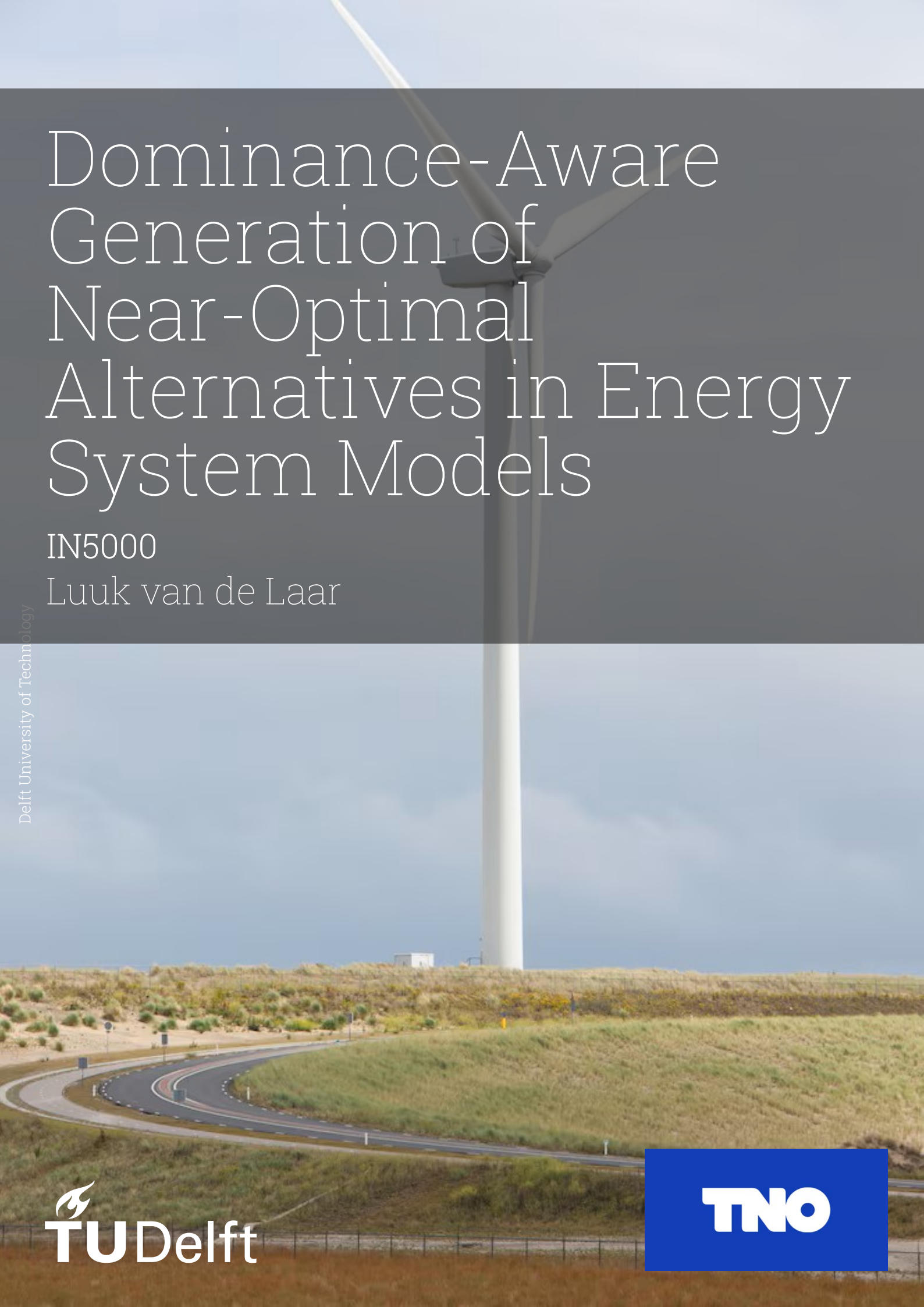
Dominance-Aware Generation of Near-Optimal Alternatives in Energy System Models

IN5000

Luuk van de Laar

Delft University of Technology



TU Delft

TNO

Dominance-Aware Generation of Near-Optimal Alternatives in Energy System Models

by

Luuk van de Laar

Responsible professor:	M. de Weerdt
Supervisor TNO:	G. Morales
Co-supervisor:	G. Neustroev
Project Duration:	December, 2024 - September, 2025
Faculty:	Faculty of Computer Science, Delft

Abstract

Optimization models are widely used in energy system planning to identify cost-effective investment strategies. However, relying solely on a single optimal solution can be misleading, as it fails to account for model uncertainty, competing objectives, and stakeholder preferences. To address this, near-optimal alternatives, solutions that are close in cost to the optimum but structurally different, are increasingly used to support robust and flexible decision-making.

