with bio-inspired heuristics.

Once relevant literature is identified, the applied metaheuristics will be classified to determine which types have been used in similar optimization problems. This process will provide insight into existing research on MGA and metaheuristic combinations and the specific metaheuristics employed. Ultimately, this will help in selecting the most suitable algorithm for the thesis.

The literature review is conducted using Scopus and Google Scholar as primary databases. To ensure comprehensive coverage, predefined search strings incorporating multiple synonyms are used (see Appendix A). The search strategy first targets studies integrating MGA with bio-inspired heuristics in energy system optimization. If necessary, it expands to more general applications of MGA while maintaining a focus on its use with metaheuristics.

To refine the search results and exclude irrelevant papers, several measures are applied. "Modeling to Generate Alternatives" is explicitly searched instead of the acronym "MGA," which is commonly used in unrelated fields. Multi-objective optimization is included in the search criteria, as it often intersects with bio-inspired heuristics and MGA-related approaches, potentially offering relevant insights even if not the primary focus of this research. Results are also filtered based on relevance, citation count, and recency to ensure that only the most impactful studies are reviewed.

The screening process follows the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) framework. A PRISMA flow diagram will document the search process, providing a transparent overview of the number of initial search results, the filtering and exclusion process, and the final selection of relevant studies. This structured review ensures that the study builds upon existing research while addressing gaps in knowledge.

The results indicate that most relevant studies employed Genetic Algorithms when integrating a metaheuristic with an MGA approach. A detailed overview of these findings, along with a clear justification for using GA in this thesis, is provided in Section 3.1. Accordingly, the following methodology section will proceed under the assumption that GA is the chosen metaheuristic for this thesis.

2.3. Algorithm Configuration Methodology

Evolutionary Computing serves as the basis for various optimization algorithms such as the well known multi-objective approaches NSGA-II [33, 41]. These type of algorithms operate on the principle of iterative improvement: a population of candidate solutions is evaluated, and the best-performing individuals are selected to generate new solutions through crossover and mutation. This process continues until a stopping condition, such as convergence or iteration limits, is met. Figure 2.1 provides a visual representation of this evolutionary process.

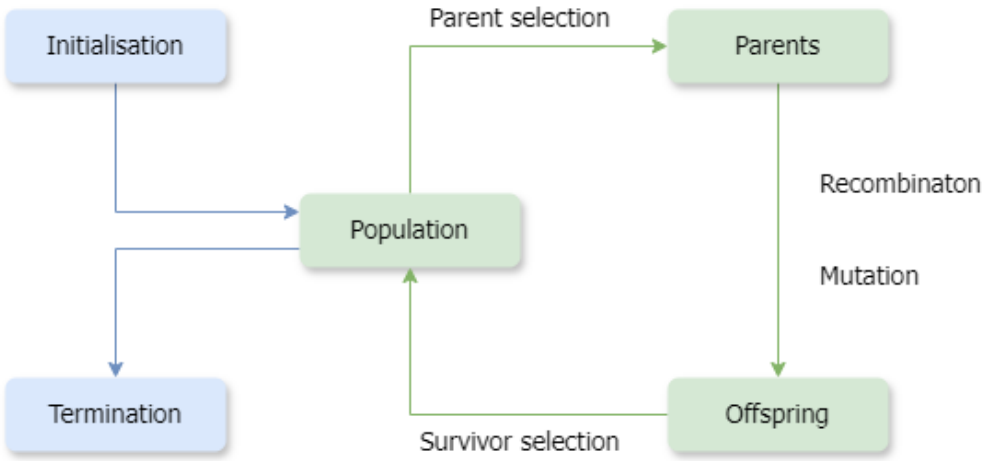


Figure 2.1: Flowchart of a General Genetic Algorithm.

GA is a specific type of evolutionary algorithm (EA) that encodes solutions as binary or non-binary strings and applies evolutionary principles to search for optimal solutions [42]. GAs typically rely on selection, crossover, and mutation, with modifications tailored to problem-specific requirements. For example, NSGA-II employs non-dominated sorting and crowding distance to maintain diversity, making it particularly effective for multi-objective optimization [43]. Designing an effective GA requires aligning its operators with the problem's objectives, a principle that applies when integrating MGA with a bio-inspired heuristic.

Previous studies, including Caicedo & Yun[10], Zechman[17], and Loughlin et al. [44], provide insights into combining MGA with evolutionary algorithms. Understanding these approaches is essential for adapting them to energy system spatial optimisation. The following section presents the mathematical foundation of MGA, followed by an analysis of Zechmans [17] MGA-GA combination. Finally, this thesis proposes a refined MGA-GA approach tailored for spatial energy system optimisation.

2.3.1. Mathematical Overview: Modeling to Generate Alternative

A brief overview of the MGA process is provided to facilitate a better understanding of the underlying mathematical framework. The modeled optimization problem in this case is defined as:

$$\min Z = f(x_{ij}) \tag{1}$$

Subject to:

$$g_k(x_{ij}) \leq b_k \quad \forall k = 1, \ldots, M \tag{2}$$

In function (1), $f(x_{ij})$ represents the objective function where (x_{ij}) are the system design variables across technology (i) and nodes (j). Function (2) includes all the constraints to which the modeled objective is subject, with M representing the number of constraints.

Suppose the optimal solution to equation (1) is denoted as x^0_{ij}, with the optimal value of the objective function being Z^0. The goal of the MGA process is to generate a second solution that is maximally different from the initial solution Z^0. Equation (3) expresses this in the most intuitive formulation.

$$\max D = \sum_{ij} |x_{ij} - x^0_{ij}| \tag{3}$$

The goal is to maximize the difference D, which represents difference between the optimal objective value (x^0_{ij}) and the new design expressed by ($xIij$). The maximum distance must be achieved while still being subject to the constraints of equation (2). But a new constraint gets introduced that also needs to be adhered to. This is the 'slack constrain' which makes sure the results remain in a determined relaxation of the optimal objective function value. The slack T thus to this relaxation. A slack of 10% means that solutions are allowed to deviate by up to 10% from the optimal objective function value.

$$f(X) \leq T(Z^0) \tag{4}$$

The procedure can be repeated if more alternative solutions are sought. If this is the case, equation (3) can be updated to become equation (5).

$$\max D = \sum_{a} \sum_{ij} |x_{ij} - x^a_{ij}| \tag{5}$$

Here (x^a_{ij}) represents the values in any a-th previously found alternative solutions, also including the optimal objective value. The generation of new alternatives stops when no new alternatives can be found, the difference between found alternatives is too small, or the desired amount of alternatives have been generated.

2.3.2. Designing the co-evolutionary MGA-GA for Energy System Optimization

The process described in equations 3, 4, 5, and 6 can be integrated into a genetic algorithm by iteratively solving for D, thereby generating multiple solutions that are maximally distinct within the solution space.

A method for combining MGA with GA is formulated by [17] where subpopulations, or niches, collectively search for different alternative solutions. This method can be further adjusted so that the GA-MGA combination can be applied to spatial explicit energy systems. The following method to create such an algorithm is as follows:

multiple subpopulation/niches, p are created. Each niche correspond to the an alternative solution being sought. The subpopulations consists of n individuals, where each individual represents the capacities of the technologies i at location j in the energy system.

Initially, the problem is solved in a standard way using Calliope to determine the optimal system cost. The costs of newly generated individuals are then restricted by the slack value T. If the cost of a solution exceeds this slack, the solution is deemed infeasible and assigned a fitness value of zero. Solutions that fall within the allowed cost range proceed to the next step.

Next the fitness of an individual is calculated. The fitness reflects the objective of generating maximally different design options. This is the minimal distance between the capacity value of an individual n in niche q and the centroid value c_{ij}^{p} of niches p, as seen in equation (7). Equation (6) represents the calculation of the centroid value. The centroid value is c is calculated for each technology i at location j in each niche p.

$$c_{ij}^{p} = 1/N \sum x_{ij}^{n,p} \tag{6}$$

$$D_{min}^{n,q} = min_{p}(min_{ij}|x_{ij}^{n,q} - c_{ij}^{p}| \quad \forall p \neq q) \tag{7}$$

The distance represents how close a variable of an individual is to the mean value of the same variable across other subpopulations. This distance will serve as the fitness of the individual, provided the individual was previously labeled as feasible. As explained before, if the individual was deemed infeasible it will keep its zero value for its fitness. An individual can represent multiple variables x_{ij}, such as when multiple technologies are deployed across multiple locations. In this case, the lowest distance value will be used as its fitness score, which is represented by min_{ij} in equation (7).

At this stage the genetic algorithm operators need to be implemented. First, all subpopulations undergo a selection operator. A predefined percentage of individuals are marked as elite solutions and carried over to the next step. The remaining individuals are selected through tournament selection: two random individuals are chosen from the subpopulation, their fitness values are compared, and the one with the higher fitness is selected to undergo the crossover and mutation process. If both the individuals have a fitness value of zero because they were both deemed infeasible, the one with the lowest system cost will be selected. Once selection is completed, the individuals will undergo the crossover operator. Two random individuals are chosen, and a percentage of their variables are swapped to produce new offspring. This percentage is adjustable and differs for each situation that is being optimized. Finally, mutation is applied: each variable in the individual is iterated through, and a small random change is applied to allow for mutation. Figure 2.2 shows a visual representation of the operators process. The values in the bar represent the capacity for a technology and f represents the fitness of the individual. After the new subpopulations are formed, the feasibility of all individuals modified by crossover or mutation needs to be recalculated, as their variable changes affect their associated costs, and thus their feasibility. Next, the centroids and fitness for all the individuals are recalculated, and the process repeats. The algorithm stops when the stopping criteria are met or when the desired number of generations has been reached.

The theoretical foundation of the GA works. But because it must be applied to energy systems in general, certain algorithm design options need to be implemented to ensure that the process does not crash. In any given generation, it is possible for a large number of infeasible solutions to emerge. This may occur due to mutation and crossover operators creating many infeasible solutions simultaneously, or as a result of the selection process, where multiple pairs of infeasible solutions are compared and subsequently selected. When this happens, the population may eventually consist entirely of infeasible solutions, resulting in fitness no longer being part of the selection process.

To address this issue, a limit is imposed on the number of infeasible solutions that can be selected within the population. If the population reaches a specified threshold of infeasible solutions, and two infeasible solutions are selected for a tournament, the algorithm will instead choose a random feasible solution from the population. Allowing some infeasible solutions to pass is intentional, as they may contain high-performing values within their genes that could contribute positively to the evolutionary process. However, the limit ensures that the subpopulation remains viable and avoids becoming dominated by infeasible solutions. Moreover, when an individual is labeled as infeasible, it has a chance to be replaced with an individual previously selected by the elitism operator. This mechanism prevents populations from starting with only infeasible individuals, a scenario that could hinder the population's ability to mutate and evolve toward an optimum solution effectively.

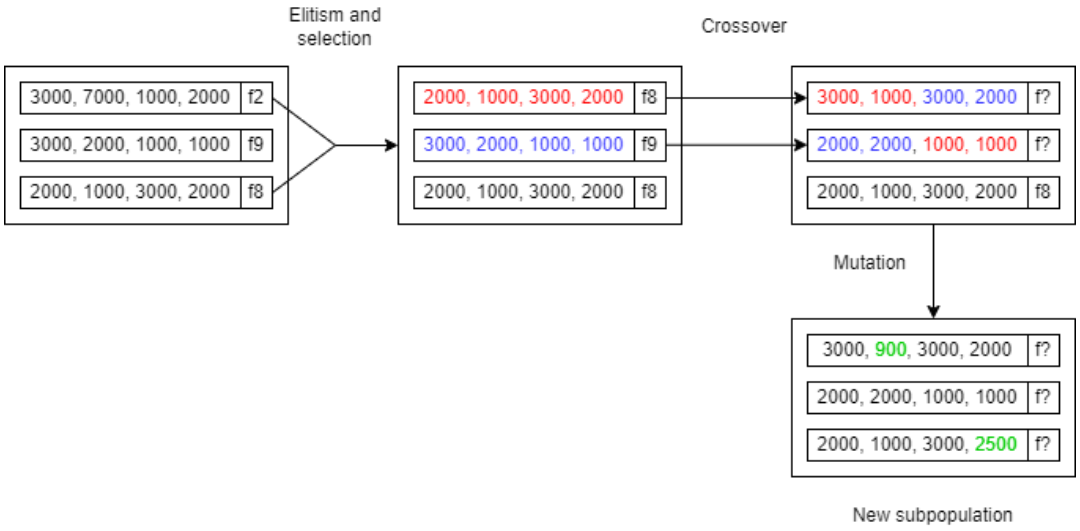


Figure 2.2: Energy system genetic algorithm operators

2.3.3. Optimization of Algorithmic Parameters and Model Efficiency

Test Model and Computational Considerations

The algorithm's initial structure has been developed but requires refinements to align with the energy system spatial optimization problem. A simplified test model is created to evaluate the GA performance before scaling it to a full energy system. The model consists of two nodes connected by a single transmission line, each equipped with demand, battery storage, and solar-PV technology. One node includes a combined heat and gas cycle, which, while costly, ensures demand is met. The test model validates the algorithm's ability to optimize spatial energy distribution while maintaining solution diversity.

Despite GA's efficiency in optimization, prior studies highlight its computational intensity, particularly when handling large populations and multiple attributes such as cost and fitness [10, 17]. To mitigate this, the following efficiency-enhancing techniques are implemented:

- Skipping redundant cost calculations for individuals unaffected by crossover or mutation.
- Optimizing solver settings within Calliope.
- Utilizing high-performance computing resources.

When recombination occurs and an individual remains unaffected, it does not need to be processed through the backend again; instead, it retains its previously associated costs. Gurobi is chosen as the solver due to its superior performance over CBC and CPLEX for large-scale mixed-integer programming problems [45]. To improve efficiency, the barrier method without crossover is used, reducing solving time by up to threefold [46]. While crossover can enhance solution quality, computational constraints favor a trade-off prioritizing speed, with small optimality and feasibility tolerances implemented to balance precision and efficiency. Given the computational intensity of multiple runs, DelftBlue, the TU Delft supercomputer, is utilized to significantly reduce solving time. DelftBlue's computing capabilities allow for rapid execution of optimization runs, making large-scale testing feasible [47].

Resolution

To manage computational demand, the model dynamically adjusts temporal resolution during optimization. Initially, the algorithm operates at a lower temporal resolution (e.g., monthly), increasing to daily or hourly as promising solutions emerge. An approach proven by Trondle et al. to be effective in reducing computational demand while also maintaining aspects of high resolution [48]. By intelligently adjusting the model's resolution, it is possible to reduce unnecessary solving time. Once the subpopulations converge, the resolution should then be increased so that the solution space identified at the lower resolution can be further refined and explored by the algorithm. Calliope provides two primary resolution adjustment methods:

- **Time Clustering**: Groups similar days to create representative clusters, reducing computational requirements while maintaining temporal diversity.
- **Time Masking**: Selects or excludes specific time periods, allowing focused analysis on critical energy system fluctuations (e.g., peak PV production).

The full-scale model operates at a six-hour resolution, as studies indicate that maintaining accuracy beyond this threshold provides diminishing returns [49]. The algorithm will initialize all resolutions at the beginning of the algorithm process, ensuring seamless transitions between temporal scales.

Population Initialization

At the beginning of the algorithm process populations with individuals need to be created. First the system will be solved to find the optimal solution. The corresponding technology capacities will be used to base the capacities of the individuals on. The initial individuals capacities are slightly adjusted to encourage exploration while ensuring feasibility. Next part explains exactly what is meant with the technology capacities from the optimal solution.

Capacity Initialization and Mutation

The model is solved to determine the optimal solution and extract the corresponding optimal capacities. Calliope categorizes technologies into several types: supply, demand, storage, transmission, and conversion technologies. However, demand and transmission technologies are excluded from the set of technologies that make up the individuals in the population. The capacity values included in each individual will be subject to mutation and crossover.

The algorithm restricts capacity mutation to predefined lower and upper bounds, ensuring that the mutation operator functions within an appropriate range. The mutation range is determined by the `energy_cap_min` and `energy_cap_max` values of the technology, which define the lower and upper bounds. These values can be extracted from the backend of the model. In some cases, `energy_cap_max` may be set to infinite for certain technologies. When this occurs, the value is replaced with the highest maximum capacity observed in another technology.

The DEAP mutation operator `MutPolynomialBounded` is used to mutate values within the specified bounds. The degree of variation between mutations is determined by the ETA value which is a variable build in the `MutPolynomialBounded` operator. The higher the ETA value, the smaller the difference between each mutation.

Exploration vs Exploitation in Genetic Algorithms

When designing a GA, a fundamental challenge is maintaining the right balance between exploration and exploitation, as these two mechanisms directly influence the algorithm's effectiveness. Exploration refers to the ability of the algorithm to search new areas within the solution space, preventing premature convergence to local optima. Exploitation, on the other hand, focuses on refining existing solutions by searching their immediate neighborhood, improving overall solution quality [50].

These two components are often in opposition. Excessive exploitation can lead to premature convergence, while excessive exploration may prevent convergence altogether [51]. The balance between these forces is primarily determined by the mutation rate, crossover rate, and selection pressure. Adjusting these parameters is essential, as even minor modifications can significantly impact performance. Since no universal parameter configuration guarantees optimal results, various parameter tuning techniques are required to find an effective balance [52].

Parameter Tuning vs Parameter Control

The design and performance of a GA are strongly influenced by parameter selection, requiring careful tuning to enhance efficiency. There are two fundamental approaches to parameter optimization, namely parameter tuning and parameter control. Parameter tuning means that the values are set before the execution of the algorithm and remain fixed throughout the run. While with parameter control the values are dynamically adjusted during the execution of the algorithm, adapting to problem-specific characteristics [53].

A well-tuned genetic algorithm performs significantly better than one with default or randomly assigned parameters [53]. While existing literature provides general parameter recommendations, fine-tuning is often problem-specific, requiring empirical testing. For complex optimization tasks, surrogate models, such as the designed test model, can assist in efficiently estimating optimal parameter values [54].

In this study, the following parameters are categorized under parameter tuning:

- Crossover rate
- Population size
- Solution amount
- Mutation rate

Additionally, mutation distribution index ETA (η) and resolution change fall under parameter control, as they dynamically adjust during the algorithm's execution [55]. Despite the conceptual difference between these two approaches, this study applies the same tuning method for all parameters for simplicity and efficiency.

Performance Metrics for Parameter Evaluation

Since GAs involve stochastic processes, evaluating their performance requires statistical validation. The fitness value of a GA is deterministic, as it is based on the optimization problem being solved, but the parameters and results are stochastic, requiring multiple runs to assess performance reliably [56]. To systematically measure GA performance, three key performance metrics are used [57]:

1. **MBF (Mean Best Fitness)** – Evaluates the highest fitness achieved across multiple generations.
2. **AES (Average Evaluations to Solution)** – Measures computational efficiency by assessing the number of function evaluations required.
3. **SR (Success Rate)** – Assesses the algorithm's reliability in consistently finding high-quality solutions.

Not all of these metrics are necessary for every study. Their relevance depends on the specific design goals of the algorithm. However, they provide a solid foundation for evaluating different parameter configurations.

F-Race Method for Parameter Optimization

To identify the optimal combination of parameter values, this study employs the F-race method, an iterative brute-force approach designed for tuning stochastic search algorithms [58, 59]. The F-race method evaluates multiple parameter combinations and ranks their performance, progressively eliminating the worst-performing configurations in each iteration. This process continues until either one optimal combination remains or there is no statistically significant difference among the remaining configurations. To assess statistical significance, the Friedman test is applied. This non-parametric test compares multiple parameter configurations across several independent runs, determining whether performance differences are statistically meaningful. The null hypothesis assumes no significant difference between parameter combinations. If the test rejects this hypothesis, pairwise comparisons are conducted using a post hoc test to determine which configurations perform significantly better. An indepth explanation of the F-race and corresponding mathematical formulations are given by Montero et al. [58].

Implementation of Parameter Tuning

To define the initial set of control parameters, this study focuses on tuning parameters independent of solving time, as these adjustments do not significantly impact computational runtime. Fitness evaluation will be based on Mean Best Fitness (MBF), summing the highest-performing individuals' fitness values across all generations.

Each experiment will be run five times, and the mean performance across these runs will determine the fitness score used for the Friedman test. While ETA, crossover rate, and mutation rate are varied, other parameters remain constant:

- **Number of generations**: 100
- **Subpopulation size**: 10
- **Number of subpopulations/niches**: 3

The resolution change schedule is also fixed. These variables and parameters are set at a constant to make sure they do not interfere with the results of the F-race method. Table 2.1 summarizes the initial control parameters selected for tuning:

Control parameter	Values
ETA	[(0.5, 2, 4), (1, 4, 7), (3, 8, 15)]
Probability of Crossover	[0.4, 0.5, 0.7]
Probability of Mutation	[0.05, 0.1, 0.2]

Table 2.1: Genetic algorithm control parameter sets.

A crossover rate of 1 means that each value of an individual has a 100% chance to get swapped with the value of a different individual. A mutation rate of 1 means that a single value of an individual has a 100% to mutate. While there is a lot of literature stating diverse parameter values, the values from 2.1 are based on what is generally considered average used rates. The crossover rate is normally half, which is why the ranges 0.4 - 0.7 are chosen. For mutation 0.05 is considered normal but sometimes it can be higher. Therefor a mutation rate of 0.2 is also incorporated to see how the algorithm behaves with a generally high mutation rate [60, 61].

Finalizing the Optimal Parameter Configuration

After completing the designated F-race iterations, or if no significant differences are found, the tuning process is finalized. If multiple parameter combinations yield equally strong results, the best configuration is selected based on highest fitness and speed of convergence, determined by the generation in which the solution was achieved.

The best-performing parameter set will then be used to assess the impact of resolution changes, population size, and subpopulation count. Other high-performing configurations with structural differences will also be analyzed for comparison. Alongside fitness evaluation, the solving time will be considered, as adjusting population size or resolution changes at different time steps can significantly affect computational efficiency.

No statistical methods will be applied when evaluating population size and subpopulation count. Instead, the time to solve and fitness results will be compared. The best parameter configuration from the statistical tuning process will serve as the baseline for these measurements. When analyzing population size, key factors include the generation in which the best solution is found, the solving time, and fitness quality. While increasing population size may enhance solution robustness, it also extends solving time. This trade-off will be evaluated to optimize the algorithm for larger energy system models.

Resolution change is the final parameter under investigation and represents a distinctive feature of this algorithm. Once other parameters are optimized, multiple runs will be conducted with different resolution transition points. The primary objective is to determine whether switching to a higher resolution affects population stability or causes divergence from previously converged values. Adjustments, such as accelerating the transition to the highest resolution or extending high-resolution runs, may yield further insights into algorithm performance.

Figure 2.3 provides a high-level summary of the GA process described in this section. It outlines the key steps, including initialization, selection, crossover, mutation, feasibility checks, and fitness calculations. Specific implementation details, such as the fitness function calculation or customized tournament operator, have been omitted for simplicity. The next section will expand on this framework, ensuring the algorithm is effectively adapted for large-scale energy system optimization.