This thesis explores the generation and evaluation of near-optimal alternatives within energy systems, with a focus on improving the decision relevance of the generated alternatives. This thesis introduces a unified analytical framework, formalizing existing Modeling to Generate Alternatives (MGA) methods using weight vector formulations. This formulation enables a clearer comparison of different techniques that generate these alternatives. This analysis highlights the limitations of current evaluation metrics, particularly their inability to distinguish decision-relevant alternatives from decision-irrelevant ones.

To overcome this gap, the thesis proposes a novel evaluation metric based on dominance relations from multi-objective optimization. This metric identifies non-dominated alternatives, those not strictly worse than any other across all decision variables, as decision-relevant. The thesis introduces a new method that uses Directionally Weighted Variables to generate alternatives aligned with this dominance criterion.

The proposed approach is evaluated using a stylized energy investment model and benchmarked against existing MGA techniques. Results show that traditional methods tend to generate fewer non-dominated alternatives, while the new method generates more non-dominated alternatives within the near-optimal space. This work contributes a new perspective on alternative generation, bridging the gap between mathematical optimality and practical decision support.

Contents

1

Introduction

Energy systems planning plays a critical role in shaping the transition toward reliable, affordable, and sustainable energy futures. As societies face rising energy demand, decarbonization targets, and increasing integration of renewable resources, planning decisions must balance technical, economic, and environmental considerations. The International Energy Agency emphasizes that such planning guides decisions on when, where, and how to invest in the energy sector within broader policy frameworks [15].

Optimization models are widely applied to support complex decisions in energy system planning. These models often aim to find a single cost-optimal solution, under the assumption that minimizing economic cost leads to the most desirable outcome. However, this focus on optimality can be too narrow. In real-world contexts, where multiple objectives, stakeholder preferences, and model uncertainties must be considered, a single optimal solution offers limited value. It may obscure trade-offs, fail to account for model misspecifications, and not take into account the robustness of decisions [21, 31].

To address this limitation, there is growing interest in generating near-optimal alternatives, solutions that lie within a defined tolerance of the optimal objective value but differ structurally in their decision variables. These alternatives help reveal the flexibility of the solution space, expose hidden trade-offs, and enable decision-makers to consider a broader set of options. Near-optimal alternatives are particularly useful in energy transition modeling, where competing investment priorities and political constraints must be balanced [9, 18].

Several methods have been proposed to generate such alternatives, most notably Modeling to Generate Alternatives (MGA) techniques, including deterministic methods such as Hop-Skip-Jump (HSJ) [5] or stochastic methods such as Min/Max Variables [33]. While effective in simple settings, these methods face limitations: they often do not explore the near-optimal solution space evenly, may produce redundant alternatives, and can be computationally intensive for large-scale problems. Moreover, current evaluation metrics fail to capture the decision relevance of alternatives, favoring alternatives that either cover the near-optimal solution space or are robust, but not necessarily meaningful.

This thesis makes four key contributions. First, it provides a unified analytical framework for analyzing existing MGA methods by expressing them in terms of weight vector formulations, allowing for clearer comparison and exposing implicit assumptions. Second, the thesis compares current MGA methods by using the metrics used in literature and showcases their shortcomings. Based on these shortcomings, a new metric called dominance is introduced, which is based on dominance in multi-objective optimization problems. Alternatives are considered non-dominated if there is no other alternative that is strictly better across all variables. Third, it introduces a novel algorithm that uses Directionally Weighted Variables to explicitly guide the search of the near-optimal solution space to non-dominated alternatives, inspired by multi-objective optimization. Lastly, this new algorithm is validated based on experiments done on the Generation Expansion Planning (GEP) problem. For which the results show that the new algorithm produces a large share of non-dominated alternatives, while remaining computationally feasible.