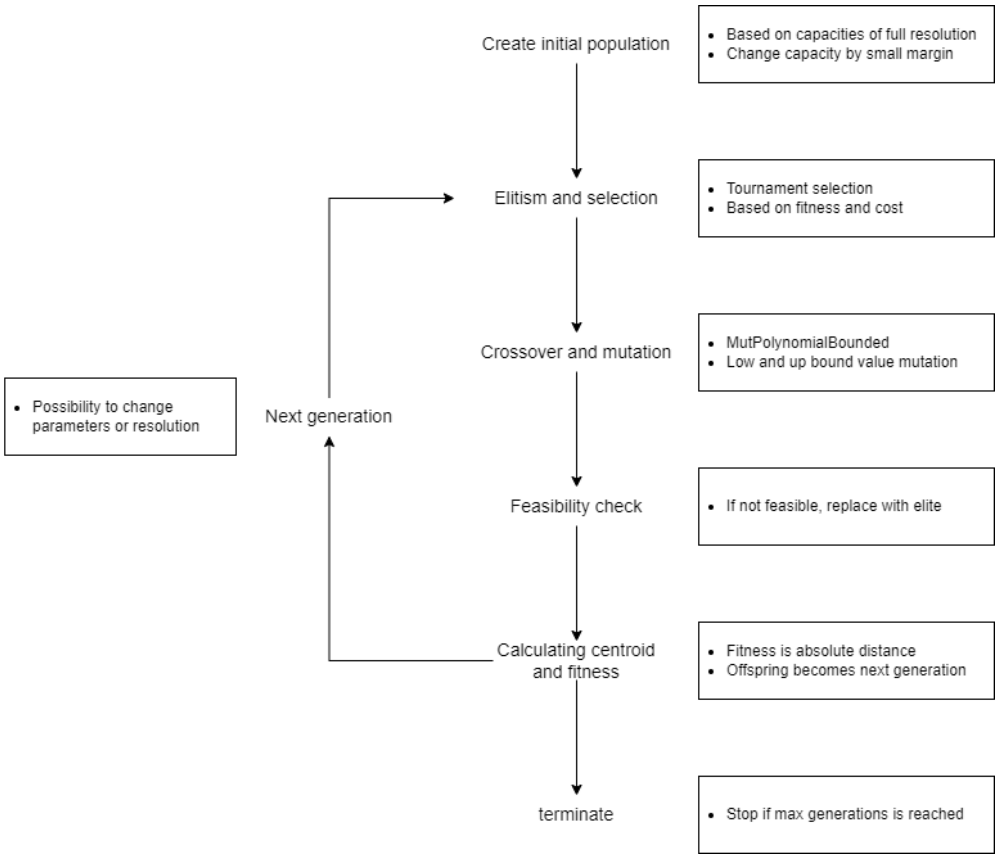


Figure 2.3: Test Model GA Flowchart.

2.4. Scalability, Evaluation and Benchmarking Methodology

The behavior and interaction of parameters have been clarified through their application to a simple energy system. The next step is to determine whether the designed algorithm can produce insightful results when scaled up and applied to larger, more detailed models. The findings from the previous section provide a foundation for adjusting the algorithm to fit a larger model. However, scaling up introduces new questions and challenges that must be addressed to ensure successful integration.

This section focuses on the adjustments required for the algorithm to function effectively with a large-scale model. The selected model is the one presented by Lombardi et al. [12], with its code available in the associated Zenodo repository [62]. This model serves as an ideal test case due to its complexity and relevance to real-world energy system problems. Its structure captures diverse technologies, spatial variations, and dynamic constraints, making it a robust environment for testing scalability. Moreover, as the model is built in Calliope, it remains fully compatible with the genetic algorithm developed in this study.

The algorithm has been intentionally designed without hard-coded structures, allowing it to be easily applied to other models built in Calliope. This flexibility ensures seamless integration with the model by [12]. With the test model there were just a few technologies present in the system. The European model on the other hand implements over a thousand technologies dispersed over many locations. As a results, the number of technologies involved in mutation and crossover operations is significantly higher, raising the question of whether the algorithm will still perform as intended with this increased number of technologies, locations, and links. For instance, the mutation process may face increased combinatorial complexity as the number of technologies grows, while crossover operations might encounter structural mismatches when dealing with diverse locations and link configurations. Additionally, the computational load increases substantially due to the system's greater complexity and higher memory requirements for operation. The following sections will address these challenges by examining adjustments in data management, computational optimization, algorithmic efficiency, and evaluation strategies.

2.4.1. Optimizing Computational Resources

To evaluate the small model algorithm, the DelftBlue cluster was utilized to perform multiple runs in parallel. Leveraging its substantial computational resources, a large dataset was generated to analyze the algorithm's behavior. As the focus shifts toward modeling the European energy system, the necessity of DelftBlue has become even more evident. While running numerous simulations is no longer the primary requirement, the supercomputer's extensive memory capacity remains crucial.

To configure the execution of the Python script, a SLURM job submission script is used to define the required computational settings. The exact SLURM file format is provided in Appendix F. Due to the model's high memory demands, the high-memory partition is employed, as it is specifically designed for CPU-intensive jobs requiring substantial RAM [47]. To ensure optimal performance, several test runs will be conducted to determine the most suitable configuration for the model.

2.4.2. Data Management Adjustments

The algorithm currently retrieves all technologies and their corresponding capacities from the backend, excluding transmission and demand technologies, and stores them in a structured table. In the mock-up model, which included only five technologies, such data did not require intricate structuring. However, the larger model, featuring over a thousand technologies distributed across multiple locations, demands a more thoughtful and efficient approach to data organization. Proper structuring is essential not only for creating clear tables and insightful visualizations but also for enabling a deeper understanding of how the algorithm performs when applied to complex large-scale models.

Instead of creating visual figures to represent the data, the algorithm's output will be saved in an Excel file. This approach makes it easier to modify and analyze the data, providing greater flexibility when emphasizing different aspects of the results. To ensure the data can be utilized in various ways, it is important to incorporate the core aspects of the algorithm. In this case, it is of importance to know the capacities of the technologies the total system cost and the assigned fitness when storing generated individuals in external files.

An extra feature of the algorithm is to identify combination of individuals that form the best performing combination of solutions. The best individual from the complete run is identified by examining the highest cumulative fitness of the best-performing individual from each population across all generations. However, it is equally valu-

able to gain insight into how the algorithm behaves throughout the process. To facilitate this, the Excel file will include snapshots of the populations at specific generation intervals. This approach allows for a detailed analysis of population evolution over time, by looking at how the individuals in the populations evolve over time.

The old algorithm worked by storing the best-performing technology combination in an allocated variable. Each time a new best-performing combination of individuals was identified, the algorithm would overwrite the previously stored data in the variable. For the larger model, a new feature was added. Instead of overwriting the previous data, each new best-performing combination is now stored in a separate sheet within the Excel file. This approach provides a more structured overview of how the best-performing capacity combinations behaved throughout the algorithm's run.

2.4.3. Algorithm Structural Adjustments

The fitness of an individual is determined by calculating the distance between the individual's capacity values and the centroid values of all other subpopulations for capacity i. Specifically:

1. For each capacity value i in individual k of subpopulation q, compute the difference between the capacity value and the centroid values of subpopulation p capacity value i, where $p \neq q$.
2. Repeat this calculation for all capacity values of the individual.
3. The lowest value becomes the fitness of the individual.

The issue with the large model is that many technologies have both a maximum and minimum capacity value of zero. Currently, this means that these technologies will be included in the individual but cannot undergo mutation. Consequently, the centroids for these capacity values and their distances from the centroid will always be zero. As a result, the fitness of the individual is determined under this assumption, which causes a problem in the algorithm, as the fitness will always be zero and cannot evolve away from this value. Technologies with both lower and upper capacity bounds of zero will therefore be eliminated from the algorithmic process. Since their capacities are always zero, their presence in the algorithm is redundant. Removing them ensures the algorithm functions as intended.

Due to the way population creation and randomization work in Python and DEAP, multiple duplicates can arise in the created subpopulations. This can result in values that differ by zero, causing the fitness of an individual to be zero from the start. To solve this, a check is implemented to detect identical values in the initial subpopulation. If duplicates are found, a small random adjustment is applied to the values. Making sure there are only non-identical individuals in the initial populations. As a result, this ensures that the all initial individuals start off with a fitness value, albeit a low one.

In the large model, when seeking the optimal solution, some technologies may be assigned extremely low capacities as part of the final solution. For instance, capacities as low as 1×10^{-6} (1 MW) may appear. Compared to the scale of the model, where some technologies represent tens or hundreds of GW, such small values are almost negligible. These small capacities can slow down the model, as they require the solver to process minor adjustments that have little to no impact on the overall system. To address this, any technology capacity below 1×10^{-5} will be automatically set to 0 during initialization. This adjustment reduces computational overhead, improves solving efficiency, and streamlines the optimization process.

In the test model algorithm, a custom tournament function and a standard elite function were introduced. The goal of these operators is to ensure that better-performing individuals were more likely to advance to the next generation. When individuals were mutated and selected, and their fitness is deemed infeasible, a small fraction would be replaced with previous unaltered elite individuals. This mechanism prevented scenarios where an entire generation had zero feasible individuals, ensuring that some feasible individuals were always present to continue the algorithm's process. Infeasible individuals were retained with the hope that their genetic information could eventually mutate into a feasible solution. This approach ensured that no genetic data is lost and that the algorithm would not crash due to a lack of feasible individuals.

However, given the understanding of how the algorithm operates and the structure of the large model, it is anticipated that maintaining the same settings will lead to problems when applied to the larger model. Due to the high dimensionality of the individuals' values, the likelihood of an infeasible individual being created is extremely high. In addition, the probability of this individual mutating back to a feasible state is minimal due to the vast number of variables. Keeping such infeasible individuals in the population disrupts the algorithm's optimization process.

To address this issue, a new mechanism is introduced. Before the mutation and crossover steps, a list of all individuals is created. After crossover and mutation, if an individual is found to be infeasible, it is replaced with a random individual from this pre-generated list. If the individual remained feasible, it continued as normal. This updated process ensures that the algorithm does not carry infeasible solutions forward, allowing it to operate exclusively with feasible individuals and thereby maintaining the algorithm's effectiveness and stability. While stabilizing the algorithm it can still be argued that such a mechanism eliminates potentially valuable genetic material and maintaining more homogeneous solutions in the populations. such structures could cause the algorithm to prematurely converge to an optimal solution.

2.4.4. Balancing Resolution and Solve Time in Large-Scale Models

Due to the large size of the European model, it is expected that the time to solve will take a while. The results of the European model have a resolution of 3 hours, so the aim is to do produce final results which also have a resolution of 3 hours. The resolution change that was proposed in the previous method help reduce the time needed to solve all the individuals of each generation. The approach on how to structure these resolution changes can differ a lot. The following methods will be tested:

- Initialize a low resolution and end at the desired resolution
- Apply the masking technique as initial low resolution
- Apply the clustering technique as initial low resolution

Test runs are needed to determine the time required for each approach. By measuring the time taken to solve an individual in each scenario, it will be possible to calculate the total runtime using the initialized resolution method.

2.4.5. Dynamic Slack Adjustment for Resolution Transitions

One of the primary concerns in the algorithm's design is ensuring stability when switching between resolutions. A key challenge arises from the fact that costs are calculated differently at each resolution level. A low resolution generally produces lower cost, compared to solutions generated at a higher cost. The reason for this can lie with how costs are calculated when different time steps are involved. For example, a low resolution maybe does not take the discharge of batteries into account as well with higher resolutions. This variation can cause the algorithm to crash or become non-functional when transitioning from lower to higher resolutions. To mitigate this issue, a dynamic slack adjustment approach is implemented.

The core idea behind this approach is to gradually adjust the slack value as the resolution increases. Instead of maintaining a fixed slack threshold throughout the optimization process, the slack value starts at a lower percentage (e.g., 5%) and is progressively increased with each resolution transition. When the final resolution is introduced, the desired slack value is also initialized. This method ensures that individuals in the population can gradually evolve towards the final cost constraint rather than being abruptly classified as infeasible.

This adaptive slack approach helps to ensure that individuals remain cost-feasible across different resolution levels, preventing the algorithm from being pushed towards unrealistic or impractical solutions. Moreover, it reduces the likelihood of algorithm crashes, as the gradual tightening of constraints allows solutions to naturally adapt rather than being suddenly deemed infeasible.

To validate this approach, multiple test runs will be conducted. While the model's size and computational demands limit the number of tests, these runs will provide crucial insights into whether gradual slack adjustments effectively prevent crashes, ensure solution feasibility, and maintain computational efficiency. By integrating this strategy, the algorithm can transition smoothly across resolutions, maintaining stability while optimizing energy system configurations.

2.4.6. Structuring Data for an Effective GA Benchmarking Analysis

The algorithm that was applied to the small model iteratively stored generations in an Excel file. Whenever a new highest-performing individual combination was identified, the results were saved in a separate sheet. Each stored individual includes the following data: the generation in which it was created, the subpopulation it belongs to, its index within the subpopulation, its associated cost, its fitness, and the capacity values for each technology.

Lombardi et al. [12] employed the SPORES method, generating 210 SPORES solutions for four different weighting methods, resulting in a total of 840 solutions. All results and corresponding data are publicly available in the Zenodo repository [62]. These results are stored as CSV files, providing structured access to all 840 solutions and their respective capacities. The first 50 runs of each weighting method, referred to as the ”main batch,” focus on identifying spatially distinct energy deployment configurations by adjusting weights assigned to high- and low-performing technologies. The remaining 160 runs (the ”parallel batch”) involve systematic exploration, including minimizing or maximizing specific technologies such as wind, solar, or nuclear energy.

For benchmarking, only the first 50 runs will be used, as these results best align with the search objectives of the developed GA. The GA's performance will be compared against each of these 50 runs across the four weighting methods: integer, relative deployment, random, and evolving average. A detailed explanation of these weighting methods is provided in [12].

SPORES results are visualized in a three-dimensional scatter plot, where capacities are normalized and expressed in terms of transmission capacity utilization, renewable capacity utilization, and electricity storage capacity utilization. The normalization is based on the maximum deployment potential of each technology, which corresponds to the maximum allowed capacity. In Calliope, this is represented as `max_cap_value`.

Since the developed GA does not adjust transmission capacities, a direct benchmark against the exact technologies from the SPORES model is not feasible. However, given the availability of raw SPORES data, the GA can still be bench-marked against renewable and storage technologies. Renewable capacity will be further categorized into solar and wind technologies, as they form the majority of renewable capacity deployment. A three-dimensional comparison will be made using the capacities of solar PV, wind, and battery technologies. While transmission is omitted, this structure still enables a comprehensive comparative analysis.

Instead of normalizing data using `max_cap_value`, capacities from the optimal solution in the existing CSV files will serve as a reference. The PV capacity consists of open-field PV (`open_field_pv`) and roof-mounted PV (`roof_mounted_pv`). Wind capacity is divided into offshore wind (`wind_offshore`), competing onshore wind (`wind_onshore_competing`), and monopoly onshore wind (`wind_onshore_monopoly`). Finally, storage capacity includes both battery storage (`battery`) and hydrogen electricity storage (`hydrogen_storage`). The corresponding capacity values are:

- PV capacity = 4,774.1 GW
- Wind capacity = 7,317.7 GW
- Storage capacity = 703.8 GW

Beyond benchmarking against the large model, the algorithm's evolution over generations will also be visualized. Every ten generations, the total battery, PV, and wind capacities of all individuals in the population will be plotted, tracking their progression until the final generation. This will provide insights into how the GA converges over time, highlighting trends that might not be apparent in the final results alone.

First, all 50 initial spore runs for each method (integer, relative deployment, random, and evolving average) are compared alongside the results of the GA niches. Due to the limited number of results generated by the GA, an equal number of results will be selected from the SPORES outcomes to ensure a fair comparison. Doing so creates consistency in the sample size. If not done, the large sample size could skew the comparison. For this reason, a second plot is created that just takes the first three SPORES answers for each method. The SPORES is an MGA method which identifies answers iteratively. The first three answers should be answers that already thoroughly explored the solution space. Using just three SPORES solutions should still create a reasonable comparison.

The plots are generated using Python. The x-axis represents the PV energy deployment as a percentage, meaning the values on this axis can be multiplied by the total PV energy deployment in the optimal solution. The y-axis represents wind energy deployment. The color axis indicates the deployment of electricity storage technologies, with dark blue representing lower deployment and lighter yellow indicating higher deployment. Both figures are presented on the next page.

3

Results

The results chapter presents the findings from the literature review, algorithm configuration, and algorithm scaling, benchmarking, and evaluation. A discussion of these results will be provided in Chapter 4.

3.1. Results Literature Review

First, the literature on energy system optimization, MGA, and bio-inspired heuristics was searched and reviewed. Figure 3.1 shows that fifty-three relevant papers were identified. An initial screening excluded literature that did not involve energy system optimization and bio-inspired heuristics. Subsequently, research not focused on the spatial optimization of energy systems using bio-inspired heuristics were further excluded. Only one paper, by Prina et al. [13], addressed spatial optimization. It uses a multi-objective optimization approach combined with a bio-inspired heuristic, making it the closest resemblance to the proposed research in this thesis when considering energy system optimization. Some papers focused on regional optimization for example of a country [63] but did not include spatial deployment of technologies, while other papers looked at spatial optimization of just a certain technology [64].

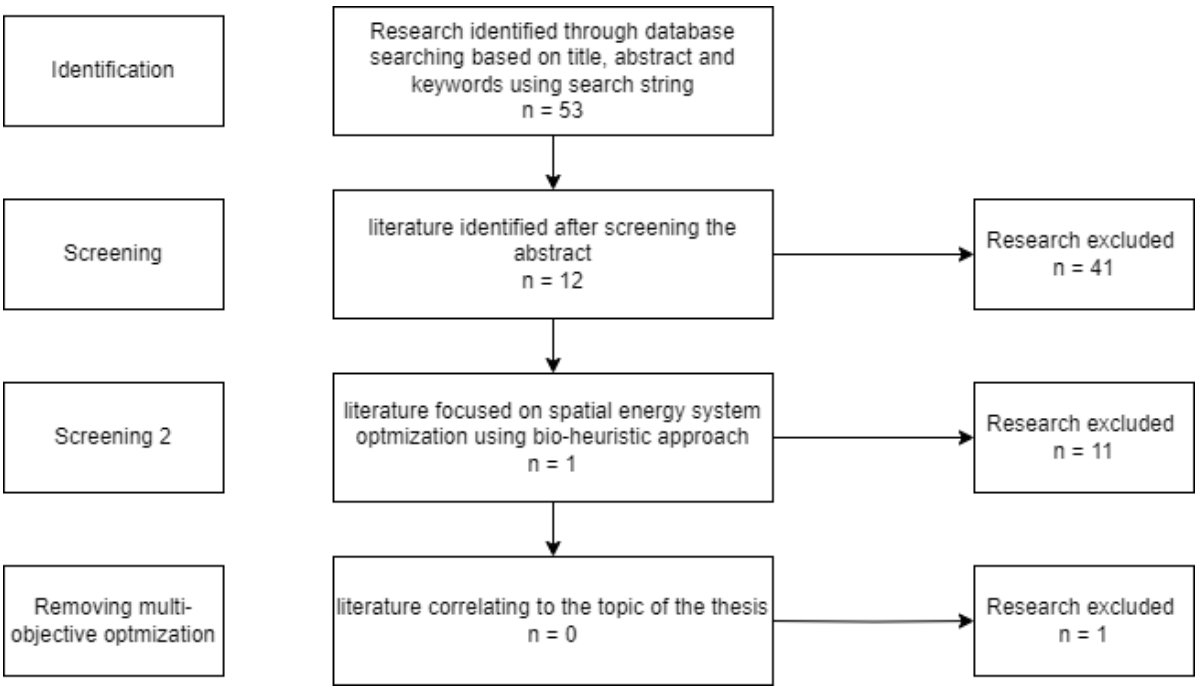


Figure 3.1: Systematic literature review focused on MGA, bio-inspired heuristics and energy system optimization

Figure 3.1 and the reviewed literature highlight a gap in knowledge within the specific field of this thesis. While MGA is widely applied to energy optimization problems, the method that incorporates a bio-inspired heuristic is relatively novel [40].

Due to the Limited research literature, the search expanded its scope to include studies discussing MGA and bio-inspired heuristics, applied to optimization problems in general. The literature review was conducted using the second search string provided in appendix A. This search string was applied to both Scopus and Google Scholar to identify relevant studies. Initially, the Scopus search yielded a structured overview of several papers that combined MGA with bio-inspired heuristics. Applying the same search string in Google Scholar resulted in additional sources; however, after screening the Google Scholar results, only two papers were considered unique compared to the Scopus findings. In total, twenty sources were identified that integrate MGA and bio-inspired heuristics, all applied within the context of an optimization problems (figure 3.2).

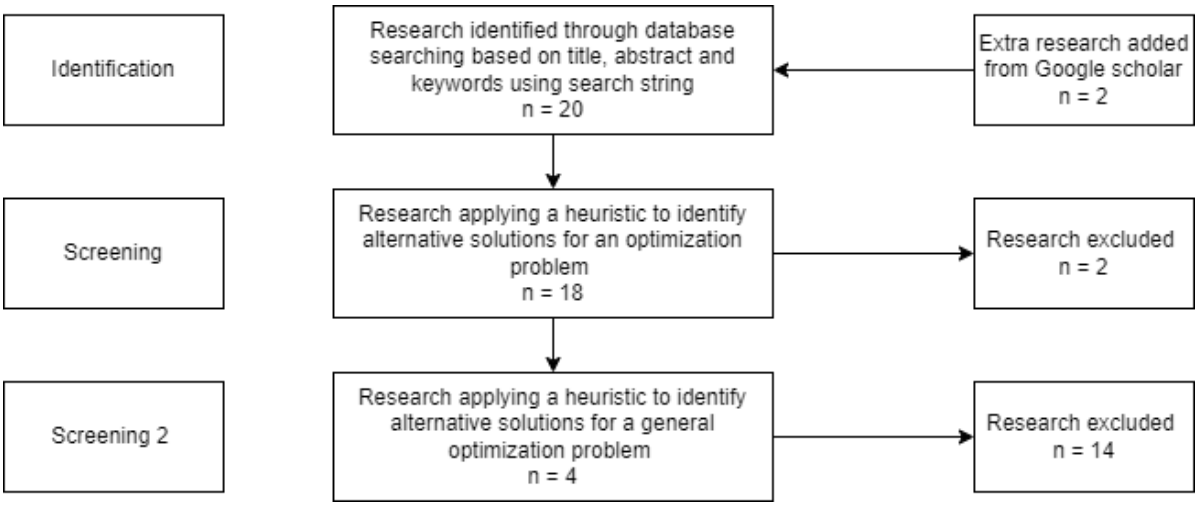


Figure 3.2: Systematic literature review focused on MGA, bio-inspired heuristics and a general optimization problem

The first screening excluded papers that did not apply a bio-heuristic to find alternative solutions, meaning that these papers only discussed or reviewed the concept without applying it to solve an optimization problem. While useful, this part of the literature review tries to identify examples on which the further algorithm design can be based on. Meaning example of implementation is of interest. The second screening excluded all scientific papers that applied both concepts to specific optimization problems. With specific it is meant that the literature solved an exact problem. Meaning the theory used was not useful based on general optimization problem theorem. Although this literature may hold value for the thesis, as the methods could potentially be adapted to the thesis case, the final selection focused solely on research that applied both concepts to general optimization problems. Ultimately, four papers were identified that met the established criteria (table 3.1).

There was minimal diversity in application, with most studies focusing on specific problems such as engineering [65] [66], waste management [7] [67], and microgrid optimization [68]. However, beyond the MGA-heuristic combinations applied to specific fields of research, the final selected papers concentrated on more general applications of the MGA and bio-inspired heuristic concepts. These four papers primarily explain the mathematical processes and evaluate the performance of the MGA-heuristic combination. Table 3.1 presents the articles that provide a broader view of the application of MGA combined with bio-inspired heuristics for general optimization problems. The table also specifies the type of bio-inspired heuristic used in each study.