1.1. Research Question

This thesis addresses the following research questions:

RQ 1: *What are the conceptual differences between known MGA approaches for generating near-optimal alternatives?*

RQ 2: *How can these differences be exploited to reformulate the alternative generation problem in a way that supports decision relevance?*

RQ 3: *How does the proposed dominance-based method perform compared to existing MGA techniques in terms of decision relevance and computational efficiency?*

1.2. Outline

Chapter 2 introduces the core concepts of near-optimal alternatives and establishes the theoretical foundation for their use in modeling, with a focus on modeling in energy models. It presents the formulation of the original optimization model and explains how a near-optimal solution space can be defined through the use of slack variables. The chapter then surveys a broad range of existing techniques for generating alternatives and discusses their computational characteristics and limitations.

In chapter 3, the focus shifts to the challenges of evaluating near-optimal alternatives. While various methods exist to generate alternatives, assessing their usefulness and quality is non-trivial. This chapter explores the key difficulties in comparing alternatives and introduces existing evaluation metrics, such as convex hull coverage and shadow pricing, which are commonly used in the literature.

Chapter 4 builds on comparing the existing alternative generation methods by critically analyzing and evaluating them. It argues that current evaluation metrics, while informative, fail to capture whether alternatives are genuinely relevant for decision-making. To address this, the chapter introduces a new perspective grounded in multi-objective optimization theory, proposing that each decision variable can be interpreted as an implicit objective. This leads to a new evaluation framework based on dominance relations, where non-dominated alternatives are prioritized.

Chapter 5 formalizes a dominance-based approach and presents a new metric for evaluating near-optimal alternatives. This metric identifies alternatives that are not strictly worse across all variables compared to others in the set, providing a more decision-relevant basis for evaluation.

Chapter 6 proposes a novel method for generating alternatives using Directionally Weighted Variables. This method is designed to align with the dominance metric introduced in chapter 5, guiding the search process toward structurally non-dominated alternatives within the near-optimal region.

Chapter 7 describes the experimental setup used to evaluate the performance of the proposed method. It outlines the model formulation, solver configuration, and algorithmic workflow, including the procedures used to benchmark existing and new techniques.

The results of these experiments are presented in chapter 8. Here, traditional MGA methods are compared against the proposed dominance-aligned approach across decision relevance and computational efficiency. The analysis demonstrates that, while conventional techniques often produce redundant and dominated alternatives, the new method yields a more representative and meaningful set of alternatives.

Finally, chapter 9 discusses the broader implications of the findings, reflects on the limitations of the study, and outlines promising directions for future research. It concludes by revisiting the research questions and summarizing the main contributions of the thesis.

2

Modeling of Near-Optimal Alternatives

The transition toward sustainable energy systems requires decision-support tools. Energy system models have become central to this effort, helping stakeholders evaluate trade-offs that align with technical, economic, and policy goals. These models typically optimize for a well-defined objective, such as minimizing cost. However, relying solely on an optimal single-objective solution can be misleading or insufficient, especially when dealing with real-world uncertainties, competing objectives, or the need for stakeholder engagement.

This chapter introduces the concept and methodology of generating near-optimal alternatives, which provide decision-makers with a broader and more informative set of options than a single optimal solution. The alternatives generated through these methods are referred to as sets of alternatives. It begins by describing the formulation of the original energy system models and the definition of the near-optimal solution space. A simple example is provided to illustrate these concepts visually. The chapter then surveys a variety of techniques used to generate these sets of alternatives, first, it discusses Modeling to Generate Alternatives (MGA) methods such as Hop-Skip-Jump [9], Spores [22], Min/Max Variables [23, 33], Random Vector [3] and Maximizing Distance Metrics [28]. Finally, it explores newer developments, including Modeling for All Alternatives [26], Parallel Optimization [12], and Multi-Objective Optimization [16].

2.1. Defining and Exploring the Near-Optimal Solution Space

Solving complex energy network models is increasingly used to support decision-making in the energy transition. These models minimize economic costs, which can be insufficient, as reported by real-world stakeholders [31] and literature surveys [21, 29]. In such cases, near-optimal alternatives of the optimal solution are considered. These alternatives can better support decisions by offering vastly different alternatives, or—when the parameters of the original model are not fully known—give alternatives that could be optimal given small changes in the model, improving its robustness. To achieve this, methods that generate near-optimal alternatives are used to find sets of alternatives that are—in some way—as different as possible from the optimal solution while being bounded by the near-optimal solution space.