Author	Year	Article Title
Imanirad et al.	2013	A concurrent modelling to generate alternatives approach using the firefly algorithm
Caicedo & GunJin Yun	2011	A novel evolutionary algorithm for identifying multiple alternative solutions in model updating
Zechman & Ranjithan	2004	An evolutionary algorithm to generate alternatives (EAGA) for engineering optimization problems
Loughlin et al.	2001	Genetic algorithm approaches for addressing unmodeled objectives in optimization problems

Table 3.1: Summary of research on bio-inspired heuristics used in Multi-Objective Genetic Algorithm (MGA) approaches.

The literature review shows that there is currently no research incorporating an MGA approach combined with bio-inspired heuristics focused on the optimization of an energy system. Prina et al. [13] closely resembles what this thesis aims to achieve but does not incorporate an MGA approach. When searching for an MGA-heuristic combination applied to general optimization table 3.1 showed all the relevant literature found. Noticeable is that three methods used an EC approach with just one using an SI method [69].

The literature in table 3.1 will provide foundational knowledge for the algorithm that needs to be developed in this research. To ensure comprehensive coverage of relevant literature, a snowball method will be implemented to identify any significant research that has used one of the studies in table 3.1 as its basis. To achieve this, the identified literature will be searched in Google Scholar, where citations of the research can be reviewed. Google Scholar is used for this step because it includes a broader range of sources. For instance, the paper by Imanrad et al. [69] shows three citations in Scopus, whereas Google Scholar lists seventeen. The method developed by Zechman [17] serves as the foundational basis for this body of research, with some studies representing further work by Zechman themselves [70, 71, 72], while others apply Zechman's concept to new approaches [73, 74]. These articles are particularly valuable when designing the algorithm, as they each provide detailed explanations of how the algorithm functions.

Additional research has been identified that applies genetic algorithms to urban planning [75] and transportation networks [76]. Both studies demonstrate the necessity of tailoring genetic algorithms to fit specific applications. This literature provides valuable examples of how such adaptations are implemented, offering insights into the process that can inform the integration of these techniques in the context of this thesis.

3.2. Results Algorithm Configuration

3.2.1. Parameter Tuning

The results derived from the algorithm configuration method are primarily related to parameter tuning. For the initial stage of parameter tuning, the F-race method was employed to systematically evaluate and optimize the parameters. Using the different parameter values for mutation, crossover, and ETA, a total of twenty-seven combinations were generated. Each of these twenty-seven combinations were run five times, and the fitness and the generation at which the best solution was found were recorded. All the results can be found in appendix B.

The fitness metric was then applied in the F-race test, which identified significant differences among nineteen of the twenty-seven parameter combinations, while the remaining eight showed no significant difference. These eight combinations were allowed to go to the next step of the the F-race method. This process repeated until no significant differences were found between the parameter combinations.

In the final iteration of the F-race test, no significant differences were found among seven remaining combinations, marking the conclusion of the method. Table 3.2 presents the remaining combinations along with their corresponding parameters.

Combination	ETA	Mutation Rate	Crossover Rate
6	(0.5, 2, 4)	0.2	0.5
12	(1, 4, 7)	0.2	0.4
15	(1, 4, 7)	0.2	0.5
18	(1, 4, 7)	0.2	0.7
21	(3, 8, 15)	0.2	0.4
24	(3, 8, 15)	0.2	0.5
27	(3, 8, 15)	0.2	0.7

Table 3.2: Remaining combinations after F-race and their specific parameters

There are still many combinations remaining that can be used as parameter settings for the algorithm. In addition to fitness, the generation in which the best solution was found was also recorded, allowing for the calculation of the average fitness and average generation for each combination. The F-race method is useful for comparing a single metric, such as fitness, across combinations. It helps identify and exclude significantly worse-performing combinations from the results. However, this method does not support the simultaneous comparison of multiple metrics, such as fitness and generation.

After excluding combinations using the F-race method, the remaining combinations can be analyzed further by comparing their performance on the generation metric. By standardizing both metrics (fitness and generation) and plotting them against each other, it becomes possible to evaluate the trade-offs between these metrics for each combination. This comparison provides insights into how the remaining combinations perform relative to one another. The resulting plot is shown in Figure 3.3.

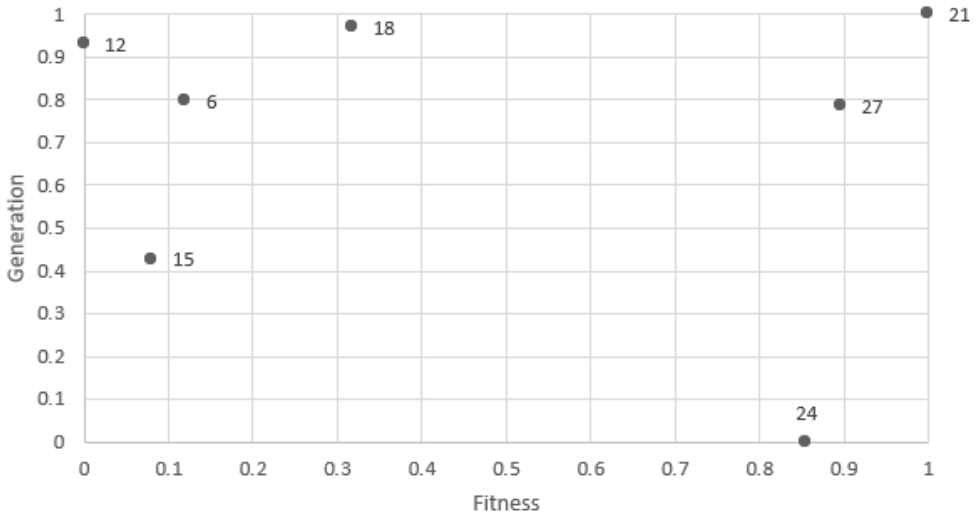


Figure 3.3: F-race final results generation and fitness plotted

Figure 3.3 shows which combinations perform overall the best. Combinations that have a lower value and a higher fitness value are deemed better performing. This is because a high fitness value is preferred, while a lower generation value, which indicates an earlier discovery of the optimal solution, is also desirable. The top-performing combinations based on fitness are 21, 24, and 27. The key attributes these combinations share are their mutation rate and ETA value. Additionally, for all combinations, the optimal mutation rate is 2%. While combinations 21 and 27 achieve high fitness values, they do not perform as well as combination 24 on the generation metric. Combination 24 not only performed the best on the generation metric but also achieved strong performance on the fitness metric. For this reason, parameter combination 24 will be used as the base parameter combination in future testing.

The next step involves testing the parameters that are expected to have a significant impact on the time required to solve the problem. For this analysis, the F-race method will not be applied. Instead, a new metric will be introduced: the time taken to complete a single run of the algorithm.

The first parameter tested, with time measured, is the number of niches that can be initialized in the algorithm. For this parameter, it is important to understand the effect of increasing the number of niches on both the fitness and the time required to complete a run of the algorithm. All results are detailed in Appendix C.The number of niches tested ranged from three to five, with each configuration run multiple times. The size of each niche was fixed at ten, while the mutation, crossover, and ETA parameters were set according to the values from combination twenty-four.

3.2.2. Number of Niches

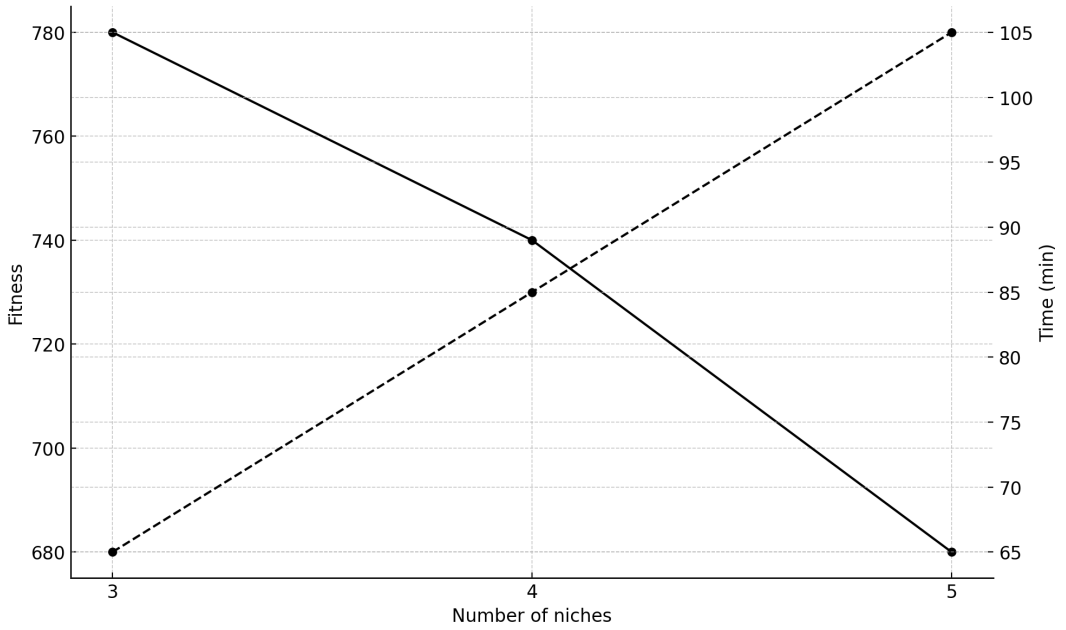


Figure 3.4: Effect of increasing niches on time and fitness

Figure 3.4 shows the correlation between the fitness and time to solve when increasing the fitness. From three niches to five niches there is an decrease in overall fitness of 13% while the time to solve the run increases with 60%. Full percentage comparison of the difference between the number of niches can be found in appendix C. The results indicate that increasing the number of niches causes a slight decrease in the overall quality of the algorithm's performance, while the time required to solve increases significantly. It is important to note that these differences specifically apply to the model used for testing. However, the findings can still be considered when applying the algorithm to larger models. No "best-performing value" can be determined from the tests conducted, as the number of solutions sought depends entirely on the preferences and requirements of the specific situation that the algorithm is being used for. Understanding the trade-off between increased solve time and the quality of fitness solutions is valuable for determining whether the additional time required is justified by the potential for generating more solutions.

3.2.3. Size of the Niches

To fully understand how the algorithm behaves under different circumstances, it is important to consider changes in the size of the population. Figure 3.5 presents three distinct plots (fitness, generation, and time). These plots illustrate the size of the niches on the x-axis, and the different numbers of individuals tested for varying niches sizes, which are represented as p3 and p4. The fitness plot clearly demonstrates a linear improvement in fitness quality as the number of individuals in a population increases. This trend is also observed when increasing the number of niches; however, the overall fitness quality for p4 is lower compared p3. Increasing the number of niches results in a more crowded solution space, with multiple niches competing to find diverse optima. A more crowded solution space reduces the overall fitness value of all niches, as it becomes more challenging to identify an optimum that is sufficiently distinct from other solutions. This crowding effect limits the ability of the algorithm to refine solutions effectively, leading to a decrease in overall fitness.

The generation in which the optimal solution is found is shown in Figure 3.5. Unlike fitness, which demonstrated a linear increase with the number of individuals, no such trend is observed for the generation in which the solution is found. For both p3 and p4, there is a clear performance improvement when the population size reaches twenty-five. However, a consistent increase in performance with each iterative increase in population size cannot be concluded from the plot. Therefore, it can be stated that increasing the population size helps speed up the process of finding the best individual, but it does not guarantee improvement with every incremental increase.

The linear increase in solving time with both the size of the population and the amount of niches is clearly visible. Using this plot and the associated data, it is possible to predict the solving time of the algorithm, even for different models. This allows for solving a model with a small amount of niches and a small population size, and then predicting the solving time when these parameters are increased. Such predictions can help determine the computational requirements needed for different scenarios. All the data for the runs that were completed can be found in D.

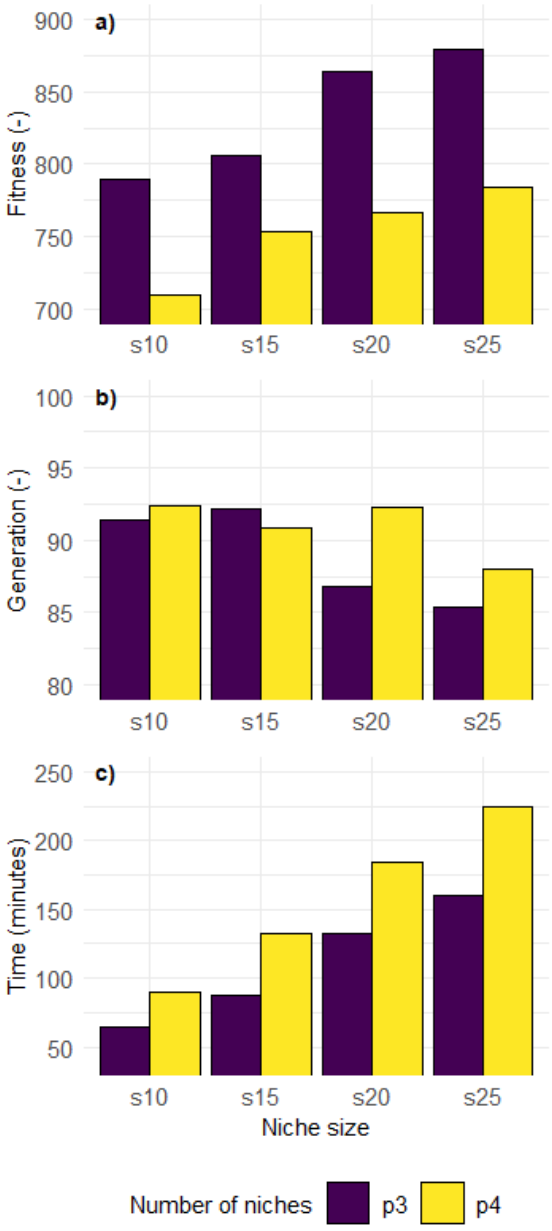


Figure 3.5: Effect of increasing niche size and number of niches on fitness (a), best generation (b) and time to solve (c).

3.2.4. Resolution Change

Multiple runs were conducted to observe the behavior of the resolution when it was switched. To do so the total generations that were doing were increased to hundred-fifty and two-hundred. The sum of the highest fitness values in each generation was plotted, as shown in Appendix E. For these runs, the following parameters were used: the amount of niches was set to three, the population size was set to ten, and parameter combination twenty-four was chosen. The switch to high resolution happened at generation ninety-five. An example of such a plot is presented in Figure 3.6.

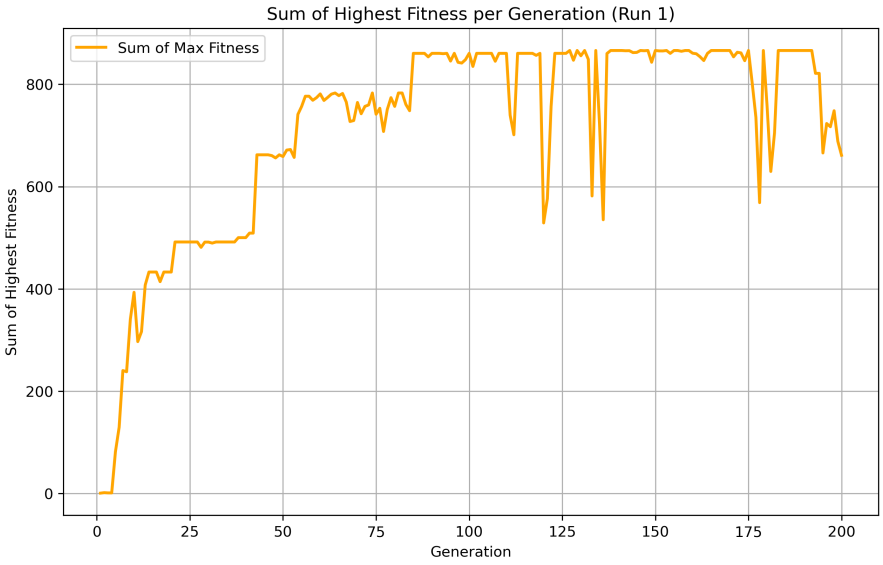


Figure 3.6: plot of sum highest fitness for each generation

Each plot exhibits a similar characteristic: at a certain point, the fitness reaches a barrier where it appears that the maximum sum fitness has been achieved. Even when switching to a higher resolution, the sum of the fitness does not increase, indicating that the optimal solution has already been found before the resolution change. It is important to note that while the plot can decrease in sum fitness when switching to the highest resolution, it never exceeds the found fitness optimum by a large amount. If it does and it finds the best fitness after the highest resolution has been initialized, the new found optimum increased by a very small margin. For example in figure 3.6 the best solution was found in generation hundred-twenty-eight. which is hard to conclude from the figure itself. But eventually it is of interest how the capacity distribution behaves in such a situation.

We will therefor look at another run which plot can be found in appendix E, figure E.1b. In this case the highest performing individuals right before the resolution change were taken, and compared to the eventual highest performing individuals which were found after the highest resolution. Table 3.3 shows individuals from before (generation 95) and table 3.4 shows the individuals after (generation 132) the highest resolution change. The tables shows the capacity and the fitness of the individuals. Table 3.5 shows the percentage difference between the capacities before and after the resolution change.

Population	Solar PV 1	Solar PV 2	CCGT	Battery 1	Battery 2	Fit (Before)
Niche 1	2325	12869	5000	6175	3419	305
Niche 2	7165	12430	4710	3768	7373	237
Niche 3	10850	14977	4413	9301	1921	280

Table 3.3: Capacity and fitness values before resolution change. Appendix E, figure E.1b

Population	Solar PV 1	Solar PV 2	CCGT	Battery 1	Battery 2	Fit (After)
Niche 1	2938	12339	5000	5720	4980	299
Niche 2	9155	11796	4697	4542	7166	284
Niche 3	9880	14988	4413	9709	2007	287

Table 3.4: Capacity and fitness values after resolution change. Appendix E, figure E.1b

Technology	Niche 1 (%)	Niche 2 (%)	Niche 3 (%)
Solar PV 1	26.4	27.8	-8.9
Solar PV 2	-4.1	-5.1	-0.1
CCGT	0.0	-0.3	0.0
Battery 1	-7.4	20.5	4.4
Battery 2	45.7	-2.8	4.5

Table 3.5: Percentage difference in system design variables in each niche before and after the resolution change

Table 3.3, 3.4 and 3.5 show that while the fitness values are generally the same or very close to each other, there is still a significant percentage difference for some technologies. The resolution change indicates that the largest capacity adjustments occur for battery and solar PV technologies, as these are particularly sensitive to the initialization of a higher time-of-day resolution. But the low percentage change in other technologies indicate that some optimal values have ben found for other technologies even before the high resolution is initialized. When the highest resolution is applied, some technologies have already reached their optimum and do not undergo significant adjustments. However, certain technologies, particularly those sensitive to a higher resolution, tend to change, though this does not greatly affect the sum of the fitness. Therefore, in this model at least, it is necessary to allow the model to run for some time after initializing the highest resolution.

In some cases, the lines show a steep decline, as observed in Figure 3.6. This behavior occurs when the niches explore alternative solutions, which are often close to the defined slack value. During mutation or crossover, these solutions are highly likely to cross the slack threshold, causing their fitness to reset to zero. The steep declines indicate that the algorithm attempts to mutate the values but fails to produce a better solution.

3.2.5. Spatial Distribution of Different Solutions

Figure 3.7 illustrates the values of the technologies for each niche. To compare the algorithm's performance against the optimal solution, the capacities of the cost-optimal solution are also included. Overall, the solutions vary significantly. In Niche 1, more solar power is used in Region 2, while battery storage is distributed between Regions 1 and 2. Niche 2 relies on solar power from both regions and battery availability in Region 2, whereas Niche 3 takes the opposite approach.

None of the solutions were able to significantly reduce the CCGT capacity, indicating that this technology is essential for system operation. However, the differences in CCGT capacity across Niches 1, 2, and 3 appear to follow a linear trend. This pattern is not only observed in CCGT but is also present, to some extent, in all technologies. This suggests that the algorithm performed as intended, successfully identifying maximally distinct solutions for each niche.

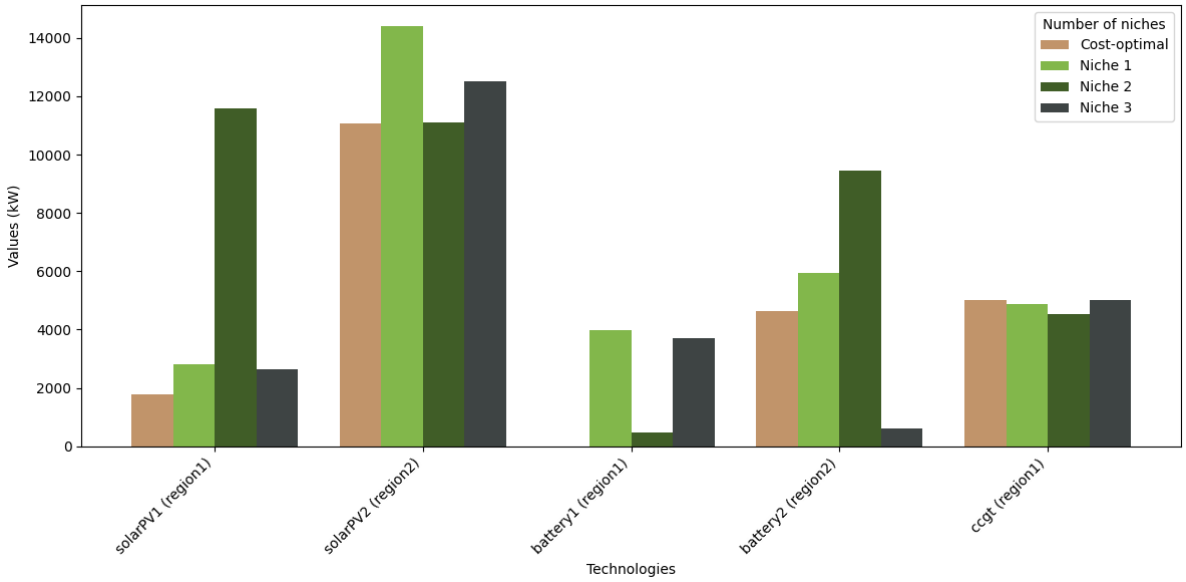


Figure 3.7: Bar graph comparing three different solutions and their technological capacities

3.3. Results Scalability, Evaluation and Benchmarking

Test runs were conducted iteratively until the algorithm reached a point where it was sufficient for generating data for the large model. During this process, several obstacles were encountered which required adjustments to ensure the algorithm would function as intended.

3.3.1. Structural Errors Encountered During Testing

The first structural issue identified was the ETA value. In the test model, the ETA value was set between 1 and 15, which was sufficient for the algorithm to function as intended. However, when applied to the large model, the smaller capacity values caused the ETA to behave unexpectedly. The ratio of the ETA value to the capacity values was significantly smaller compared to the mock-up model. As a result, individuals with capacities from the large model became stuck after only fifteen generations. To address this issue, a much higher ETA value was required. The revised ETA values now start at 1,000, are increased to 5,000 around generation 20, and reach as high as 10,000 by generation 60. This adjustment ensures that the algorithm maintains progress throughout the evolutionary process.

The test runs also showed that although the increase in memory usage was anticipated and accounted for in the SLURM file, the allocated memory still proved insufficient. Consequently, the memory allocation was increased from 128GB to 150GB. It was also observed that the job did not require 16 CPUs, so this was reduced to 5 CPUs, each with 30GB of memory. Finally, the time limit for solving the problem was initially set at 9 hours. However, it became apparent that 9 hours was insufficient. The time limit was subsequently increased to 24 hours, which is the maximum allowable time. The new SLURM file that was used can be found in appendix F.