2.1.1. Motivation

Classic energy network optimization models typically minimize a single (or multiple) objective function(s), often expressed as a summation of investment variables. E.g., the sum of all investment variables that minimizes the total economic cost.

The key insight motivating the exploration of near-optimal alternatives is to view these classic models not as definitive representations of a “true” objective, but rather as synthetic representations of different, potentially competing investment decisions. E.g., instead of viewing the sum of all investments as the “true” objective. Each investment variable can be interpreted as reflecting its own implicit objective,

shaped by distinct political, institutional, or social constraints. This insight allows for the exploration of fundamentally different variable assignments that still yield low objective function values.

The process begins from a single reference solution, either the globally optimal solution or a well-representative point from the Pareto front when considering Multi-Objective Optimization [7]. This optimal solution represents the minimal cost of the model. Because the different investment variables can be seen as competing investment decisions, the optimal solution is used as an indicator for the near-optimal solution space. Based on this reference solution, it systematically generates alternatives that deviate structurally from this baseline, but stay within a predefined tolerance of optimality.

An example of this process is given here: If the cost-optimal solution recommends investing €10 million in renewable energy and €10 million in fossil fuels (minimizing the total sum), a viable near-optimal alternative might involve €15 million in renewable energy and €6 million in fossil fuels (politically more attractive). Although the overall cost increases slightly, the alternative presents a significantly different investment configuration, thus offering meaningful insight into trade-offs.

Crucially, this process aims to generate sets of diverse alternatives, not merely isolated alternatives. Each alternative is—in some way—constructed to be as different as possible from the others while maintaining acceptable objective performance. This set of diverse alternatives enhances understanding of the decision landscape, particularly for decision-makers who must account for policy, uncertainty, or qualitative preferences alongside strict cost minimization. In this way, the method supports a richer, more informed basis for strategic investment planning [9].

2.1.2. Energy Systems

Models of energy systems follow the following structure:

$$\begin{aligned} \text{Minimize } & f(\mathbf{x}) \\ \text{subject to } & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \geq 0. \end{aligned} \tag{2.1}$$

The model minimizes its linear objective functions f. $\mathbf{x}$ is a vector containing all variables x_i, and all variables are constrained to be greater than or equal to 0. $\mathbf{A}$ and $\mathbf{b}$, a matrix and a vector respectively, are the coefficients used to specify all the constraints, where the inequality $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ is understood element-wise. This model is assumed to be a Linear Program (LP) problem. The objective function can be described as

$$f(\mathbf{x}) = \sum_i c_i x_i, \tag{2.2}$$

where c_i indicates the coefficient of variable x_i in the objective function. It is assumed that the optimal solution of this model is known. A critical reader might wonder why there are no equality constraints. These constraints, however, can be modeled by clever use of the inequality constraints, where each equality is split up into two inequalities so that the upper and lower bounds are the same [4].

2.1.3. Near-Optimal Solution Space

The solution space is the space defined by the constraints of the original model [9]. When determining the near-optimal solution space, a slack variable s is used. This near-optimal solution space is defined in

$$\begin{aligned} \text{subject to } & f(\mathbf{x}) \leq (1+s)T \\ & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \geq 0. \end{aligned} \tag{2.3}$$

Here s indicates the slack for the objective function and T indicates the original objective value of the objective function. A, b, and x are the same as in the original model. As the original model is an LP problem, finding alternatives within this newly created near-optimal solution space is also an LP problem. A lot of research has been done to find alternatives in this near-optimal solution space [3, 9, 12, 16, 22, 23, 26, 28, 33].

It is important to emphasize that this thesis focuses exclusively on single-objective optimization problems. However, the concept of exploring near-optimal alternatives can also be extended to models with multiple objectives. In a multi-objective setting, each objective function can be indexed by an index j, and a near-optimal solution space would be defined by applying a separate slack condition to each objective. This means that for each objective, a new near-optimal constraint is added to the model instead of a singular one as is done with a single-objective optimization problem. Specifically, if T_j denotes the optimal value of objective f_j, the near-optimal constraints are described:

$$f_j(\mathbf{x}) \leq (1 + s_j)T_j \quad \forall j, \tag{2.4}$$

where s_j represents the permissible slack for each respective objective. While the near-optimal solution space is defined differently when multiple objectives are considered, the methods that generate the alternatives can be applied to this new near-optimal solution space without needing adjustments. While this generalization is well-established in the literature, it lies outside the scope of this work, which is limited to the single-objective case for conceptual clarity.

2.1.4. Example

Given the definitions of the original model and near-optimal solution space, let us consider a simple visual example. First, an original model needs to be defined. The problem that this chapter considers tries to minimize x and y, in more technical terms, $f(\mathbf{x}) = x_1 + x_2$. The variables x_1 and x_2 are bound by 0 subject to some other constraints, which are specified in (2.5). The solution space encapsulated by these constraints is shown in Figure 2.1. When solved, the optimal solution will have an objective value of $f(\mathbf{x}) = 0.5$ with $x_1 = 0.25$ and $x_2 = 0.25$.

$$\begin{aligned} \text{Minimize } & f(\mathbf{x}) \\ \text{subject to } & x_1 + 3x_2 \geq 1 \\ & 3x_1 + x_2 \geq 1 \\ & \mathbf{x} \geq 0 \end{aligned} \tag{2.5}$$

Based on the original model and its optimal value, this section defines the near-optimal with a typical slack from literature, 10% [32]. In this case, s takes on a value of 0.1. The near-optimal solution space consists of the original solution space bounded by the $f(\mathbf{x}) \leq (1+s)T$. This example problem considers the objective $f(\mathbf{x})$ and thus this new bound is defined as $x_1 + x_2 \leq (1+0.1) \cdot 0.5$, as shown in (2.6). The near-optimal solution space encapsulated by these constraints is shown in Figure 2.2. The feasible region is colored in blue.

$$\begin{aligned} \text{subject to } & f(\mathbf{x}) \leq (1 + s)T \\ & x_1 + 3x_2 \geq 1 \\ & 3x_1 + x_2 \geq 1 \\ & \mathbf{x} \geq 0 \end{aligned} \tag{2.6}$$

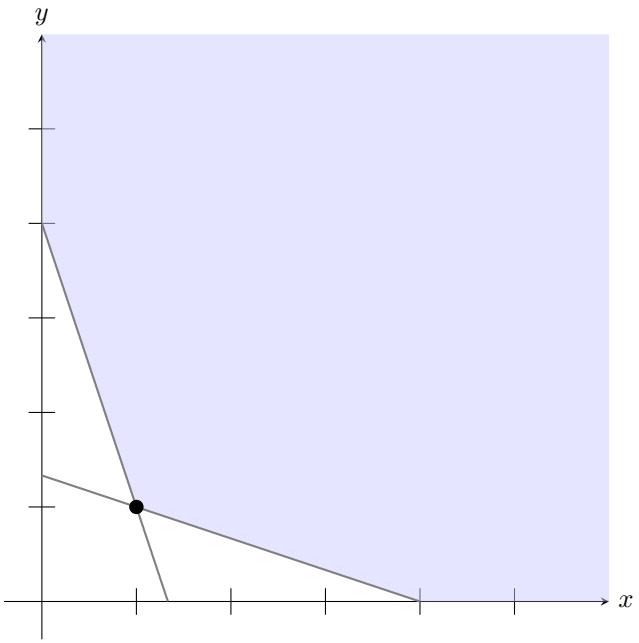


Figure 2.1: Feasible region and optimal point of the simple example problem

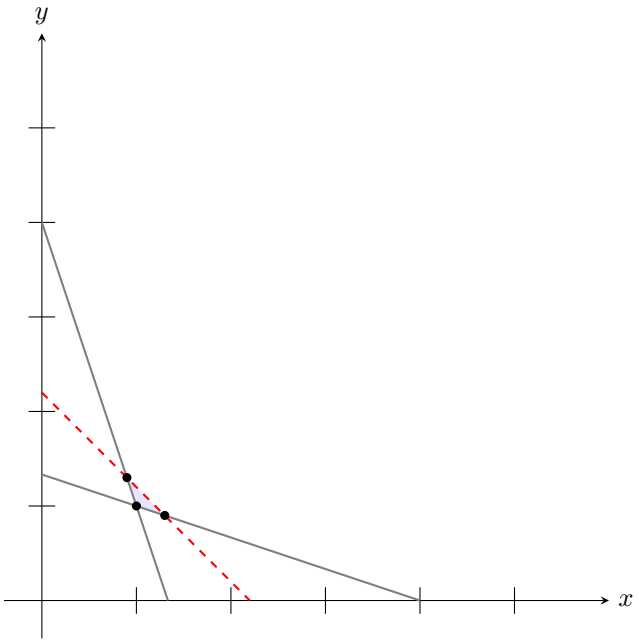


Figure 2.2: Near-optimal solution space, 10% slack is applied to the objective.