3.3.2. High Resolution Initialization

The different resolution methods were evaluated to observe their influence on the total time required to solve the model. The time clustering and masking method run in some problems when applied to the large model. Normally, the process of initializing the model is time-consuming, but updating the model via the backend should not take long to solve. However, when clustering and masking are introduced, the process gets stuck for an extended time while creating the constraints. Additionally, when the model is sent to the solver, the solver struggles to generate an optimal solution in reasonable time. A process that should take no more than a second suddenly requires significantly more time.

The GA continuously updates the capacities and feeds them back through the backend of the model. This process occurs dozens of times per generation. With the time clustering and masking method the total solve time would exceed the 48-hour limit mandated by DelftBlue. Additionally. using a full resolution of 3 or even 6 hours would take an excessive amount of time to solve. Therefore, an alternative approach is necessary to ensure that high-resolution data still accurately reflects hourly intervals.

Table 3.6 presents the various options tested for full-resolution runs. Each final resolution is initialized at generation 95, and the total time represents the time required for the algorithm to complete generations 95 through 100. Moreover, the population size and number are set at 10 and 3 respectively. As shown, the masking and clustering methods perform significantly better in terms of time efficiency. However, a drawback is that these resolutions do not provide the same level of detail as the 3- or 6-hour resolutions.

Resolution	Time / Individual	Time / Generation	Total Time
3 hours	30 minutes	15 hours	75 hours
6 hours	15 minutes	5 hours	25 hours
1 month masking	2 minutes	1 hour	5 hours
k = 1 cluster	1 minute	0.5 hour	2.5 hours

Table 3.6: Time to solve an individual, generation, and total time for different resolutions

As a solution, the high-resolution runs will employ the masking method. While masking is not a viable options for low-resolution initialization, they are manageable at high resolutions. This approach ensures that the run can be completed within a reasonable time-frame while still incorporating hourly resolution for two of the most extreme wind events in a year. Initially, the resolution will start at 14 days and transition to the highest resolution at generation 95. The final 5 generations will be executed using the highest resolution.

3.3.3. Dynamic Slack Adjustment in Large-Scale Model

Gradually increasing the slack cost appeared to be an effective approach to prevent the algorithm from crashing when switching resolutions. The crash caused by the resolution change was no longer encountered. The difference in optimal cost was calculated between the initial resolution and the highest resolution. This difference came out to be 17%. when the cost would increase with 2% when switched to the highest resolution instead of introducing a slack of 17%, we subtracted the 2% plus a bit extra. So, the initial slack cost at the start of the GA run was set at 14%, and after the resolution switch this was set to 17%.

3.3.4. Final Genetic Algorithm Process

The final GA design after all the design changes is displayed in figure 3.8 as a flowchart. The process of the large model algorithm resembles the process of the small model algorithm, but the design changes discussed in the method are highlighted.

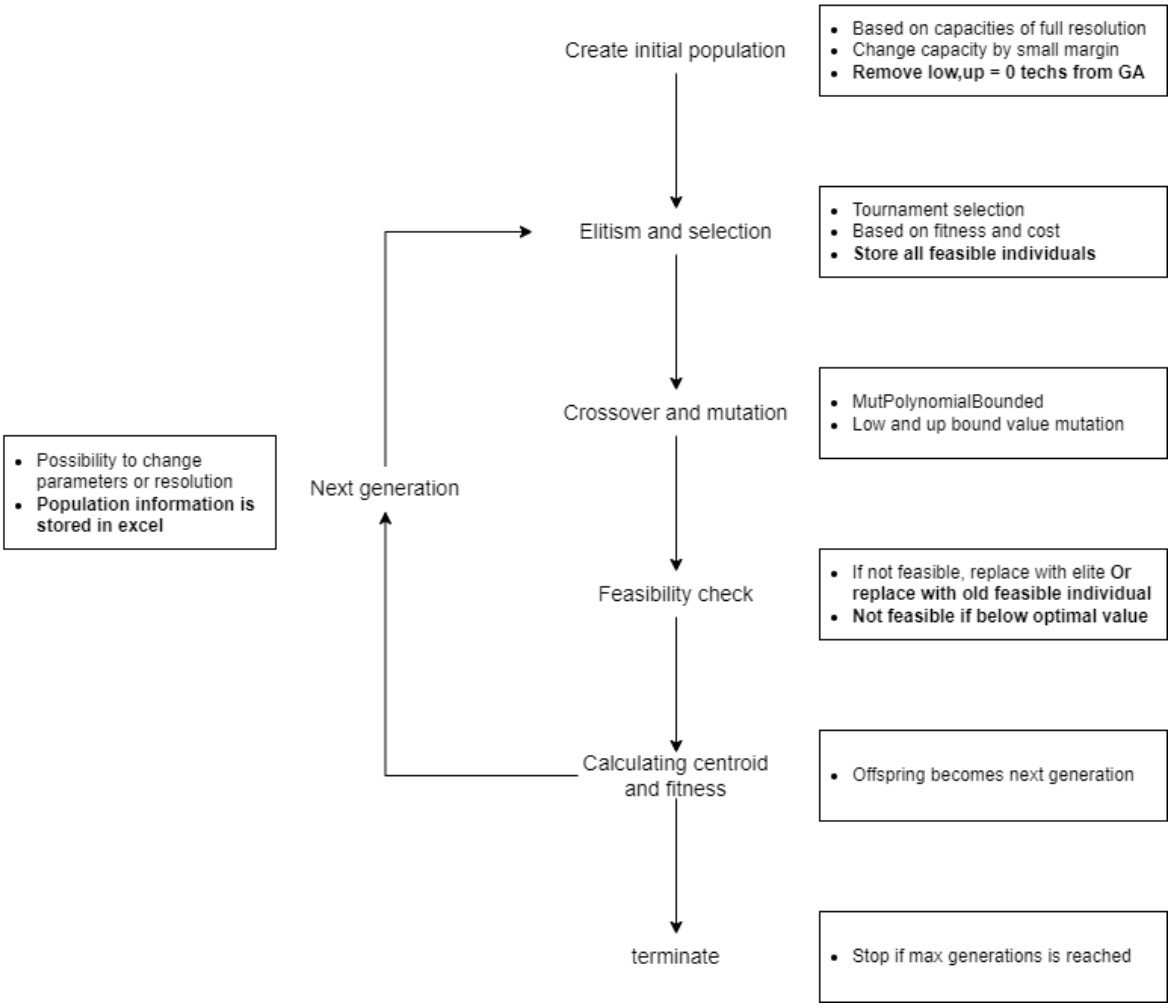


Figure 3.8: GA for Large Model Flowchart

3.3.5. Evaluation and Benchmark Results

Complete runs of the algorithm have been conducted. Based on prior runs and testing, the following parameters have been selected for use in the algorithm:

- `Generations = 100`
- `Number of populations = 3`
- `Population size = 10`
- `Crossover rate = 0.5`
- `Mutation rate = 0.2`
- `ETA1, ETA2, ETA3 = 1000, 5000, 10.000`
- `ETA change generation = 21, 60`
- `Resolution change generation = 95`
- `slack start = 0.05`
- `slack after res change = 0.1`

In total, the results of one complete algorithm run will be used for comparison. The results of the algorithm have been saved in an XLSX file. Each technology with its corresponding capacity value representing a column. The data has been restructured by extracting all offshore wind, onshore wind, PV, battery, and hydrogen technologies from the raw XLSX file. This is accomplished using Python, with the code provided in appendix G. A new XLSX file is created, where the capacities of these technologies are stored. Moreover, the code sums all the technology types together so it can be compared with the existing large model results. The raw XLSX file contains multiple sheets corresponding to different generations and the best-found individuals at the end. In the new file, the results are also stored across different sheets, with each sheet retaining the same name as in the original file.

3.3.6. Capacity Distribution Compared To SPORES

First the plots have been created as explained in the methodology section. Figure **??** show the three niches and the 50 SPORES results for each SPORES method. The GA niches results have been marked with a red circle. Because there are just 3 results generated by the GA, for each method the first three SPORES results have been plotted for a more fair comparison 3.10.

Table 3.7 presents the capacity values of the technologies for each niche. These values are derived from the optimal solution obtained after initializing the highest resolution. In this case, the final solution was determined at generation 100.

Niche	Total Wind (GW)	Total PV (GW)	Total Storage (GW)
Niche 1	4766.4	3983.6	3860.0
Niche 2	3786.3	5027.2	3750.4
Niche 3	4782.1	3233.5	3532.0

Table 3.7: Total capacity deployment for wind, PV, and storage across different niches.

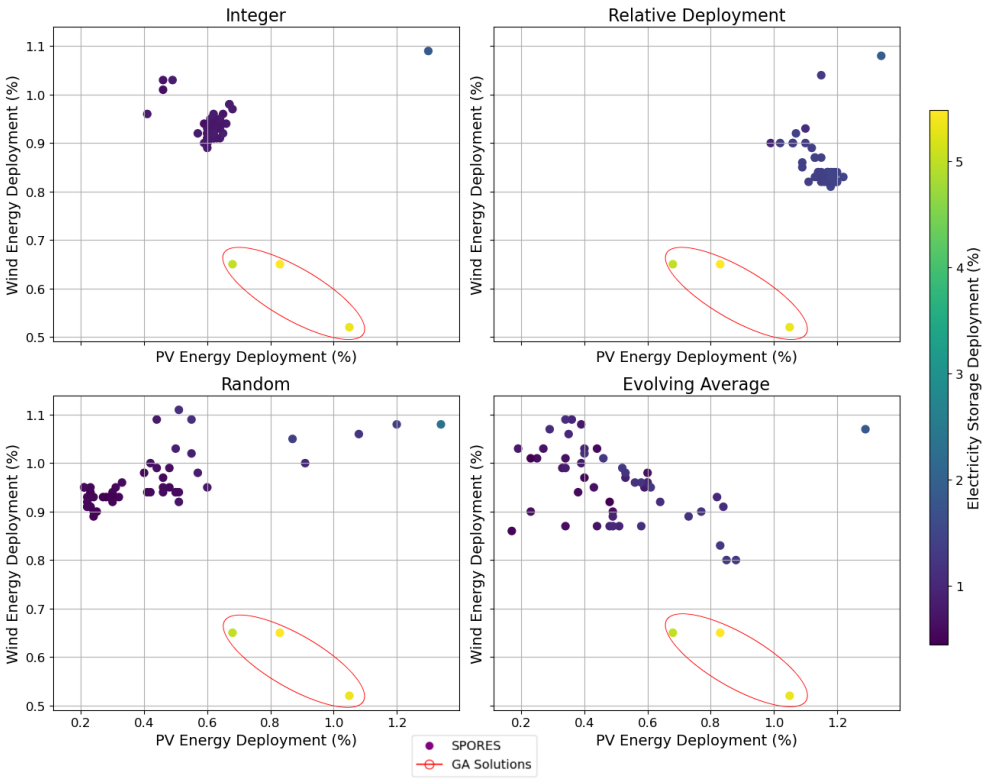


Figure 3.9: Algorithm result compared to SPORES

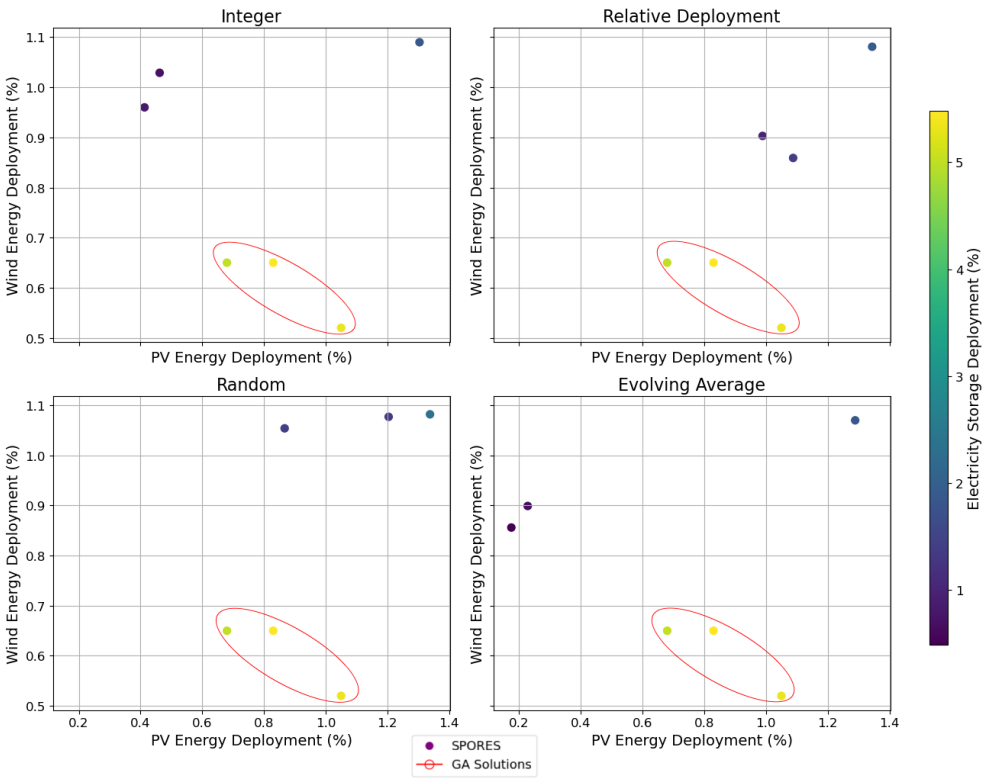


Figure 3.10: Algorithm result compared to first 3 SPORES results

The figures represent the four alternative weight methods. The relative deployment method is a method with a focus on spatial dissimilarity. The integer method is the simplest method, but due to its simplicity it has some drawbacks. With the random integer method, random numbers are assigned to the weights. And lastly, the evolving average method tries to retain more memory of the past iterations. More information about the method can be found in the original literature [12].

At first glance, the results appear relatively similar in terms of PV and wind deployment across the three GA solutions. Two solutions share the same wind deployment, while another two share the same PV deployment. However, the most notable differences between the GA solutions are observed in storage capacity, as illustrated in Figure 3.9. The GA identified three distinct storage capacities, with a general difference of 1.5 in normalized values.

Figure 3.9 shows that PV capacity performs similarly overall when compared to the SPORES results, with the three GA solutions being fairly close to other SPORES outputs, except for those generated using the relative deployment method. Wind energy deployment, however, is where the GA consistently identifies solutions with lower capacities compared to the optimal solution. The lowest capacity identified by the GA uses only half the wind energy deployed in the optimal result. As previously noted, the largest differences are observed in storage technology, where the highest capacity found by the GA is seven times greater than the optimal value identified by the SPORES model. Figure 3.10 compares the GA results with the first three SPORES results from each method. This comparison reinforces the previous findings: the electricity storage capacity is significantly higher in the GA results, while wind energy deployment is notably lower.

All the niches have been compared to the optimal values. Table 3.8 shows the comparison between the niches and the optimal values.

Comparison	**Diff Wind %**	**Diff PV %**	**Diff Storage %**	**Euclidean Distance**
Niche 1 vs Optimum	63.12	19.82	-81.82	432.08
Niche 2 vs Optimum	68.54	-4.26	-81.16	425.42
Niche 3 vs Optimum	52.60	46.66	-80.17	409.44
Total Difference Optimum	61.42	20.74	-81.05	422.31

Table 3.8: Pairwise Percentage Differences and Euclidean Distance between Niches and Optimum

The percentage difference is calculated by dividing the optimal value by the niche value and then subtracting 1. This shows how much the optimal value deviates proportionally from the niche value.

The Euclidean distance is calculated using the following formula:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (u_1 - u_2)^2}$$

where:

- x represents the total wind capacity,
- y represents the total PV capacity,
- u represents the total storage capacity,
- 1 denotes the niche values,
- 2 denotes the optimal/SPORES values.

This distance measures the overall difference between the niche and optimal configurations in terms of capacity. Table 3.8 shows that the differences are smallest in some cases, with the lowest being a 4% difference. However, when examining wind and storage, it becomes evident that the niche configurations do not closely compare to the optimum. The smallest difference in wind capacity is 52%, while for storage, it is 80%. The Euclidean distance was used to compare the differences across all three technologies, but they all appear to be roughly at equal distance from the optimum, with only minor variations.

The same measurements were conducted for all the SPORES solutions. Appendix H contains all the tables with detailed comparisons. Table 3.9 summarizes these findings by presenting the highest value, lowest value, and the value closest to zero for all comparisons. This provides an overview of the range of percentage differences and Euclidean distances while also identifying the closest match to one of the niche configurations. Once again, it can be concluded that while PV exhibited a wide range of variation, some solutions had very similar PV capacities. The smallest difference in wind capacity between two SPORES methods was 30%, while for storage, the closest match was 54%.

	Wind %	**PV %**	**Storage %**	**Euclidean Distance**
Highest value	83.56	96.39	-54.18	978.04
Lowest value	30.70	-83.26	-91.13	223.29
Closest to 0	30.70	3.47	-54.18	223.29

Table 3.9: Summary of Wind, PV, Storage, and Euclidean Distance Values

To come to a better conclusion on how the algorithm arrived at these findings, we will take a closer look at the results generated by the algorithm. Figure 3.11 illustrates the total deployed RES plotted against the total deployed storage technologies across each generation. Generation 1 starts with many individuals clustered in the same region of the plot, as their capacities are all based on the same optimal solution values. By generation 20 it is noticeable that all the niches are evolving in the same direction. This indicates that both their RES and storage technology deployments are progressing in a similar manner. It appears that increased RES deployment is accompanied by a proportional degree of storage technology deployment. Eventually the mutation value rate is lowered and the maximum barrier is being reached. This results in the niches slowly converging into one similar answer.

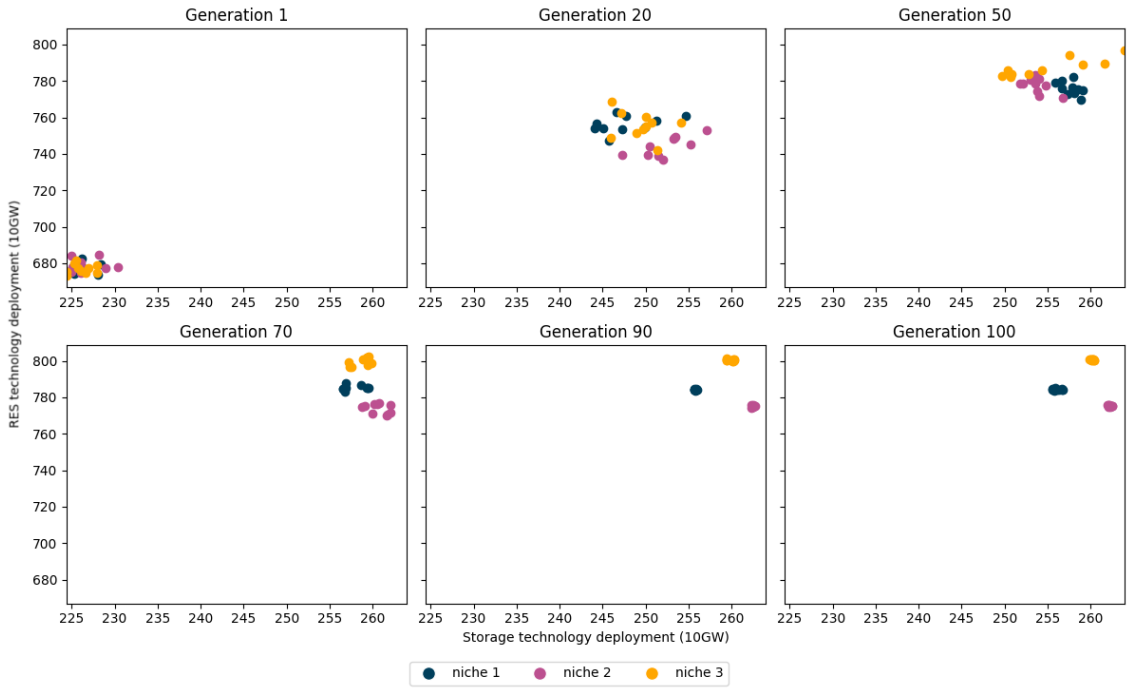


Figure 3.11: Evolution of Storage and RES Technology Deployment Across Generations

To provide a more comprehensive overview of the algorithm's behavior, it is essential to examine the development of both the cost and the fitness of the individuals within the subpopulations. Figure 3.12 illustrates the mean cost and mean fitness of the individuals. The left plot represents the average cost, with the cost values shown on the y-axis. The right plot represents the average fitness, with fitness values also plotted on the y-axis. In both figures, the x-axis represents the generation in which the corresponding results were generated.

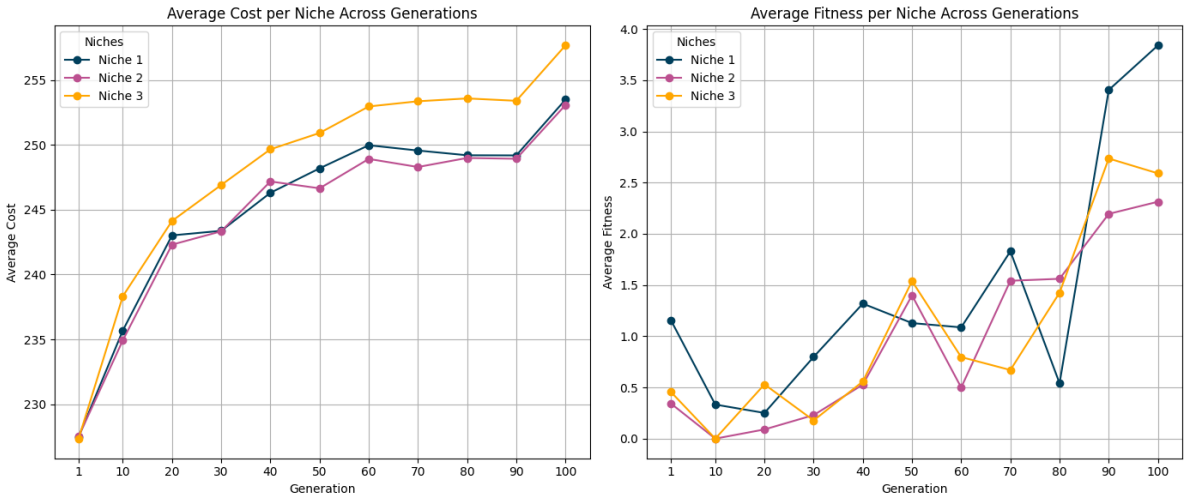


Figure 3.12: Mean Cost and Fitness of Each Niche Across Generations

The cost graph indicates that each niche is gradually evolving toward the set slack. This can be seen in the average cost of each niche stabilizing after around 60 generations. However, when the slack bound is loosened at the resolution change occurring in generation 95, a rise in cost becomes noticeable again.

After the start of the algorithm process, the fitness of each niche decreases to its lowest value throughout the entire process. It is only after 20 generations that the fitness across the niches begins to increase gradually. Significant improvements in the average fitness of the niches are observed around generation 60, with a more noticeable rise occurring by generation 80.

3.3.7. Spatial Capacity Distribution of Niches

The objective of the algorithm was to generate diverse capacity deployment solutions for a spatial energy system. To better understand the algorithm's behavior concerning the spatial distribution within the country, the following approach was undertaken. The model subdivides the country into multiple regions, each representing a distinct spatial unit. To enhance the comprehensibility of the data, the results from all regions within a country were aggregated, allowing for the visualization of the total capacity deployment at the national level. Figures 3.13, 3.14, and 3.15 illustrate the deployment of wind, photovoltaic (PV), and battery storage capacity across each country, respectively. The different niches are plotted adjacent to one another to emphasize the variations in deployment across the different solutions.

The algorithm successfully generated distinct capacity deployment solutions for each niche. When analyzing the deployment of a specific technology, it becomes evident that the capacity distributions across different countries are rarely identical. Moreover, the algorithm demonstrated the ability to produce highly divergent solutions.

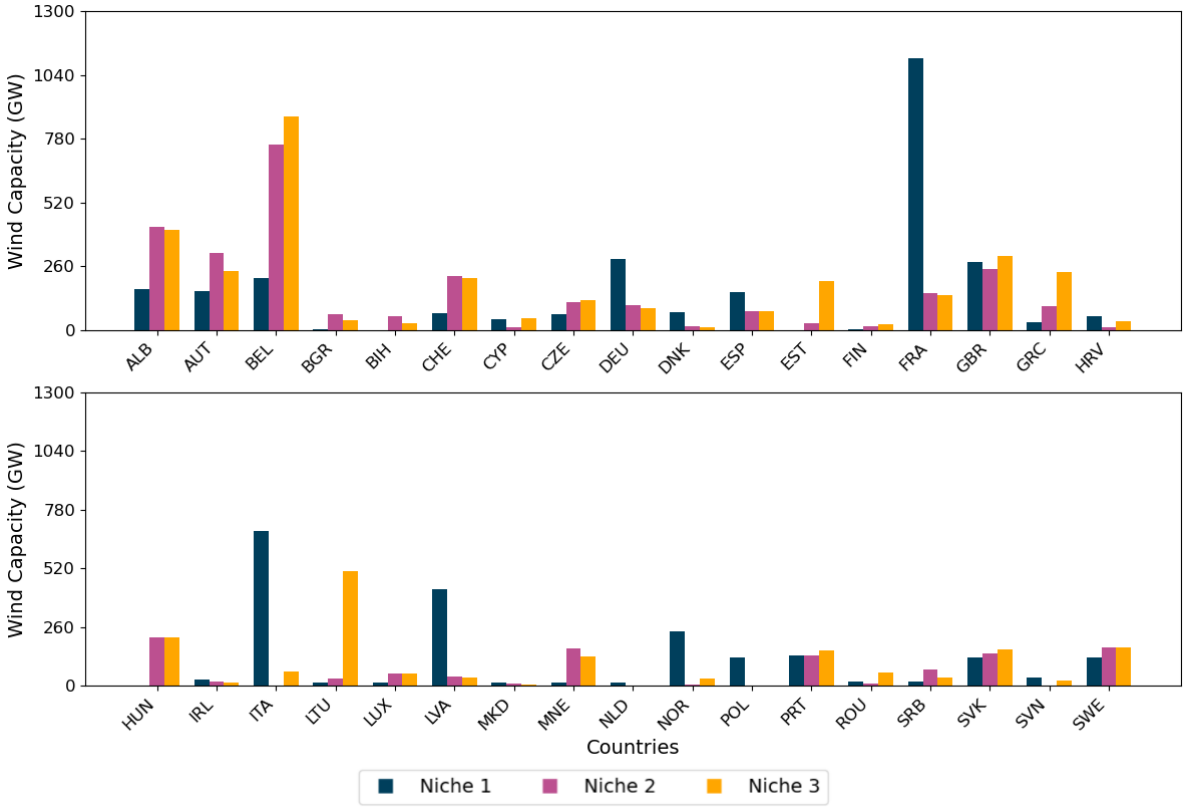


Figure 3.13: Spatial wind capacity deployment per country in GW for each niche

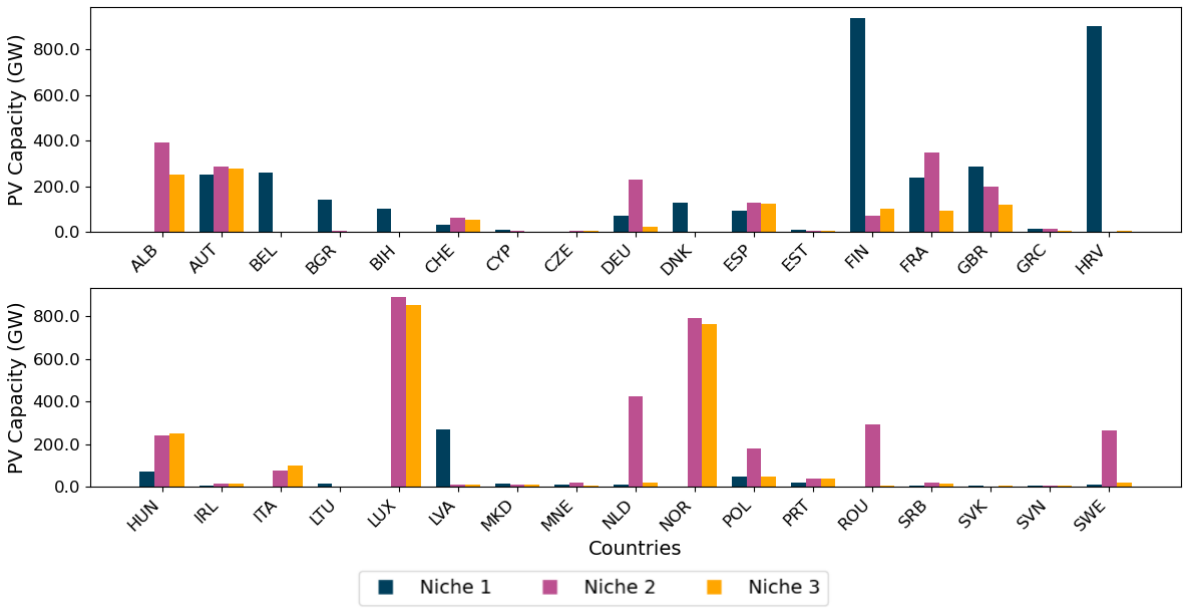


Figure 3.14: Spatial PV capacity deployment per country in GW for each niche

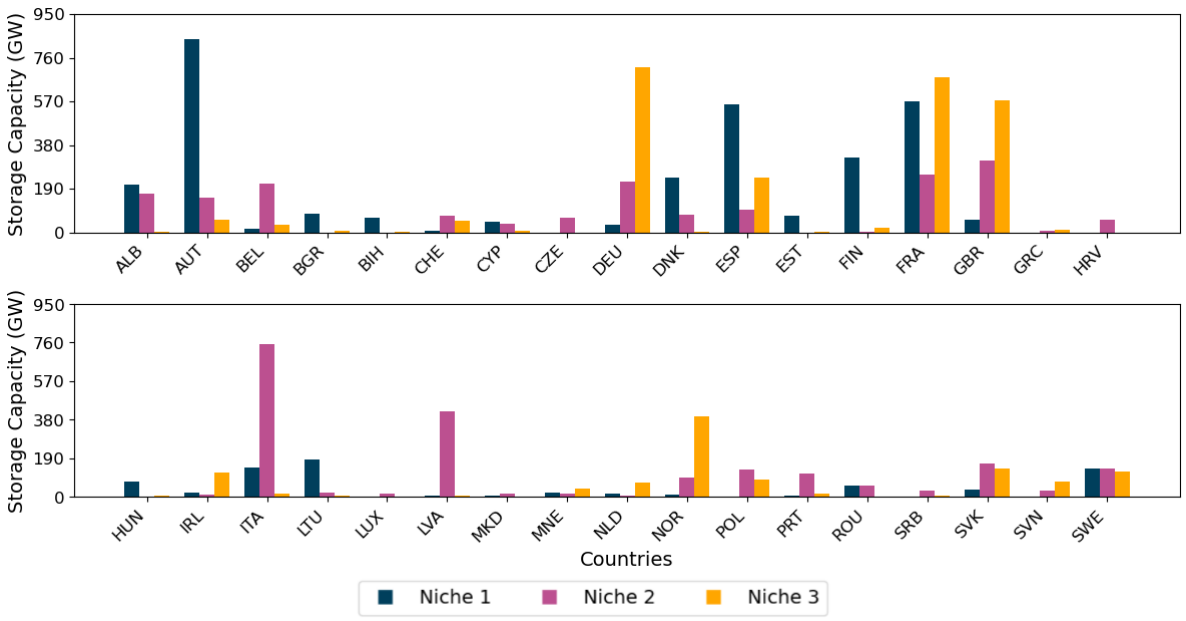


Figure 3.15: Spatial battery capacity deployment per country in GW for each niche

Figure 3.13 presents the wind capacity deployment across European countries for the three different niches. The distribution varies significantly among niches, illustrating the diversity of solutions generated by the GA. In some cases, such as Italy, France, and the Latvia, niche 1 exhibits a dominant allocation of wind capacity, while niche 2 and niche 3 allocate capacity more evenly across multiple countries, such as Belgium and Hungary and Albania. These differences highlight how the GA is able to produce alternative spatial strategies.

The deployment of PV capacity, shown in Figure 3.14, follows a different spatial pattern compared to wind. The algorithm has assigned significant PV capacity to Finland and Croatia in niche 1, while niche 2 has concentrated higher allocations in Luxembourg and Norway, with also smaller deployments across Europe. Niche 3, on the other hand, distributes PV capacity more broadly, with notable allocations in Luxembourg and Norway. The variation between niches indicates that different deployment strategies were created, favoring large-scale concentration in a few countries but also deploying smaller amounts in different countries.

Figure 3.15 illustrates the spatial allocation of battery storage capacity across European countries. The results show a strong differentiation between niches, with niche 1 concentrating storage in Austria, Spain, and France, whereas niche 2 prioritizes Lithuania and Italy and a more widespread allocation. Niche 3 deploys the storage in France, Germany, Norway and the UK.

4

Discussion

The discussion section will take a closer look at the results presented in Chapter 3, providing a more detailed analysis. It will explain the significance of the results and their implications. This analysis will help in drawing conclusions for each sub-question and the main research question, which will be addressed in Chapter 5.

4.1. Discussion Literature Review

Both the SI and EC approaches have their advantages and disadvantages, which vary depending on the context of their application. To determine which method is the best option for answering the research question, a more in-depth analysis is needed, incorporating additional characteristics of both methods. However, based on the existing literature, ECs combined with MGA approaches are more commonly discussed. The literature is transparent and understandable and can help informing the development of an algorithm aimed at addressing the main research question of this thesis. The literature describes the use of adjusted evolutionary algorithms, their application to case studies, and argues for further application as the results have been promising [10] [17].Existing research already utilizes a Python EC framework called DEAP, which provides an easy way to build custom EC algorithms, such as GA [13]. Given the wider availability of examples and detailed explanations for EC methods in the reviewed literature, this thesis will adopt an EC algorithm to integrate bio-inspired heuristics with the MGA approach. Using an EC method provides a solid foundation for developing and tailoring an algorithm that can effectively address the research question.

4.2. Discussion Algorithm Configuration

The analysis of crossover, mutation, and ETA identified seven effective parameter combinations that yielded results with no statistically significant differences. All the combinations utilized a mutation rate of 20%. A high mutation rate allows the algorithm to explore the solution space more thoroughly. For the scope of this model, a 20% mutation rate was highly favored, ultimately producing better results. In contrast, the ETA value and crossover rate did not exhibit such consistency in their effects, making it more challenging to select the best combination solely based on fitness values. However, when the combinations were also evaluated based on the generation in which the best individuals were found, combination 24 outperformed the others on both criteria.

A mutation rate of 20% is generally considered a high mutation rate. For this reason, it was selected as the highest mutation rate used for the test runs. However, since all non-significant results were associated with a mutation rate of 20%, it is difficult to conclude with certainty that this is the optimal mutation rate. It is possible that a higher mutation rate could be more effective, but this has not been tested.

For the number of niches and the size of the population, an increase in the number of niches resulted in a linear increase in computation time. This is because more individuals need to be evaluated, which is the most time-consuming part of the algorithm. Additionally, an increase in niches caused greater crowding in the solution space, reducing the algorithm's ability to find diverse and optimal solutions compared to scenarios with fewer niches.

Increasing the population size demonstrated a positive linear effect on fitness quality. This outcome is logical, as a larger population increases the likelihood of discovering better solutions within the solution space. However, increasing the population size also led to longer computation times. Similar to increasing the number of niches, this increase in time is due to the higher number of individuals that need to be evaluated.

The resolution change parameter proved effective in reducing computation time. Although not flawless, resolution masking successfully generated individuals that better accounted for time-sensitive technologies, which are more prominent at higher resolutions. However, using the resolution change sometimes caused the algorithm to crash. This occurred because some individual solutions found at a low resolution became infeasible when their capacities were evaluated for feasibility at full-scale resolution.

This method provided valuable insights into the interaction between algorithm parameters and their influence on both performance and computational efficiency. By introducing a structured approach to parameter selection, the method enhances the scientific rigor of algorithm design optimization. This approach not only reduces structural complexity but also facilitates a more systematic and efficient tuning process. Additionally, the dynamic resolution adjustment technique demonstrated its potential to improve computational efficiency, highlighting its applicability in optimization problems.

Building on prior research that explored the combination of GA and MGA in small-scale optimization problems [71], this study extends these concepts to a new domain. The successful integration of GA-MGA into energy system optimization underscores its adaptability and effectiveness in generating diverse yet feasible capacity configurations. The results indicate that this method can uncover alternative solutions within the solution space, demonstrating its capacity to balance exploration and exploitation.

4.3. Discussion Algorithm Scalability

The structural design choices that have been made seemed to operate as intended. But there were some problems with the implementation of the algorithm to on a larger scale. Test runs were conducted to identify the components that caused errors, which were then fixed. The choices made were effective to some extent but may not necessarily be the optimal solutions. This section will reflect on the most important design choices, their performance, their limitations, and potential alternatives that could have been implemented.

4.3.1. Selection Operator and ETA

Changes were made to the selection operator and the ETA value, which is a part of the mutation operator. The selection operator was modified to play a more prominent role, ensuring that if all individuals in a generation mutated into infeasible solutions, they would not be used for mutation in the next generation. Instead, they were replaced by previously feasible solutions. The ETA value that worked for the small model did not perform well for the large model due to differences in capacity values. As a result, ETA values were adjusted to better suit the large model's capacities. Both operators functioned as intended, with the selection operator preventing the algorithm from crashing and the ETA value successfully mutating individuals and generating new solutions. However, due to the stochastic nature of the algorithm, the exact impact of these adjustments on performance remains uncertain. This is primarily due to a lack of extensive testing. To better understand the effects of these operator changes, multiple runs should be conducted under different conditions to evaluate their performance more thoroughly.

4.3.2. Resolution settings and Initialization

The test runs with the large model also showed that the initial resolution options were not feasible. The desired solutions were as followed:

- Start resolution: 14 days with the min/max masking of wind for two weeks
- End resolution: 3-hours

The reason for these resolution not being feasible is that it took to much time for the solver to solve an individual. Even overwriting the allowed time to run file on the DelftBlue cluster. The next part explain what the reasons is for the high solving time of an individual.

When a model is created via Calliope, several unseen steps take place behind the code. During initialization, a model_run dictionary is created using the provided YAML and CSV files. At this stage, any specified overrides, whether from scenarios or location/link-specific adjustments, are applied. The model_run dictionary is then reformulated into multidimensional arrays and collated into a dataset called model_data, which is managed using xarray. At this point, the model initialization is complete, allowing the user to access and, if necessary, edit the model inputs.

When the model.run() command is executed, only the model_data is sent to the backend, where a Pyomo concretemodel is created. The Pyomo model's parameters and are created using data from model_data. Additionally, decision variables , constraints , and the objective function are initialized. The model is then sent to the solver.

After the problem is solved, the backend Pyomo model is attached to the main Model object, and the results are added to model_data. Post-processing is performed to clean up the results and calculate various indicators, such as the capacity factor of technologies. At this stage, the model run is complete, and the results are accessible for saving, analysis, or further use [77]. Figure 4.1 shows a visualization of the internal workflow previously described.

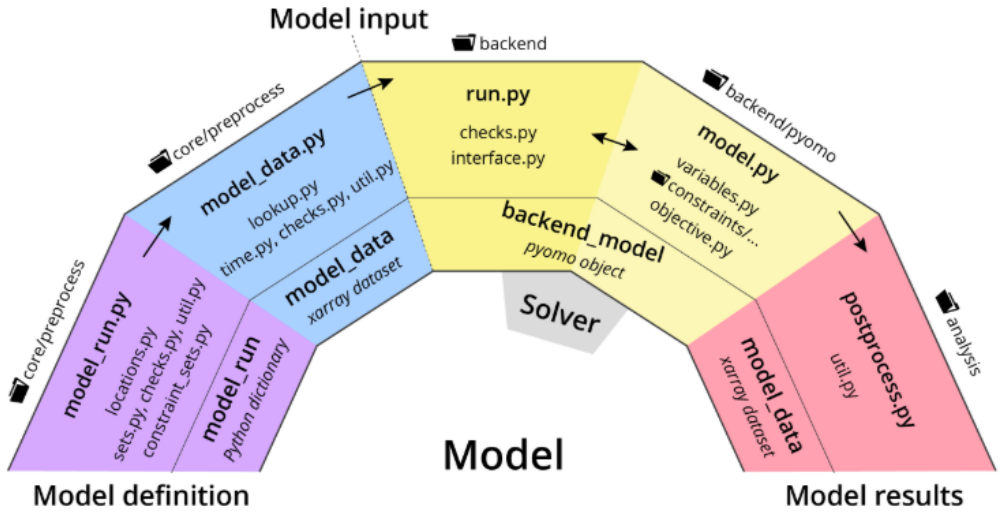


Figure 4.1: Representation of Calliope internal implementation workflow [77]

The root of the problem could have multiple causes. It can be that it just takes a long time to solve the problem, which is the case for higher resolutions. But previous testing with the time-clustering and masking methods in Calliope have proven to be bugged at times.

When resampling the masking method, index errors occurred. This misalignment could result in the creation of more complex constraints that the model must satisfy, making it even more challenging for the solver to find an optimal solution. The problem could also originate from the solver itself. During the model setup, certain solver options need to be specified. Options such as feasibility tolerance, mode, and threads can significantly influence how the solver handles the model optimization problem. If the problem occurs under a broader range of circumstances, it may indicate an issue that needs further attention. Solving such a problem could lead to a significant increase in time. Time masking and clustering, in particular, offer unique design options that can be effectively integrated into energy system models.

As a result, a different approach was taken that would still incorporate a resolution switch, but mitigated the time required to solve for a full resolution. The current resolution would be as followed:

- Begin resolution: 1-month
- End resolution: 14 days while min/max masking wind for two weeks

With these resolutions, the algorithm can be solved for 100 generations within a reasonable time. Moreover, the highest resolution still retains an aspect of full-scale resolution. In this case, instead of a 3-hour time-frame, it spans two weeks—capturing the minimum and maximum extremes of wind—at a 1-hour resolution.

Although the high-resolution model incorporates hourly data and can be solved within a manageable time, a 3-hour resolution is still highly preferred. This aligns with the resolution of the large-scale model and has also been shown to be well-suited for energy system models [49]. Optimisation of the resolution switch, but also the performance of the algorithm in general could realize the desired 3-hour resolution.

4.3.3. Slack initialization

During the resolution switch, the cost of individuals would slightly increase. To address this, a slow slack adjustment was implemented, which gradually increased the slack when the resolution changed. This feature worked as intended; however, the full impact of this adjustment should be further tested and researched. Such tests could help determine the optimal slack values for the best results. Ideally, the switch would not be necessary. Investigating why the cost changes during resolution switching and exploring ways to mitigate this increase could eliminate the need for slack adjustment altogether.

4.4. Discussion Evaluation And Benchmarking

4.4.1. Battery Capacity Distribution

The significant difference in storage capacity can be attributed to the variation in resolution between the SPORES results and the GA. The SPORES results are based on a 3-hour resolution, whereas the GA employs a masking approach with a 2-week period at an hourly resolution. This difference in resolution appears to facilitate the deployment of storage technologies in substantially larger quantities.

The initial capacities of the individuals in the subpopulations are determined based on the full-scale resolution used by the model. They are then randomly modified by a small amount to introduce divergence while ensuring all individuals remain feasible.

In this case, the full-scale resolution spans 14 days, with min/max masking applied to wind data over a two-week period. When solved at this resolution, the storage capacity is already significantly higher compared to the SPORES results. Since all individuals have storage capacities around these values and this high storage capacity is already optimal, the algorithm may struggle to find solutions with lower storage capacities.

The reason for this could be as follows:

First, a storage capacity must be selected for mutation by a significant amount, with the mutation amount not increasing, but decreasing the capacity. If this occurs, the mutation must ensure that the individual remains feasible. This must take place several times to decrease the storage, as multiple technologies make up the total storage capacity.

4.4.2. Wind Capacity Distribution

Another conclusion that could be drawn is that the GA structurally deployed less wind energy compared to the SPORES results. Several factors could explain this. For instance, the availability of substantial storage capacity reduces the need for RES deployment, as sufficient energy can be retrieved from the batteries whenever needed. As a result, the GA tends to explore solutions with reduced reliance on RES. This outcome may again be influenced by the initialized population values, but the low resolution used in the GA model could also play a significant role in the lower deployment of wind energy compared to the SPORES results.

4.4.3. Algorithm Process Evaluation

Some observations can be made by examining Figures 3.11 and 3.12. We discuss the fitness behavior in relation to the cost. It appears that the algorithm is unable to improve the fitness of individuals while they are still able to increase system costs, as seen in 3.12. Only when system costs can no longer rise does the fitness increase significantly. Ideally, however, fitness would gradually improve with each generation.

This issue is also evident when examining the capacity deployment of storage and RES technologies across generations, as seen in 3.11. Instead of improving fitness over time, the niches evolve collectively in the same direction. This behavior prevents the algorithm from optimally exploring the solution space. Feasible solutions exist at lower costs, but the algorithm forces itself to prioritize higher-cost solutions. Only when this potential limit is reached does the algorithm begin to explore the solution space more effectively, allowing the individuals to converge.

Generally, the issue could stem from how the parameters are configured or how the operational process is structured. While it is possible that the problem is caused by the parameters and could be resolved by identifying better-suited parameter combinations. Or it could even be how Python handles the build in operators designed by DEAP library. But for simplicity, we assume that this behavior originates from the algorithm's process itself.

It must be noted that this algorithm heavily emphasizes the use of the elite and selection operators. Their function has been enhanced to reduce the likelihood of getting stuck in a cycle of only producing infeasible individuals. which was a substantial problem during the designing and testing of the algorithm. Especially with the introduction of resolution change.

At the start of the algorithm, individuals with higher costs are easier to produce, and these individuals are favored by the algorithm. This process works as follows: Higher-cost individuals are the result of larger capacity mutations. These large mutations are possible due to the cost slack not yet being reached. These larger capacity mutations lead to the individual having a high fitness value. So, due to the strong elitism and selection mechanisms, individuals with higher costs are heavily favored because of their associated higher fitness.

However, this selection process is applied to all the individuals in each subpopulation. While the RES and storage capacities are increasing for all the individuals, their fitness stays relatively close to each other due to them growing together. As a result, the algorithm can only begin identifying individuals with higher fitness once they can no longer drastically increase their cost.

One could argue that by around generation 20, when the individuals are clustered in the middle of the plot, they have an equal opportunity to move away from high-cost solutions and begin exploring lower-cost solutions more thoroughly. It needs to be explain therefor what causes them to still favor higher cost individuals over time.

The system that the algorithm tries to optimize is fairly complex. An individual consist out of more than a 1000 technology capacity values, with the technology also being bound to a specific location in the system. some main structures of the algorithm need to be explicitly mentioned again here:

- The algorithm uses the optimal capacity values for initializing the individuals
- mutation range is based on the minimum and maximum allowed capacity of the technology

It is possible that, because the capacities are based on the optimal value, reducing these values renders the system infeasible by creating unmet demand. On the other hand, increasing the capacity is almost always feasible, provided it remains within the system's cost range and the technology's minimum-maximum bounds. Furthermore, when a capacity mutates to a higher value, the likelihood of it mutating back to its exact original lower value is very low due to the large number of technologies in the system. Additionally, if technologies are near their lower capacity bounds, the likelihood of their capacity mutating to a higher value becomes more pronounced.