2.2. Modeling for Generating Alternatives

Modeling for Generating Alternatives, also known as MGA, was first introduced in 1982 by Brill Jr, Chang, and Hopkins [5]. DeCarolis [9] introduced MGA to energy network models, where it sees the most use [18].

MGA works on a one-at-a-time approach, where n alternatives are generated sequentially. Each alternative $k, 1 \leq k \leq n$ minimizes the sum of the weight vector $\mathbf{w^k}$—which indicates the search direction in the near-optimal solution space—and the decision variable vector $\mathbf{x}$, as shown in (2.7). This is subject to the near-optimal solution space defined in (2.3). In this section, various methods with different weight vectors are discussed. It is essential to note that this one-at-a-time approach is effective if the model is simple; however, as the model's size increases, generating even a single alternative becomes computationally expensive.

$$\text{Minimize } \mathbf{w^k} \cdot \mathbf{x} \tag{2.7}$$

Weight vectors are not explicitly formulated in most MGA techniques, and to my knowledge, only Spores [22] formally defines and uses a weight vector. I have observed that many of the known MGA techniques implicitly use a weight vector to steer the alternative generation in a particular direction. In this thesis, I formalize this interpretation by expressing each method in terms of an equivalent weight vector. This unifying framework helps compare different MGA techniques more clearly and exposes underlying assumptions that might otherwise go unnoticed. In this section, the algorithms are described and discussed with a generalized weight vector.

A key feature of the MGA framework is its flexibility: with appropriate domain knowledge, the modeler can intentionally modify the update rule for a selected subset of decision variables. This ability to embed expert insight directly into the alternative-generation process allows for more targeted exploration of the near-optimal alternatives. This section presents the general update rules used by each method; however, these can be adapted or overridden to reflect context-specific priorities or constraints.

There are three different deterministic heuristics namely Hop-Skip-Jump [9], Spores [22], and Max-Distance [28]. There are also considers two different stochastic heuristics, namely Min/Max Variables [33], and Random Vector [3].

2.2.1. Hop-Skip-Jump

Hop-Skip-Jump (HSJ) [5] is a method that tries to find alternatives by minimizing the variables that had a non-zero value in the previous iteration. This is done to find alternatives that invest less in the decision variable, which was already invested in in the previous iteration. More precisely, it minimizes (2.7) based on the value of the variables in the previous iteration. The weights are defined in

$$w_i^k = \begin{cases} 0 & \text{if } x_i^{k-1} = 0 \\ 1 & \text{otherwise.} \end{cases} \tag{2.8}$$

For example, if the variable values of the previous iteration included $x_1^{k-1} = 0 \wedge x_2^{k-1} = 1$ then of those 2 variables $w_1^k = 0$, while $w_2^k = 1$. When using these weights, note that x_i^0 is defined based on the value of variable x_i in the original model.

The advantage of this approach is that the resulting problem is relatively simple. First, solving for the alternatives is computationally as expensive as solving for the original model. Lastly, HSJ explores the region of the near-optimal solution space that had zero investment in the previously found alternative, guiding the search to generate as different as possible alternatives.

There are also some disadvantages. The method does not guarantee that the solution space is spanned evenly and every region of the near-optimal solution space is explored [10, 24]. Not spanning the space evenly can be detrimental when searching for sets that best represent the whole near-optimal solution space. Another disadvantage is that the method can fail to find additional meaningful alternatives if each variable in the previous alternative has a non-zero value, and the sum of those values is already the minimum possible sum. In that case, the generated alternative is the same as the previously generated alternative.

2.2.2. Spores

Spores [22] is a modification of HSJ. It does not discard the weights of the previous iteration. Instead, it updates them using the update rule as described in

$$\begin{aligned} w_i^k &= w_i^{k-1} + \frac{x_i^{k-1}}{x_i^{max}} \quad \forall k, 1 \leq k \leq n, \\ w_i^0 &= 0. \end{aligned} \tag{2.9}$$

This equation introduces x_i^{max}, which refers to the maximum possible value that x_i can take. This update rule assigns a high weight to variables whose value is close to the maximum value, while assigning a low weight to variables whose value is close to zero. This is similar to the HSJ method, but with the ability to be more expressive. This weight is the basis of the Spores method.