Another factor that could significantly impact the quality of results is the resolution change. The sudden increase in cost observed in the last 10 generations is due to individuals mutating to higher costs than before the resolution change. This is a consequence of the slack mechanism introduced in Section 4.3.3. Given the limited number of generations after the resolution change, it is possible that the individuals did not reach their sub-optimal location. While it appears that they have converged, the large ETA value may have prevented the individuals from mutating away from the optimal solution found before the resolution change. This again highlights that while resolution change is a powerful tool for reducing computational cost, it introduces considerable uncertainty regarding the quality of the results.

Lastly, when analyzing Figure 3.11, it appears that the algorithm is functioning as intended. Initially, individuals are relatively scattered, but over time, the solutions begin to converge. By the final stages, all solutions overlap, and the centers of the niches form a triangle, with each point approximately equally distant from the others.

4.4.4. Spatial Capacity Distribution

As highlighted in the results, the algorithm successfully generated distinct spatial distribution patterns for energy system solutions. The following section of this thesis will first explore the potential factors contributing to these outcomes. Subsequently, the results will be compared with those obtained by Lombardi et al. [38] to assess similarities between results.

The algorithm was successful to find deployment for all technologies in all countries. Even in countries when the deployment of a technology was extremely close, the algorithm succeeded in generating a small difference between the capacities. The large difference between the capacity deployments make the solutions extremely interested to be used for future decision making.

Some of the previously discussed shortcomings also apply to this section of the results. While the findings are insightful, the transition to a higher resolution, which could not be run for many generations, introduces additional complexity and makes the results more confounded. Over time, the algorithm converges to a local optimum. However, when the resolution is increased, the niches are suddenly able to escape this convergence, allowing them to explore the newly expanded solution space once again. As a result, it is impossible to determine with certainty whether the identified solutions are truly optimal within the high-resolution solution space.

The results of the algorithm can also be compared to the previously generated SPORES results. The study by Lombardi et al.[12] presented a summarized overview, displaying only the total deployed capacity for each technology. As a result, a quantitative comparison between the algorithm's outcomes and the SPORES results is possible.

However, the study by Lombardi et al. also included figures illustrating the spatial deployment of wind, PV, and storage technologies across Europe. These figures specifically depict the SPORES solution with the lowest

concentration of onshore wind capacity while simultaneously minimizing bio-energy capacity. Figure 4.2 presents the spatial distribution of wind and PV deployment, while Figure 4.3 illustrates the storage discharge capacity of hydrogen and batteries. In the results generated by the algorithm, this storage discharge capacity of these two technologies is represented as total storage.

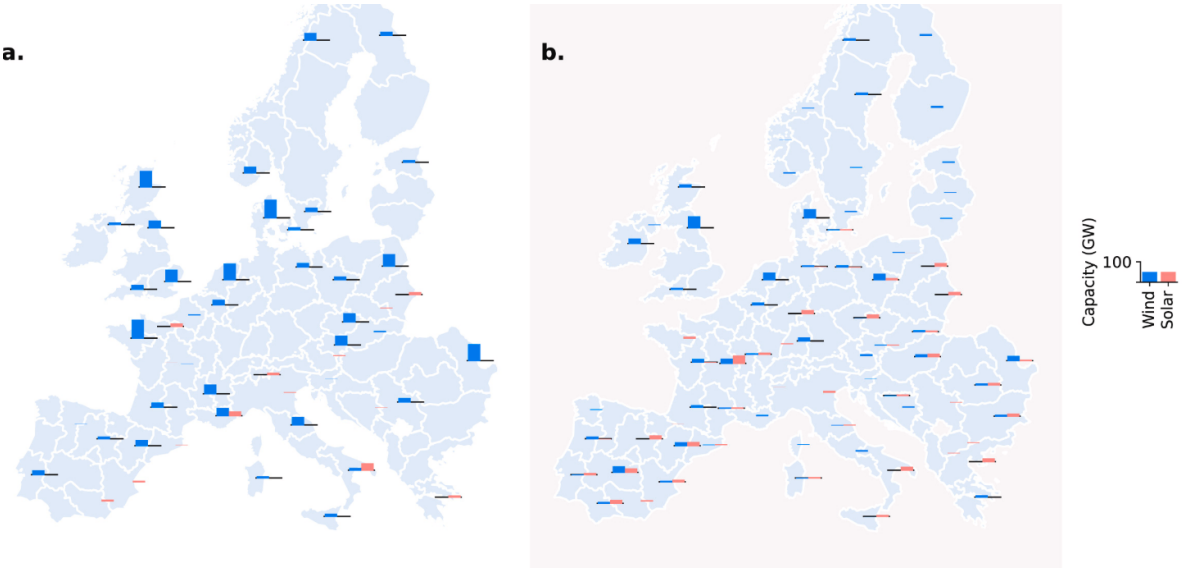


Figure 4.2: SPORES solutions with minimal onshore wind and bioenergy capacity in the random (left) and relative-deployment (right, grey background) solution spaces. Panels (a–b) show the deployment of rooftop and open-field PV alongside onshore and offshore wind across 97 model locations, where capacity exceeds 10 GW. [12]

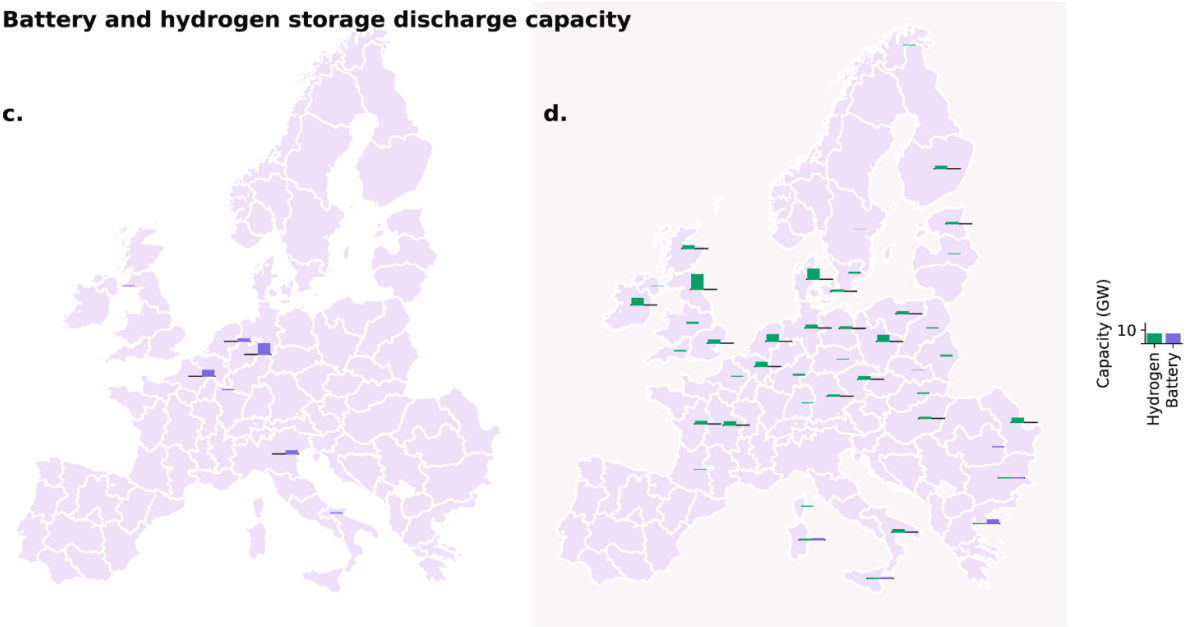


Figure 4.3: Panels (c–d) illustrate the spatial deployment of battery and hydrogen storage (discharge) capacity in regions where the locally deployed capacity exceeds 1.5 GW. [12]

When comparing Figures 3.14 and 3.13 to the spatial deployment presented by Lombardi et al. (4.2), it becomes

evident that the SPORES results depict a more dispersed deployment of wind and PV across multiple countries, whereas the GA results exhibit a more concentrated allocation in specific regions. The SPORES method achieves moderate wind and PV deployment, resulting in a more balanced and optimized spatial distribution. In contrast, the GA approach creates larger disparities between countries, with France, Italy, Finland, Latvia, and Luxembourg experiencing significantly higher deployment levels than others.

Similarly, when comparing Figure 3.15 to the European storage map (4.3), it is again apparent that SPORES results demonstrate a wider geographic distribution of storage across Europe. The GA method, on the other hand, leads to a highly concentrated deployment in a few key countries while also allocating significantly more total storage compared to the SPORES results.

A major distinction between the two methods lies in their treatment of transmission capacity. The GA algorithm does not include transmission when mutating and crossing over the individuals value. SPORES does explicitly integrates transmission capacities into the optimization process. As a result, the GA does not fully account for transmission constraints and cross-border energy flows, unlike the SPORES approach, which optimizes the energy system holistically.

Consequently, the GA prioritizes economic efficiency, placing capacity in locations where it is most cost-effective. In contrast, the SPORES method emphasizes spatial diversity, thereby reducing risks associated with local grid constraints and transmission limitations.

The comparison between GA and SPORES results reveals that, while the GA can generate distinct solutions, the variation between niches is less pronounced than in SPORES. This is evident from the high-capacity deployment at specific locations in the GA results, whereas SPORES distributes deployment more evenly.

The observed behavior of the GA is not necessarily indicative of a malfunction. Similar to the SPORES approach, the results are expected to exhibit variations, though not to extreme degrees. This aligns with previous studies that have combined GA with MGA [17]. The significant differences in technology deployment observed between SPORES and the GA-MGA approach may stem from the initial structural differences in the models or algorithms rather than from the optimization method itself.

As demonstrated during the scaling process, even small modifications in algorithm components or decision-making processes can have a considerable impact on performance. Due to the stochastic nature of GAs, assessing their performance requires multiple test runs to ensure reliable conclusions. Structural changes in the algorithm can lead to substantial variations in outcomes, a well-documented phenomenon in GA design [58].

Beyond optimizing the structure of the algorithm, it is also crucial to align its design with the methodological approach used to achieve SPORES. Differences in fundamental modeling choices, such as the inclusion or exclusion of transmission technologies from the GA-MGA process or variations in resolution, can result in distinct solution spaces. Without systematic testing of the algorithm's structural components and design choices, drawing definitive conclusions about its overall effectiveness remains challenging. But the algorithm has been successfully applied to a large-scale energy system model, producing results that can be benchmarked against existing data. This demonstrates its feasibility in real-world applications and highlights its potential for further validation.

5

Conclusion

5.1. Answer Sub-question 1

5.1.1. Summary of Key Findings

The literature review revealed a significant gap in research combining MGA methods with bio-inspired heuristics for spatial energy system optimization. While MGA is widely applied in energy optimization, no existing studies integrate it with bio-inspired heuristics in this context. The closest work, Prina et al. [13], utilizes a multi-objective optimization approach but does not incorporate MGA. Among the studies that do explore MGA with bio-inspired heuristics, Evolutionary Computation (EC) methods, especially GA, dominate the field. The extensive use of GA-based MGA frameworks suggests that EC approaches are more established and widely accepted in the literature. Given the strong theoretical foundation and broader availability of research on EC-based MGA approaches, this thesis will adopt an Evolutionary Computation method. The choice is further reinforced by the existence of well-documented implementations, such as DEAP, which have been successfully applied in scientific research. This ensures a robust, adaptable, and reproducible algorithm development process within the context of energy system optimization.

5.1.2. Answering The Sub-question

Sub-question: Which bio-inspired metaheuristic algorithm is best suited for the method being used?

The most suitable bio-inspired heuristic algorithm for the method being used is Genetic Algorithms (GA). Not only has GA been previously applied to energy system optimization problems, but there is also existing literature that explains in detail the theory and mathematical approach for combining GA and MGA in a broader optimization context. Furthermore, tools such as DEAP provide accessible frameworks for programming a GA-MGA algorithm, making its implementation more straightforward. Therefore, while GA may not necessarily be the best metaheuristic in absolute terms, it is the most suitable for the specific circumstances of this thesis.

5.1.3. Limitations

Some limitations of this literature review include the restricted scope of database searches, as only two platforms were used to identify relevant literature. As a result, additional valuable research may exist but was not found due to the limited number of databases searched. Another challenge is the inconsistent terminology used in the field. Bio-inspired metaheuristics are sometimes referred to simply as metaheuristics or are mentioned by their specific names, such as GA or EC, without explicitly categorizing them as bio-inspired metaheuristic methods. This lack of standardized terminology may have led to relevant literature being overlooked if it did not include the specific keywords used in the search strings.

5.1.4. Broader Implications

The field of metaheuristics suffers from convoluted and inconsistent terminology, making it challenging to search for and classify existing literature. This lack of standardization hampers the development and accessibility of research, as relevant studies may be difficult to identify due to inconsistent categorization. Additionally, scientific research often focuses on developing new algorithms with unique names, further contributing to the terminology inconsistency and making it harder to systematically compare and evaluate different approaches within the field.

5.1.5. Future Research

To address these issues, future research should focus on structuring the theoretical framework of metaheuristics. Key areas of interest include categorizing different types of metaheuristics, defining their distinguishing characteristics, and establishing a structured classification system. A well-organized framework would improve clarity, facilitate better comparisons between algorithms, and enhance the efficiency of future research in the field.

5.2. Answer Sub-question 2

5.2.1. Summary of Key Findings

When designing a GA, several critical decisions—including mutation rate, crossover rate, ETA value, number of niches, population size, and resolution change—must be carefully tuned to optimize performance. This study explored these parameters within a MGA framework, employing the F-race method to identify configurations that yield the most effective results in exploring the solution space. The findings demonstrated that the chosen parameter settings successfully helped identify diverse capacity combinations, all within feasible regions of the solution space. Notably, however, parameter tuning results from a small-scale model may not generalize directly to larger or more complex systems, underscoring the context-dependent nature of these optimizations.

5.2.2. Answering The Sub-question

Sub-question: How can the tuning of algorithmic parameters optimize the performance of the selected heuristic for implementing a successful MGA optimization technique?

The results show that systematically tuning parameters through quantitative methods, such as F-race, can significantly enhance MGA performance by refining exploration and exploitation within the solution space. An efficient combination of mutation, crossover, and ETA values was identified for the developed test model. Further testing highlighted the delicate balance between population size, the number of populations/niches, and resolution in managing computational cost versus solution accuracy. Consequently, informed parameter tuning is instrumental in achieving a successful GA–MGA optimisation technique.

5.2.3. Limitations

Despite its contributions, this study has several limitations. First, the conclusions stem from experiments on a single system model, limiting their direct applicability to other models. Second, due to time constraints, not all tests were exhaustively or statistically validated. Although the F-race approach provides a good foundation for parameter tuning, alternative methods might prove more effective for certain types of parameters or different modeling scenarios. Moreover, the reliability of resolution adjustments remains uncertain: the possibility of algorithmic crashes and unverified behavior in other models restricts the generalizability of this feature. Lastly, while the DEAP library offers numerous operators, only a subset was explored here. Different crossover mechanisms or other bio-inspired operators might further optimize performance. For example, the uniform crossover can be replaced with the cxSimulatedBinaryBounded, which is the same crossover that the NSGA-II algorithm uses. By doing so, the algorithm will resemble the NSGA-II operators, which are already widely popular in genetic algorithms and commonly used by many other researchers.

5.2.4. Broader Implications

The promising results obtained from combining GA and MGA in an energy system context suggest that metaheuristic approaches can address some of the current challenges in optimization, including escalating computational demands and the need to handle ever more complex models. By providing diverse solution sets rather than a single “best” outcome, MGA-based methods can encourage more informed, human-centric decision-making. Beyond energy system modeling, this line of research can extend to other large-scale optimization problems where computational overhead and model complexity continue to grow. The study thereby contributes practical guidelines for parameter tuning in evolutionary algorithms, particularly in balancing solution quality with computational efficiency.

5.2.5. Future Research

To build on these findings, future work should expand testing, refine resolution adjustments, explore alternative operators, conduct comprehensive benchmarking, and adopt adaptive parameter control. Expanding testing involves applying the optimized parameter settings to different and more complex models, thereby assessing their broader effectiveness and ensuring external validity. Refining resolution adjustments requires investigating more robust resolution-changing methods and verifying their reliability and efficiency across various system models. Exploring alternative operators includes experimenting with different crossover (e.g., simulated binary bounded) and mutation operators to determine the most suitable combination for diverse problem settings. Comprehensive benchmarking, particularly through comparative studies against established methods such as NSGA-II, will help solidify the MGA framework's competitiveness and general applicability. Finally, adopting adaptive parameter control entails examining dynamic parameter tuning—adjusting rates or operator choices during runtime—which may further enhance performance while reducing manual trial-and-error. By addressing these directions, future research can deepen our understanding of how best to implement and scale MGA-based algorithms, ultimately reinforcing their potential to tackle increasingly complex optimization tasks across various domains.

5.3. Answer Sub-question 3

5.3.1. Summary of Key Findings

The research focused on evaluating and refining the MGA-GA algorithm to ensure its applicability to large-scale energy system models. Through multiple test runs, adjustments were identified to improve the algorithm's functionality and efficiency. One major challenge was the long computational time, which made the initially intended high-resolution approach infeasible. Instead, time-masking was introduced as an alternative high-resolution method, preserving some high-resolution characteristics while maintaining computational feasibility. Additionally, modifications to the slack value were implemented to prevent crashes during resolution switches, ensuring the algorithm remained stable.

To evaluate the algorithm's performance, its results were benchmarked against SPORES outputs by comparing the capacity deployment at different locations and comparing the total deployment of RES technologies such as batteries, PV and wind. The comparison revealed notable differences, particularly in the deployment of storage technologies, where the GA-MGA algorithm allocated significantly more storage than SPORES. The large model data did not have any information about capacity deployment at specific locations. It was therefor compared to a plot given in the article by Lombardi et al. [12]. When compared with this plot it was seen that the GA-MGA algorithm also deployed large quantities of capacity at certain locations, while the SPORES result diversified its capacity deployment more spatially.

To better understand the algorithm performance the algorithm process was also researched using predefined metrics. These metrics such as RES-battery capacity deployment, fitness and cost over time showed some curious behavior of the GA-MGA algorithm. While the reason for this behavior could not be explained with certainty, it showed that the generated solutions bear some qualitative uncertainty.

5.3.2. Answering The Sub-question

Sub-question: how were metrics used to evaluate the algorithm's performance, and how do these evaluations and comparisons with existing methods inform its applicability?

The algorithm's performance was evaluated by analyzing the algorithm process and comparing its results to SPORES results. The metrics used to assess the algorithm process included storage and RES capacity deployment over time, average cost over time, and average fitness over time. These metrics provided a better understanding of how the solutions were generated. While the algorithm theoretically functioned as intended, resolution changes and the overall algorithm structure introduced uncertainties in the generated solutions.

The impact of different resolution settings was also evident when comparing the algorithm's battery and RES capacity with SPORES results, as the capacities differed significantly between the two approaches. Due to the lack of resemblance between the algorithm's outputs and the SPORES results, the generated algorithm results are not of sufficient quality to be used for consideration when looking at diverse solutions for the European energy system storage and RES capacity placement.

The limitations, broader implications and future research of sub-question 3 would resemble what would be written for the research question. They are therefor discussed after the research question is answered.

5.4. Conclusion Research Question

Societal problems are becoming increasingly complex, presenting policymakers with the challenging task of finding effective solutions. One field that has grown particularly intricate is the energy transition. It now involves not only ensuring the optimal provision of energy but also addressing the mitigation of carbon emissions. As a result of this increasing complexity, the acknowledgment and use of models have grown over time. While these models can aid in optimizing system designs, their outcomes are often inaccurate due to their inability to fully capture all relevant variables and dynamics.

In response to the known shortcomings of traditional optimization modeling, alternatives like MGA have become increasingly popular. MGA provides the ability to generate distinctive, feasible solutions while still preserving the human decision-making capabilities essential for addressing complex societal problems.

MGA has already been applied to solve spatially explicit optimization problems related to energy systems. Despite its success, the iterative process of MGA can quickly become computationally burdensome when applied to larger models. To overcome such challenges, alternative algorithmic approaches could be explored. One promising option is the adoption of a bio-inspired heuristic combined with MGA, which could potentially reduce the computational time required.

The lack of literature on combining MGA and bio-inspired heuristics, particularly in the context of energy systems, has led to the formulation of the following research question: How does a combination of MGA and bio-inspired metaheuristics, aimed at optimizing energy system design, compare to existing deterministic MGA methods?

To address this research question, it was essential to first identify the type of metaheuristic best suited for the task. A review of the literature revealed that the most appropriate algorithm would be a genetic algorithm, a method within the broader field of evolutionary computing.

A small test model was created in Calliope to test the combination of the GA-MGA method. Drawing on insights from the literature review, an integrated algorithm was successfully designed and further subjected to parameter tuning to evaluate its performance under different conditions. These tests provided valuable insights into the algorithm's design and its optimal operating parameters, which would be carried forward to the next phase of development.

The next step involved incorporating the findings and refining the algorithm to ensure its applicability to a larger, more sophisticated model. For this purpose, the model developed by Lombardi et al. [12] was selected as the basis, serving as the framework within which the algorithm would need to function effectively. A few structural adjustments were made to the algorithm, after which it was prepared for benchmarking against the MGA results already generated by the larger model.

The SPORES method, an MGA approach, was used to generate multiple solutions for the large energy system optimization problem. When comparing the results of the GA to those of SPORES, it was observed that while the GA was able to identify some alternative solutions, these differed to a certain extent from the SPORES results. Something that was caused by the structural design of the algorithm, but also by how a lower resolution was used for the GA-MGA. An analysis of the algorithm process also revealed that there are still some structural inefficiencies that do influence the reliability and robustness of the solutions generated by the algorithm.

5.4.1. Answer to the Research Question

In conclusion, the combination of MGA and bio-inspired metaheuristic concepts into an algorithm was successfully achieved and applied to a spatial optimization problem in an energy system. Scaling the algorithm to a larger, more detailed model was also feasible; however, the inefficient structure of the code meant that the computational burden was a problem when applying the GA-MGA at such a large scale. Meaning that a lower, less resembling resolution needed to be used to run the algorithm.

The results comparison revealed that storage technologies were handled differently at lower resolutions, leading to more generous deployments. Wind and solar technologies showed some variations in results but demonstrated a degree of alignment with the outcomes from the SPORES method. Regarding spatial allocation, which was only partially analyzed due to limited data, SPORES favored a more distributed capacity deployment, whereas the GA relied more on large capacity deployments in specific locations. However, due to resolution limitations and structural shortcomings in the algorithm, the results lacked reliability, making a direct comparison with SPORES results uncertain.

The research further explored the combination of MGA and GA, demonstrating that an algorithm integrating these concepts can be successfully developed. While previous literature on the topic primarily discussed possible design options, this thesis fully developed and applied such an algorithm. Although the GA was not yet successful in replicating or outperforming other MGA results, it highlighted promising possibilities that warrant further investigation.

5.4.2. Limitations

One of the key limitations identified is the computational time required for the algorithm to run, which made the originally intended high-resolution approach infeasible. As a workaround, time-masking was introduced to maintain some high-resolution characteristics while keeping the computation time manageable. But as seen in the results, this difference in resolution created extremely different solutions compared to the SPORES results. Additionally, the modification of the slack value to prevent crashes during resolution switches, while effective, is not a perfect solution, as it introduces uncertainty regarding how closely the algorithm adheres to the original slack constraints.

Another limitation is the comparison with the SPORES results. Only the total capacity deployment by the large model were available. This means that a in-depth comparison between spatial capacity deployment was not possible. instead a image given in the article was given and was used to compare the data generated by two SPORES results and the GA-MGA results.

More limitations lie with how the algorithm is structured and operates. For example, the algorithm currently operates sequentially, which increases runtime significantly. Although parallelization was identified as a potential solution, its implementation was beyond the scope of this thesis due to time constraints. Furthermore, a lot of design choices have been made to solve certain problems that were encountered when developing the algorithm. Operators were tailored, a slack adjustment was introduced and a not so efficient individual initialization was created. These decisions make sure that results can be created, they also are structured that create a ”black-box” when looking at the operation of the algorithm. It is unclear to what extent they affect the operation and quality of the algorithm.

5.4.3. Future Research

The study highlights the importance of benchmarking optimization algorithms against established models like SPORES. However, discrepancies in resolution, constraints, and assumptions show that direct comparisons can be misleading if methodological differences are not carefully accounted for. This suggests that future research should develop standardized evaluation metrics for comparing alternative energy system optimization approaches.

Other research areas should focus on optimizing the algorithm process. Especially the resolution change is something that made the results less reliable. By optimizing the algorithm less computational time is required and higher resolutions can be used during the process, eventually creating more reliable and robust solutions.

To optimize the algorithm, future research should also try to enhance the structure of the developed GA-MGA. It currently has a lot of design options integrated which can easily be replaced by different design choices that are better suited for the algorithm. One important aspect that could be changed is the population initialization. The initialization could make it so that the individuals are all more scattered throughout the solution space before the algorithm process even starts.

Another highly beneficial optimization could be the parallelization of the algorithm process. Currently, each individual is solved iteratively, but multiple individuals could be processed simultaneously. If implemented, this could significantly reduce the time required to complete an algorithm run.

5.4.4. Broader Implications

The findings of this research emphasize the challenges of applying bio-inspired heuristics to large-scale energy system models. While a GA combined with a MGA method can effectively generate diverse solutions, the results indicate that algorithmic inefficiencies, resolution settings, and computational constraints can significantly impact outcomes. This underscores the need for further methodological refinement before such approaches can be widely applied in real-world energy system planning.

As with the smaller model, this part of the thesis demonstrated that the theoretical principle of GA-MGA does indeed work. While it is not yet fully optimized, this novel GA-MGA algorithm has shown potential not only for optimizing complex energy system problems but also for broader applications. More complex disciplines could potentially implement metaheuristics, such as GA, in their fields to tackle optimization problems more effectively.

Further research is needed to determine the exact benefits of integrating metaheuristics with MGA. If computational costs can be reduced through a well-developed GA-MGA algorithm, the need for high-powered computing resources could be minimized. Consequently, this could broaden access to MGA, enabling more people to use it for large-scale complex optimization problems.

The goal of MGA is to provide multiple diverse solutions for policymakers and decision-makers, giving them a broader perspective when tackling complex problems. Unlike traditional models, which typically generate a single optimal solution, MGA presents a range of viable alternatives, allowing for greater flexibility and creativity in decision-making. By doing so, MGA enhances the imaginative capacity of humans, helping policymakers explore different trade-offs and potential outcomes rather than being constrained by a single-model-driven perspective. Traditional models, while useful, are not absolute guides for policy design in complex environments, as they inherently fail to fully capture the complexity and unpredictability of reality. A more widely adoption of the MGA method could help to make better understand the problem at hand and make better suited decisions to solve them.

References

[1] E. Ruffing and V. Brendler. “The Game of Energy Transition: A Game Theoretical Perspective on Public Participation Procedures in Infrastructure Planning”. In: *European Policy Analysis* 10.1 (2023), pp. 39–60. DOI: 10.1002/epa2.1199.

[2] T. S. Nguyen, S. Mohamed, and K. Panuwatwanich. “Stakeholder Management in Complex Project: Review of Contemporary Literature”. In: *Journal of Engineering, Project, and Production Management* 8 (2018). DOI: 10.32738/JEPPM.201807.0003.

[3] B. Neumayr, M. Schrefl, and B. Thalheim. “Modeling Techniques for Multi-level Abstraction”. In: *The Evolution of Conceptual Modeling: From a Historical Perspective towards the Future of Conceptual Modeling*. Springer, 2011, pp. 68–92. DOI: 10.1007/978-3-642-17505-3_4.

[4] E. D. Brill. “The Use of Optimization Models in Public-Sector Planning”. In: *Management Science* 25.5 (1979), pp. 413–422. DOI: 10.1287/mnsc.25.5.413.

[5] Z. Yang et al. “The improved multi-criteria decision-making model for multi-objective operation in a complex reservoir system”. In: *Journal of Hydroinformatics* 21.5 (2019), pp. 851–874. DOI: 10.2166/hydro.2019.150.

[6] E. D. Brill, S.-Y. Chang, and L. D. Hopkins. “Modeling to Generate Alternatives: The HSJ Approach and an Illustration Using a Problem in Land Use Planning”. In: *Management Science* 28.3 (Mar. 1982), pp. 221–235. DOI: 10.1287/mnsc.28.3.221.

[7] J. Kao, E. D. Brill, and J. T. Pfeffer. “Generation of Alternative Optima for Nonlinear Programming Problems”. In: *Engineering Optimization* 15.3 (1990), pp. 233–251. DOI: 10.1080/03052159008941155.

[8] J. F. DeCarolis. “Using modeling to generate alternatives (MGA) to expand our thinking on energy futures”. In: *Energy Economics* 33.2 (2011), pp. 145–152. DOI: 10.1016/j.eneco.2010.05.002.

[9] L. Nolting and A. Praktiknjo. “The complexity dilemma – Insights from security of electricity supply assessments”. In: *Energy* 241 (2022), p. 122522. DOI: 10.1016/j.energy.2021.122522.

[10] J. M. Caicedo and GunJin Yun. “A novel evolutionary algorithm for identifying multiple alternative solutions in model updating”. In: *Structural Health Monitoring* 10.5 (2010), pp. 491–501. DOI: 10.1177/1475921710381775.

[11] J. Price and I. Keppo. “Modelling to generate alternatives: A technique to explore uncertainty in energy-environment-economy models”. In: *Applied Energy* 195 (2017), pp. 356–369. DOI: 10.1016/j.apenergy.2017.03.065.

[12] F. Lombardi, B. Pickering, and S. Pfenninger. “What is redundant and what is not? Computational trade-offs in modelling to generate alternatives for energy infrastructure deployment”. In: *Applied Energy* 339 (2023), p. 121002. DOI: 10.1016/j.apenergy.2023.121002.

[13] M. G. Prina et al. “Multi-objective investment optimization for energy system models in high temporal and spatial resolution”. In: *Applied Energy* 264 (2020), p. 114728. DOI: 10.1016/j.apenergy.2020.114728.

[14] J. C. Liebman. “Some Simple-Minded Observations on the Role of Optimization in Public Systems Decision-Making”. In: *Interfaces* 6.4 (1976), pp. 102–108. DOI: 10.1287/inte.6.4.102.

[15] W. L. Sprouse and G. A. Mendoza. “Modeling to Generate Alternatives: A Shawnee National Forest Example”. In: *Computers, Environment and Urban Systems* 14.3 (1990), pp. 203–211. DOI: 10.1016/0198-9715(90)90009-i.

[16] A. Goicoechea. *Multiobjective Decision Analysis with Engineering and Business Applications*. en. Nashville, TN: John Wiley and Sons, 1982.

[17] E. M. Zechman and S. R. Ranjithan. “An evolutionary algorithm to generate alternatives (EAGA) for engineering optimization problems”. In: *Engineering Optimization* 36.5 (2004), pp. 539–553. DOI: 10.1080/03052150410001704863.

[18] J. S. Yeomans. “A Simulation-Optimization Algorithm for Generating Sets of Alternatives Using Population-Based Metaheuristic Procedures”. In: *Journal of Software Engineering and Simulation* 5.2 (2019), pp. 01–06. DOI: 10.35629/9795-05020101.

[19] H. T. Sadeeq and A. M. Abdulazeez. “Metaheuristics: A Review of Algorithms”. In: *International Journal of Online Engineering* 19.9 (2023), pp. 142–164.

[20] V. Tomar, M. Bansal, and P. Singh. “Metaheuristic Algorithms for Optimization: A Brief Review”. In: *RAiSE-2023*. Basel, Switzerland: MDPI, 2024.

[21] R. Thomsen. “Multimodal Optimization Using Crowding-Based Differential Evolution”. In: *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*. CEC-04. IEEE, 2004, pp. 1382–1389. DOI: 10.1109/cec.2004.1331058.

[22] T. Stützle et al. “A Comparison of Nature Inspired Heuristics on the Traveling Salesman Problem”. In: *Parallel Problem Solving from Nature PPSN VI*. Springer Berlin Heidelberg, 2000, pp. 661–670. ISBN: 9783540453567. DOI: 10.1007/3-540-45356-3_65.

[23] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. 1st. IOP Publishing Ltd., 1997. DOI: 10.1201/9780367802486.

[24] W. Gao. “A comprehensive review on identification of the geomaterial constitutive model using the computational intelligence method”. In: *Advanced Engineering Informatics* 38 (2018), pp. 420–440. DOI: 10.1016/j.aei.2018.08.021.

[25] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford Academic, 1999. DOI: 10.1093/oso/9780195131581.001.0001.

[26] Y. Tan and Z. Zheng. “Research advance in swarm robotics”. In: *Defence Technology* 9.1 (2013), pp. 18–39. DOI: 10.1016/j.dt.2013.03.001.

[27] S. Maroufpoor, R. Azadnia, and M. Bozorg-Haddad. “Stochastic Optimization”. In: *Handbook of Probabilistic Models*. Elsevier, 2020, pp. 437–448. ISBN: 9780128165140. DOI: 10.1016/b978-0-12-816514-0.00017-5.

[28] D. V. Arnold and H. Beyer. “Random Dynamics Optimum Tracking with Evolution Strategies”. In: *Parallel Problem Solving from Nature — PPSN VII*. Springer Berlin Heidelberg, 2002, pp. 3–12. ISBN: 9783540457121. DOI: 10.1007/3-540-45712-7_1.

[29] A. Zhou et al. “Multiobjective evolutionary algorithms: A survey of the state of the art”. In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 32–49. DOI: 10.1016/j.swevo.2011.03.001.

[30] G. B. Lamont and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer eBooks, 2007. DOI: 10.1007/978-0-387-36797-2.

[31] S. Bechikh, R. Datta, and A. Gupta. *Recent Advances in Evolutionary Multi-objective Optimization*. Springer International Publishing, 2017. ISBN: 9783319429786. DOI: 10.1007/978-3-319-42978-6.

[32] X. Zhang, H. Liu, and L. Tu. “A Modified Particle Swarm Optimization for Multimodal Multi-Objective Optimization”. en. In: *Engineering Applications of Artificial Intelligence* 95.103905 (2020), p. 103905.

[33] J. Del Ser et al. “Bio-inspired computation: Where we stand and what’s next”. In: *Swarm and Evolutionary Computation* 48 (2019), pp. 220–250. DOI: 10.1016/j.swevo.2019.04.008.

[34] K. Sörensen. “Metaheuristics—the metaphor exposed”. In: *International Transactions in Operational Research* 22.1 (2013), pp. 3–18. DOI: 10.1111/itor.12001.

[35] A. Mallégol et al. “Handling Non-Linearities in Modelling the Optimal Design and Operation of a Multi-Energy System”. In: *Mathematics* 11.23 (2023), p. 4855. DOI: 10.3390/math11234855.

[36] S. Pfenninger, A. Hawkes, and J. Keirstead. “Energy systems modeling for twenty-first century energy challenges”. In: *Renewable and Sustainable Energy Reviews* 33 (2014), pp. 74–86. DOI: 10.1016/j.rser.2014.02.003.

[37] N. Spittler et al. “Understanding the Current Energy Paradigm and Energy System Models for More Sustainable Energy System Development”. en. In: *Energies* 12.8 (2019), p. 1584.

[38] F. Lombardi et al. “Policy Decision Support for Renewables Deployment through Spatially Explicit Practically Optimal Alternatives”. In: *Joule* 4.10 (2020), pp. 2185–2207. DOI: 10.1016/j.joule.2020.08.002.

[39] J. DeCarolis et al. “Modelling to generate alternatives with an energy system optimization model”. In: *Environmental Modelling and Software* 79 (2016), pp. 300–310. DOI: 10.1016/j.envsoft.2015.11.019.

[40] M. Hoffmann et al. “A review of mixed-integer linear formulations for framework-based energy system models”. In: *Advances in Applied Energy* 16 (2024), p. 100190. DOI: 10.1016/j.adapen.2024.100190.

[41] J. K. Mandal, S. Mukhopadhyay, and P. Dutta. *Multi-Objective Optimization: Evolutionary to Hybrid Framework*. Springer Singapore, 2018. DOI: 10.1007/978-981-13-1471-1.

[42] G. Jones. “Genetic and Evolutionary Algorithms”. In: *Encyclopedia of Computational Chemistry*. Chichester, UK: John Wiley and Sons, Ltd, 2002.

[43] K. Deb et al. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017.

[44] D. H. Loughlin et al. “Genetic Algorithm Approaches for Addressing Unmodeled Objectives in Optimization Problems”. In: *Engineering Optimization* 33.5 (2001), pp. 549–569. DOI: 10.1080/03052150108940933.

[45] J. Jablonský. “Benchmarks for Current Linear and Mixed Integer Optimization Solvers”. In: *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis* 63.6 (2016), pp. 1923–1928. DOI: 10.11118/actaun201563061923.

[46] M. Bröchin et al. “Harder, Better, Faster, Stronger: Understanding and Improving the Tractability of Large Energy System Models”. In: *Energy, Sustainability and Society* 14.1 (2024). DOI: 10.1186/s13705-024-00458-z.

[47] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 2)*. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2. 2024.

[48] T. Tröndle et al. “Trade-Offs between Geographic Scale, Cost, and Infrastructure Requirements for Fully Renewable Electricity in Europe”. In: *Joule* 4.9 (2020), pp. 1929–1948. DOI: 10.1016/j.joule.2020.07.018.

[49] S. Pfenninger. “Dealing with Multiple Decades of Hourly Wind and PV Time Series in Energy Models: A Comparison of Methods to Reduce Time Resolution and the Planning Implications of Inter-Annual Variability”. In: *Applied Energy* 197 (2017), pp. 1–13. DOI: 10.1016/j.apenergy.2017.03.051.

[50] F. Vafaee et al. “Balancing the Exploration and Exploitation in an Adaptive Diversity Guided Genetic Algorithm”. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. 2014, pp. 2570–2577. DOI: 10.1109/CEC.2014.6900257.

[51] S. Tsutsui and N. Fujimoto. “An Analytical Study of Parallel GA with Independent Runs on GPUs”. In: *Massively Parallel Evolutionary Computation on GPGPUs*. Springer Berlin Heidelberg, 2013, pp. 105–120. ISBN: 9783642379598. DOI: 10.1007/978-3-642-37959-8_6.

[52] C. Li et al. “Exploitation Versus Exploration”. In: *Intelligent Optimization*. Springer Nature Singapore, 2024, pp. 161–170. ISBN: 9789819732869. DOI: 10.1007/978-981-97-3286-9_7.

[53] A. E. Eiben, R. Hinterding, and Z. Michalewicz. “Parameter Control in Evolutionary Algorithms”. In: *IEEE Transactions on Evolutionary Computation* 3.2 (1999), pp. 124–141. DOI: 10.1109/4235.771166.

[54] M. Preuss, G. Rudolph, and S. Wessing. “Tuning Optimization Algorithms for Real-World Problems by Means of Surrogate Modeling”. In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. GECCO ’10. New York, NY, USA: Association for Computing Machinery, 2010, pp. 401–408. ISBN: 978-1-4503-0072-8. DOI: 10.1145/1830483.1830558.

[55] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. en. 1st ed. Natural Computing Series. Berlin, Germany: Springer, 2003.

[56] A. E. Eiben and S. K. Smit. “Parameter Tuning for Configuring and Analyzing Evolutionary Algorithms”. In: *Swarm and Evolutionary Computation* 1.1 (Mar. 2011), pp. 19–31. DOI: 10.1016/j.swevo.2011.02.001.

[57] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer Berlin Heidelberg, 2015. ISBN: 9783662448748. DOI: 10.1007/978-3-662-44874-8.

[58] E. Montero, M.-C. Riff, and B. Neveu. “A Beginner’s Guide to Tuning Methods”. In: *Applied Soft Computing* 17 (2014), pp. 39–51. DOI: 10.1016/j.asoc.2013.12.017.

[59] M. Birattari et al. “A Racing Algorithm for Configuring Metaheuristics”. In: *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*. New York City, New York: Morgan Kaufmann Publishers Inc., 2002, pp. 11–18. ISBN: 1558608788.

[60] Y. Widhiyasana et al. “Genetic Algorithm for Artificial Neural Networks in Real-Time Strategy Games”. In: *International Journal on Informatics Visualization* 6.2 (2022), p. 298. DOI: 10.30630/joiv.6.2.832.

[61] A. Hassanat et al. “Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach”. In: *Information* 10.12 (2019), p. 390. DOI: 10.3390/info10120390.

[62] F. Lombardi. *Customised Pre-Built Sector-Coupled Euro-Calliope Model - Focus on the Power Sector and Additional SPORES Options*. en. 2022. DOI: 10.5281/ZENODO.6655600.

[63] K. Deveci and Ö. Güler. “A CMOPSO-Based Multi-Objective Optimization of Renewable Energy Planning: Case of Turkey”. In: *Renewable Energy* 155 (2020), pp. 578–590. DOI: 10.1016/j.renene.2020.03.033.

[64] R. Vanitha and J. Baskaran. “A Fuzzy-Based Evolutionary Algorithm for Solving Multiobjective Optimal Power Flow with FACTS Devices”. In: *Mathematical Problems in Engineering* 2015 (2015), pp. 1–8. DOI: 10.1155/2015/275129.

[65] B. H. Choi. “Computational Approaches for Good Alternative Solutions Based Load Resistance Capacity in Design of Steel Moment-Resisting Frames”. In: *Damage Assessment of Structures VII*. Trans Tech Publications Ltd., 2007, pp. 563–568. DOI: 10.4028/0-87849-444-8.563.

[66] P. Kripakaran, B. Hall, and A. Gupta. “A Genetic Algorithm for Design of Moment-Resisting Steel Frames”. In: *Structural and Multidisciplinary Optimization* 44.4 (2011), pp. 559–574. DOI: 10.1007/s00158-011-0654-7.

[67] G. H. Huang et al. “Policy Planning Under Uncertainty: Efficient Starting Populations for Simulation-Optimization Methods Applied to Municipal Solid Waste Management”. In: *Journal of Environmental Management* 77.1 (2005), pp. 22–34. DOI: 10.1016/j.jenvman.2005.02.008.

[68] D. Fioriti et al. “Multiple Design Options for Sizing Off-Grid Microgrids: A Novel Single-Objective Approach to Support Multi-Criteria Decision Making”. In: *Sustainable Energy, Grids and Networks* 30 (2022), p. 100644. DOI: 10.1016/j.segan.2022.100644.

[69] R. Imanirad, X.-S. Yang, and J. S. Yeomans. “A Concurrent Modelling to Generate Alternatives Approach Using the Firefly Algorithm”. In: *International Journal of Decision Support System Technology* 5.2 (2013), pp. 33–45. DOI: 10.4018/jdsst.2013040103.

[70] E. M. Zechman and S. R. Ranjithan. “Multipopulation cooperative coevolutionary programming (MCCP) to enhance design innovation”. en. In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*. Washington DC, USA: ACM, 2005, pp. 1641–1648. ISBN: 978-1-59593-010-1. DOI: 10.1145/1068009.1068286.

[71] Emily M. Zechman. “Niched Co-Evolution Strategies to Address Non-uniqueness in Engineering Design”. In: (2006). URL: https://api.semanticscholar.org/CorpusID:7512895.

[72] E. M. Zechman, M. H. Giacomoni, and M. E. Shafiee. “An evolutionary algorithm approach to generate distinct sets of non-dominated solutions for wicked problems”. In: *Engineering Applications of Artificial Intelligence* 26.5 (2013), pp. 1442–1457. DOI: 10.1016/j.engappai.2013.03.004.

[73] L. Liu and S. Ranji Ranjithan. “An adaptive optimization technique for dynamic environments”. In: *Engineering Applications of Artificial Intelligence*. Advances in Metaheuristics for Hard Optimization: New Trends and Case Studies 23.5 (2010), pp. 772–779. DOI: 10.1016/j.engappai.2010.01.007.

[74] H. E. Raoui, M. Cabrera-Cuevas, and D. A. Pelta. “The Role of Metaheuristics as Solutions Generators”. In: *Symmetry* 13.11 (Oct. 2021), p. 2034. DOI: 10.3390/sym13112034.

[75] C. Feng and J. Lin. “Using a genetic algorithm to generate alternative sketch maps for urban planning”. In: *Computers, Environment and Urban Systems* 23.2 (1999), pp. 91–108. DOI: 10.1016/S0198-9715(99)00004-6.

[76] Y. Xiong and J. B. Schneider. “Transportation planning, programming, land use, and applications of geographic information systems”. en-US. In: *Transportation Research Record* 1364 (1992). ISBN: 9780309054034.

[77] S. Pfenninger and B. Pickering. “Calliope: A Multi-Scale Energy Systems Modelling Framework”. In: *Journal of Open Source Software* 3.29 (2018), p. 825. DOI: 10.21105/joss.00825.

Appendix

Appendix A

First search string:

(”bio-inspired” OR ”nature-inspired” OR ”evolutionary algorithm” OR ”genetic algorithm” OR ”swarm intelligence” OR ”particle swarm optimization” OR ”ant colony optimization” OR ”simulated annealing” OR ”genetic programming” OR ”neural networks” OR ”heuristics” OR ”meta-heuristics”)

AND

(”modeling to generate alternatives” OR MGA OR ”alternative generation” OR ”near-optimal solutions” OR ”multi-objective optimization” OR ”sensitivity analysis”)

AND

(”energy system optimization” OR ”energy system planning” OR ”power system optimization” OR ”renewable energy optimization” OR ”energy modeling optimization” OR ”energy transition optimization” OR ”renewable energy planning” OR “energy system modelling”)

Second search string:

(”bio-inspired” OR ”nature-inspired” OR ”evolutionary algorithm” OR ”genetic algorithm” OR ”swarm intelligence” OR ”particle swarm optimization” OR ”ant colony optimization” OR ”simulated annealing” OR ”genetic programming” OR ”neural networks” OR ”heuristics” OR ”metaheuristics”)

AND

(”modeling to generate alternatives”)

Appendix B

Table B.1: First Friedman Test Fitness

	Run 1	Run 2	Run 3	Run 4	Run 5
Fitness Test 1	460.69	286.93	432.20	640.15	358.23
Fitness Test 2	628.85	562.56	620.23	640.96	484.25
Fitness Test 3	628.85	562.56	620.23	640.96	484.25
Fitness Test 4	472.61	487.41	416.97	627.32	605.93
Fitness Test 5	674.71	368.76	496.36	680.29	702.19
Fitness Test 6	751.47	890.31	774.03	661.80	636.42
Fitness Test 7	407.74	366.60	306.43	163.61	531.18
Fitness Test 8	659.63	594.49	703.49	695.70	563.24
Fitness Test 9	601.54	665.50	573.49	785.39	648.37
Fitness Test 10	348.50	380.76	565.69	341.15	709.95
Fitness Test 11	615.06	748.38	723.70	755.33	607.80
Fitness Test 12	829.84	741.63	810.45	436.15	680.65
Fitness Test 13	309.44	610.03	382.36	662.34	359.77
Fitness Test 14	646.04	535.23	584.95	436.90	470.81
Fitness Test 15	723.41	456.53	846.10	675.87	748.29
Fitness Test 16	764.07	382.15	403.33	506.95	770.93
Fitness Test 17	536.71	745.66	719.06	606.98	516.62
Fitness Test 18	662.67	622.42	807.53	775.57	809.22
Fitness Test 19	459.10	352.94	521.83	463.31	644.89
Fitness Test 20	449.88	735.31	640.48	691.01	739.56
Fitness Test 21	816.51	738.19	871.69	821.84	686.77
Fitness Test 22	409.81	386.54	606.37	326.98	507.24
Fitness Test 23	613.69	436.55	558.48	608.28	650.45
Fitness Test 24	842.30	665.52	903.68	910.33	850.35
Fitness Test 25	454.88	386.49	426.04	585.09	802.11
Fitness Test 26	667.79	564.74	437.24	473.12	661.40
Fitness Test 27	746.47	714.59	830.35	887.42	656.21

Table B.2: Second Friedman Test Fitness

	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
Fitness Test 6	739.89	548.05	606.53	741.15	783.82	751.47	890.31	774.03	661.80	636.42
Fitness Test 12	613.18	767.19	739.82	797.24	731.30	829.84	741.63	810.45	436.15	680.65
Fitness Test 15	774.09	620.60	644.07	857.51	727.68	723.41	456.53	846.10	675.87	748.29
Fitness Test 18	681.79	759.92	616.17	897.01	758.31	662.67	622.42	807.53	775.57	809.22
Fitness Test 21	658.26	793.13	686.35	951.46	826.26	816.51	738.19	871.69	821.84	686.77
Fitness Test 24	720.52	801.86	718.70	859.57	748.11	842.30	665.52	903.68	910.33	850.35
Fitness Test 27	811.45	706.63	909.40	747.26	679.27	746.47	714.59	830.35	887.42	656.21

Table B.3: Third Friedman Test Fitness

	Fit 6	Fit 12	Fit 15	Fit 18	Fit 21	Fit 24	Fit 27
Run 1	739.89	613.18	774.09	681.79	658.26	720.52	811.45
Run 2	548.05	767.19	620.60	759.92	793.13	801.86	706.63
Run 3	606.53	739.82	644.07	616.17	686.35	718.70	909.40
Run 4	741.15	797.24	857.51	897.01	951.46	859.57	747.26
Run 5	783.82	731.30	727.68	758.31	826.26	748.11	679.27
Run 6	751.47	829.84	723.41	662.67	816.51	842.30	746.47
Run 7	890.31	741.63	456.53	622.42	738.19	665.52	714.59
Run 8	774.03	810.45	846.10	807.53	871.69	903.68	830.35
Run 9	661.80	436.15	675.87	775.57	821.84	910.33	887.42
Run 10	636.42	680.65	748.29	809.22	686.77	850.35	656.21
Run 11	660.96	645.01	698.62	754.22	827.11	688.38	799.11
Run 12	764.10	799.61	672.38	668.78	738.14	827.19	843.03
Run 13	851.54	722.84	645.10	697.34	814.20	582.45	859.56
Run 14	662.14	745.92	794.83	698.43	854.79	853.16	874.47
Run 15	706.26	569.07	844.30	812.71	777.55	712.20	669.64

Table B.4: First Friedman Test Generation

	Run 1	Run 2	Run 3	Run 4	Run 5
Gen Test 1	93	95	91	85	100
Gen Test 2	97	54	96	100	100
Gen Test 3	97	54	96	100	100
Gen Test 4	97	94	96	97	97
Gen Test 5	98	81	60	88	100
Gen Test 6	95	88	91	97	97
Gen Test 7	97	84	62	93	77
Gen Test 8	100	93	97	100	93
Gen Test 9	97	86	100	97	94
Gen Test 10	80	100	81	100	88
Gen Test 11	41	97	91	81	88
Gen Test 12	74	83	88	92	78
Gen Test 13	98	100	99	96	97
Gen Test 14	76	70	82	100	98
Gen Test 15	96	59	87	88	95
Gen Test 16	100	95	99	83	100
Gen Test 17	82	90	100	100	97
Gen Test 18	97	69	98	100	97
Gen Test 19	96	93	91	100	88
Gen Test 20	99	100	100	96	97
Gen Test 21	99	76	99	94	73
Gen Test 22	95	91	100	100	97
Gen Test 23	99	96	80	86	99
Gen Test 24	46	83	95	75	96
Gen Test 25	86	96	96	100	82
Gen Test 26	99	98	97	93	95
Gen Test 27	96	91	94	100	85

Table B.5: Second Friedman Test Generation

	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
Gen Test 6	97	100	87	89	68	95	88	91	97	97
Gen Test 12	100	89	95	88	95	74	83	88	92	78
Gen Test 15	93	86	92	83	84	96	59	87	88	95
Gen Test 18	93	97	89	81	76	97	69	98	100	97
Gen Test 21	93	79	96	86	96	99	76	99	94	73
Gen Test 24	92	81	68	95	100	46	83	95	75	96
Gen Test 25	79	91	95	100	100	86	96	96	100	82
Gen Test 27	99	92	80	80	92	96	91	94	100	85

Table B.6: Third Friedman Test Generation

	Gen 6	Gen 12	Gen 15	Gen 18	Gen 21	Gen 24	Gen 27
Run 1	76	92	92	97	84	95	94
Run 2	87	97	69	97	96	73	92
Run 3	99	95	94	85	96	87	55
Run 4	91	95	94	99	100	91	95
Run 5	78	93	90	83	94	81	94
Run 6	97	100	93	93	93	92	99
Run 7	100	89	86	97	79	81	92
Run 8	87	95	92	89	96	68	80
Run 9	89	88	83	81	86	95	80
Run 10	68	95	84	76	96	100	92
Run 11	95	74	96	97	99	46	96
Run 12	88	83	59	69	76	83	91
Run 13	91	88	87	98	99	95	94
Run 14	97	92	88	100	94	75	100
Run 15	97	78	95	97	73	96	85

Appendix C

Table C.1: Number of Populations Changed with Fitness and Time Measured

Fitness p3	Time p3	Fitness p4	Time p4	Fitness p5	Time p5
753.16	3968.91	751.61	5478.28	699.86	6551.62
721.23	3990.21	734.43	5456.86	643.32	6501.29
718.90	3980.31	660.84	5732.87	518.17	6340.10
824.51	3916.57	790.99	5536.29	707.64	6119.97
889.89	4066.14	497.78	5928.56	723.43	5983.57
921.11	4002.55	706.39	5946.22	791.49	6460.68
918.21	3922.68	720.83	6070.26	732.18	6430.08
731.74	3483.21	769.09	5970.59	553.82	6275.93
790.09	4115.34	791.08	5917.65	665.35	5889.99
748.54	4041.16	717.97	5914.03	697.10	5841.68
729.50	4206.35	735.72	5968.24	690.71	7528.51
872.22	3774.95	880.69	4581.17	761.02	5977.44
685.25	3635.11	704.35	4574.69	835.01	6000.52
767.57	3457.68	642.32	4653.19	507.48	6111.53
820.70	3738.50	617.09	5242.14	668.34	6453.61
778.15	3670.98	743.56	5246.33	721.16	6418.65
550.32	3854.25	660.53	5471.09	729.14	7119.84
874.64	3917.58	810.08	5229.35	705.01	6554.87
796.89	3942.57	779.07	5241.09	771.86	6497.72
760.26	3931.56	761.62	5163.19	624.50	6362.36
851.85	3830.21	644.47	4999.00	698.85	6113.66
816.96	4021.01	792.99	4929.43	617.21	5987.13
718.60	3971.05	588.12	4678.93	678.73	6012.80
812.73	3956.38	736.21	4486.82	638.92	5822.18
684.94	3924.05	635.17	4532.82	671.51	5677.03
739.14	4365.34	505.25	4501.37	664.77	5653.13
732.45	4322.24	696.80	5964.46	679.04	6697.49
845.68	4290.76	712.52	5045.75	696.43	6772.48
Sum fitness	**Sum**	**Sum fitness**	**Sum**	**Sum fitness**	**Sum**
780.54	3939.20	706.70	5302.17	681.86	6291.28

Table C.2: Percental Difference Between Number of Subpopulations

Subpopulation Comparison	% Difference Fitness	% Difference Time
p3 - p4	-9.5	34.6
p4 - p5	-3.5	18.7
p3 - p5	-13.0	59.7

Appendix D

Table D.1: Results of Population Size Test (s10_p3 & s10_p4)

s10_p3 Fitness	Generation	Time	s10_p4 Fitness	Generation	Time
753.16	88	3968.91	751.61	88	5478.28
721.23	100	3990.21	734.43	95	5456.86
718.90	92	3980.31	660.84	88	5732.87
824.51	81	3916.57	790.99	93	5536.29
889.89	94	4066.14	497.78	85	5928.56
921.11	84	4002.55	706.39	99	5946.22
918.21	88	3922.68	720.83	98	6070.26
731.74	83	3483.21	769.09	97	5970.59
790.09	86	4115.34	717.97	98	5914.03
748.54	86	4041.16	735.72	85	5968.24
729.50	97	4206.35	767.33	83	6346.92
872.22	95	3774.95	880.69	97	4581.17
767.57	98	3457.68	704.35	86	4574.69
820.70	97	3738.50	642.32	99	4653.19
778.15	74	3670.98	617.09	87	5242.14
550.32	95	3854.25	743.56	86	5246.33
874.64	98	3917.58	660.53	98	5471.09
796.89	95	3942.57	810.08	94	5229.35
760.26	97	3931.56	779.07	81	5241.09
851.85	97	3830.21	761.62	98	5163.19
816.96	84	4021.01	644.47	98	4999.00
718.60	97	3971.05	588.12	98	4678.93
812.73	97	3956.38	635.17	95	4532.82

Table D.2: Results of Population Size Test (s15_p3 & s15_p4)

s15_p3 Fitness	Generation	Time	s15_p4 Fitness	Generation	Time
736.31	96	98.00	738.89	83	7740.85
800.31	95	96.91	827.55	89	7950.91
861.32	91	102.09	708.49	97	8504.69
941.37	92	7066.14	666.83	79	9350.70
837.35	77	6511.16	824.07	82	7585.77
774.09	95	6281.73	786.18	98	7416.75
650.19	97	6264.96	850.57	100	7378.65
805.13	94	6231.93	660.42	76	8289.68
790.58	93	5690.57	844.35	100	8312.40
837.68	97	5538.10	710.57	99	8122.83
766.57	100	5426.88	845.88	99	7434.34
818.46	97	5421.46	796.65	81	7269.05
934.60	83	5351.27	734.02	95	7147.66
812.97	74	5956.82	667.76	82	7285.59
941.41	91	6211.89	765.40	87	7961.92
863.16	87	5950.98	785.83	96	8269.43
798.24	97	6218.82	794.60	96	8366.56
671.33	96	6240.22	609.54	98	8517.07
770.33	94	6562.74	615.57	93	8671.87
688.98	90	6647.82	617.09	71	8759.09
812.92	95	6554.38	884.22	100	7708.10
808.54	90	5507.61	783.56	92	7318.52
808.20	99	5487.10	812.26	98	7432.50

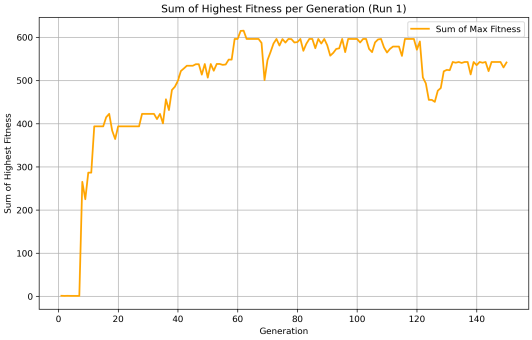
Table D.3: Results of Population Size Test (s20_p3 & s20_p4)

s20_p3 Fitness	Generation	Time	s20_p4 Fitness	Generation	Time
990.32	82	7744.76	630.16	97	9978.21
915.32	83	7953.12	835.45	100	10314.45
824.48	96	8166.71	778.58	85	10223.53
887.61	98	9588.59	798.73	100	12087.66
970.09	84	8402.36	822.28	99	9924.13
950.51	68	8349.78	726.98	97	9923.12
1012.95	67	7543.49	740.69	99	10688.84
943.15	76	7476.32	769.50	100	10723.94
881.46	96	7269.47	775.35	100	9690.58
786.16	81	7210.31	807.48	74	10138.31
847.18	83	7303.88	668.45	100	10675.30
921.84	77	7832.94	844.74	86	10947.28
797.86	96	8140.03	733.80	84	10784.95
865.11	81	8348.78	750.86	99	10870.30
671.56	91	7889.24	780.52	79	11329.98
871.43	98	8310.32	818.14	81	10782.91
922.65	80	8732.57	807.75	89	12281.42
776.50	94	8769.08	753.35	98	14347.00
755.05	95	7544.00	900.49	99	11750.98
794.30	78	7560.88	742.46	89	11637.55
840.93	100	7587.62	821.83	74	12289.33
755.96	94	7614.73	711.74	99	11584.95
903.65	99	7316.05	618.99	96	11196.26

Table D.4: Results of Population Size Test (s25_p3 & s25_p4)

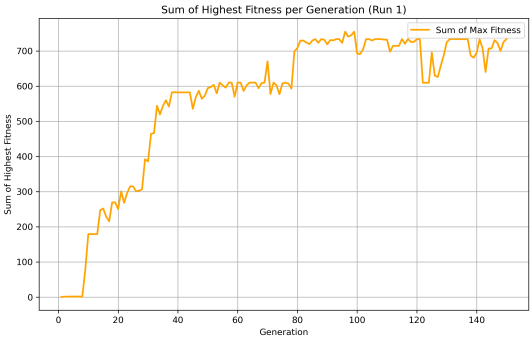
s25_p3 Fitness	Generation	Time	s25_p4 Fitness	Generation	Time
873.76	75	9594.54	861.84	99	13060.44
816.57	97	10073.90	729.26	83	13940.37
979.81	84	10597.29	780.90	99	13568.76
909.68	87	11823.26	871.02	93	14836.68
938.08	81	10505.40	719.74	77	12388.61
902.23	85	9685.77	729.50	82	12924.47
887.17	57	9736.50	669.60	91	13259.15
870.81	97	8956.24	822.08	81	13341.16
907.91	82	9271.70	842.19	100	13676.42
880.82	83	9047.56	779.51	98	14853.95
914.05	84	9096.42	750.51	88	14340.73
798.83	95	10226.32	797.71	83	12840.75
840.80	93	10486.80	748.77	95	12695.96
909.94	89	10416.87	601.51	80	12749.68
865.66	97	9249.11	835.31	83	12341.02
992.90	85	8725.11	842.02	82	12496.07
993.97	85	8463.67	903.78	100	11896.25
822.73	82	8452.00	780.80	97	15192.44
890.50	96	9186.61	678.20	83	14096.40
775.25	95	9184.21	889.21	72	14134.18
810.03	91	9151.69	791.99	100	14703.29
848.39	70	9463.80	846.59	81	14005.68
798.89	75	8899.49	759.10	77	12898.60

Appendix E

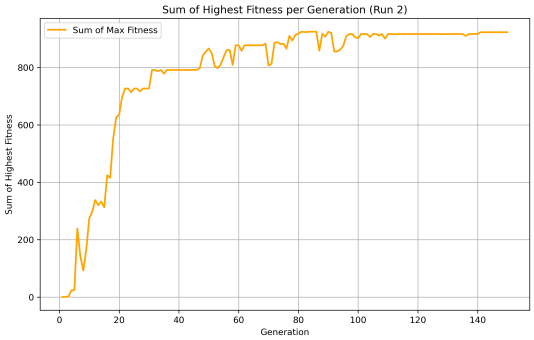


(a) Run 1

(b) Run 2

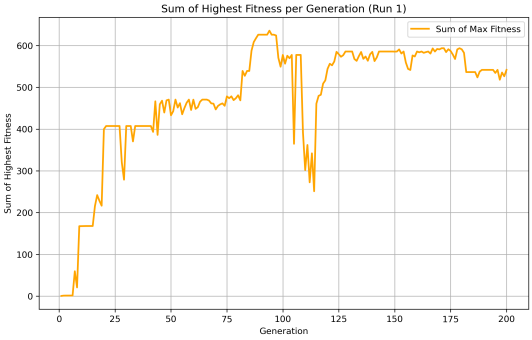


(c) Run 3

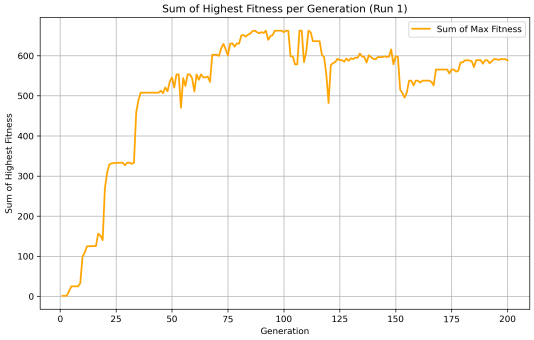


(d) Run 4

Figure E.1: multiple runs with high resolution and 150 generations



(a) Run 1



(b) Run 2

Figure E.2: Two runs with high resolution and 200 generations

Appendix F

F.1. SLURM file before results

```
#!/bin/sh
#
#SBATCH --job-name="Large model run"
#SBATCH --partition=memory
#SBATCH --time=09:00:00
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=16
#SBATCH --mem-per-cpu=8G
#SBATCH --account=education-tpm-msc-cosem
#SBATCH --output=Largemodelrun.%j.out

module load gurobi/10.0.3

srun ../python/large_model_run.py
```

F.2. SLURM file after results

```
#!/bin/sh
#
#SBATCH --job-name="Large model run"
#SBATCH --partition=memory
#SBATCH --time=24:00:00
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=5
#SBATCH --mem-per-cpu=30G
#SBATCH --account=education-tpm-msc-cosem
#SBATCH --output=Largemodelrun2.%j.out

module load  gurobi/10.0.3

srun ../python/large_model_run_2.py
```

Appendix G

Listing G.1: Data extraction and cleaning code

```python
import pandas as pd

# Load the dataset (update the file path as necessary)
input_file = 'individual_generation_interval.xlsx'  # Replace with the actual filename
output_file = 'output_with_sums.xlsx'

# Read all sheets from the Excel file
sheets = pd.read_excel(input_file, sheet_name=None)

# Define the technology categories
technology_categories = {
    'wind_offshore': ['wind_offshore'],
    'wind_onshore': ['wind_onshore_competing', 'wind_onshore_monopoly'],
    'pv': ['open_field_pv', 'roof_mounted_pv'],
    'battery': ['battery'],
    'hydrogen_electricity_storage': ['hydrogen_electricity_storage']
}

# Create a new Excel writer
with pd.ExcelWriter(output_file, engine='openpyxl') as writer:
    for sheet_name, data in sheets.items():
        # Create sum columns for each technology category
        for category, keywords in technology_categories.items():
            technology_columns = [col for col in data.columns if any(keyword in col for
                keyword in keywords)]
            data[f'sum_{category}'] = data[technology_columns].sum(axis=1) * 100

        # Keep only the required columns and the new sum columns
        required_columns = ['generation', 'subpopulation', 'individual', 'fitness', 'cost']
        sum_columns = [f'sum_{category}' for category in technology_categories.keys()]
        data = data[required_columns + sum_columns]

        # Write the updated sheet to the output file
        data.to_excel(writer, sheet_name=sheet_name, index=False)
```

Appendix H

Category	Total Wind	Total PV	Total Storage
Niche 1	448.60	398.45	387.04
Niche 2	434.19	498.64	373.66
Niche 3	479.52	325.52	354.87
SPORES Evolve 1	782.72	615.29	129.25
SPORES Evolve 2	626.71	83.46	34.34
SPORES Evolve 3	658.16	108.76	45.98
SPORES Integer 1	796.99	620.07	128.41
SPORES Integer 2	702.15	196.57	58.34
SPORES Integer 3	752.31	219.91	49.93
SPORES Random 1	790.91	636.51	162.61
SPORES Random 2	787.24	572.57	94.70
SPORES Random 3	770.83	412.27	97.00
SPORES Relative 1	790.57	639.27	133.84
SPORES Relative 2	627.60	517.38	98.08
SPORES Relative 3	660.19	470.35	68.07
Optimum	731.77	477.41	70.38

Table H.1: Comparison of Niche, SPORES, and Optimum Solutions in Energy Deployment

Comparison	Diff Wind %	Diff PV %	Diff Storage %	Euclidean Distance
Niche 1 vs Evolve 1	74.48	54.42	-66.61	474.46
Niche 1 vs Evolve 2	39.70	-79.05	-91.13	569.72
Niche 1 vs Evolve 3	46.71	-72.71	-88.12	417.51
Niche 2 vs Evolve 1	80.27	23.39	-65.41	441.38
Niche 2 vs Evolve 2	44.34	-83.26	-90.81	427.79
Niche 2 vs Evolve 3	51.58	-78.19	-87.70	668.67
Niche 3 vs Evolve 1	63.23	89.02	-63.58	476.24
Niche 3 vs Evolve 2	30.70	-74.36	-90.32	633.18
Niche 3 vs Evolve 3	37.25	-66.59	-87.04	528.23
Total Diff Niche 1 - Evolve	53.63	-32.45	-81.95	487.23
Total Diff Niche 2 - Evolve	58.73	-46.02	-81.30	512.61
Total Diff Niche 3 - Evolve	43.73	-17.31	-80.32	545.88

Table H.2: Pairwise Percentage Differences and Euclidean Distance between Niches and SPORES Evolve

Comparison	Diff Wind %	Diff PV %	Diff Storage %	Euclidean Distance
Niche 1 vs Integer 1	77.66	55.62	-66.82	487.21
Niche 1 vs Integer 2	56.52	-50.67	-84.93	512.32
Niche 1 vs Integer 3	67.70	-44.81	-87.10	422.57
Niche 2 vs Integer 1	83.56	24.35	-65.63	454.44
Niche 2 vs Integer 2	61.72	-60.58	-84.39	392.58
Niche 2 vs Integer 3	73.27	-55.90	-86.64	785.39
Niche 3 vs Integer 1	66.21	90.49	-63.81	488.70
Niche 3 vs Integer 2	46.43	-39.61	-83.56	731.47
Niche 3 vs Integer 3	56.89	-32.44	-85.93	410.22
Total Diff Niche 1 - Integer	67.29	-13.28	-79.62	474.03
Total Diff Niche 2 - Integer	72.85	-30.71	-78.89	544.14
Total Diff Niche 3 - Integer	56.51	6.14	-77.77	543.47

Table H.3: Pairwise Percentage Differences and Euclidean Distance between Niches and Integer Solutions

Comparison	Diff Wind %	Diff PV %	Diff Storage %	Euclidean Distance
Niche 1 vs Random 1	76.31	59.75	-57.99	473.52
Niche 1 vs Random 2	75.49	43.70	-75.53	455.99
Niche 1 vs Random 3	71.83	3.47	-74.94	398.61
Niche 2 vs Random 1	82.16	27.65	-56.48	436.81
Niche 2 vs Random 2	81.31	14.83	-74.66	472.67
Niche 2 vs Random 3	77.54	-17.32	-74.04	879.52
Niche 3 vs Random 1	64.94	95.54	-54.18	480.25
Niche 3 vs Random 2	64.17	75.90	-73.31	978.04
Niche 3 vs Random 3	60.75	26.65	-72.67	234.50
Total Diff Niche 1 - Random	74.54	35.64	-69.49	442.71
Total Diff Niche 2 - Random	80.34	8.38	-68.39	596.33
Total Diff Niche 3 - Random	63.29	66.03	-66.72	564.26

Table H.4: Pairwise Percentage Differences and Euclidean Distance between Niches and Random Solutions