Spores has the same advantage as HSJ, where finding alternatives is as expensive as solving the original model. Spores also improves on HSJ in one major way. It does not discard previously found alternatives by updating its weight and thus guiding the search in the less explored regions and improving the spanned space.

However, similar to HSJ, Spores still has no guarantee to evenly span the near-optimal solution space, and the method can get stuck if the function for updating weights does not generate new points, which can happen if in between iterations the relative weight of each variable stays the same (so that only the length of the weight vector changes, but not its direction). To the extent of my knowledge, no linear weight function exists that prevents this. Spores also has some new disadvantages. First, each of the variables requires a carefully chosen upper bound, which other MGA methods do not need. Otherwise, the update rule cannot be used efficiently. Lastly, in contrast to HSJ, Spores can need multiple solves to update the weights, as a small change in the weight vector can lead to finding no new alternatives; in this case, multiple iterations are needed to find a new alternative.

Spores [22] also reports a new idea to tackle the problem, where there is no guarantee to explore the solution space evenly. They addressed this problem by introducing a technique to guarantee the exploration of “important” variables. Domain knowledge is needed to indicate which variables are important. Instead of minimizing as considered in (2.7), this approach introduces an arbitrary number of new objective functions. These objective functions all take into account a specific “important” variable x_i, which is minimized together with the original objective function, see (2.10). This has the advantage that every “important” variable will be explored when minimizing. However, this specific use case of Spores is outside the scope of this thesis and is not considered further.

$$\text{Minimize } x_i + \mathbf{w^k} \cdot \mathbf{x} \tag{2.10}$$

2.2.3. Min/Max Variables

Min/Max Variables [33, 23] is an approach that minimizes and/or maximizes a random sample of variables. This is done by randomly sampling the weights as specified in

$$w_i \sim \text{Uniform}(\{-1, 0, 1\}) \, \forall i. \tag{2.11}$$

Contrary to HSJ and Spores, the Min/Max Variables does not consider any previously found alternatives. When applying this approach, all variables with weight 1 get minimized, the variable with weight -1 gets maximized, and the variables with weight 0 are free.

Using Min/Max Variables has some advantages. First, Min/Max Variables has the added benefit of randomly sampling the space, thus allowing for the use of known sampling methods to get a better idea of the true near-optimal solution space. Second, Min/Max Variables has the added benefit that, given enough iterations, each possible unique weight vector can be considered. Thus, it will find all alternatives that can be found by using the specific weight set of $\{-1, 0, 1\}$. In practice, there are too many different weight vectors to consider, so this means that each iteration has a distinct weight vector.

However, it still has some of the same disadvantages. The space spanned by this method is not guaranteed to be spanned evenly. It is also not very expressive in the possible weights by which the variables are explored, -1 or 1, thus potentially leaving some dimensions unexplored.

2.2.4. Random Vector

Random Vector [3] is similar to Min/Max Variables. Where Min/Max Variables samples either -1, 0, or 1, Random Vector samples a predefined distribution for all variables, most of the time this distribution is the uniform distribution between -1 and 1, as described in

$$w_i \sim \text{Uniform}(-1, 1) \; \forall i. \tag{2.12}$$

However, the distribution can be different for each variable to best fit the model being solved.

This method has mostly the same advantages and disadvantages as Min/Max Variables. The biggest difference is that the weight vector used in Random Vector is more expressive and can more easily explore parts of the near-optimal solution space that are not found by Min/Max Variables. This, however, comes at the cost that there are an infinite number of possible random weight vectors, which leads to the curse of dimensionality when exploring high-dimensional models [1].

2.2.5. Max-Distance Metrics

Max-Distance Metrics are different from previous approaches that scale the variables by weight. Instead of minimizing the dot product between the weight w and the variables x, it maximizes the distance between the variables x and the sum of all previous alternatives (2.13). Although there are different distance metrics, not much research regarding different distance metrics has been conducted within the literature. To the extent of my knowledge, only Price and Keppo [28] used the L1 norm (also known as Manhattan or city-block distance), which results in a linear objective function. Price and Keppo also considered other distances (such as the Euclidean distance), which changes the new objective function to a quadratic programming problem. This was quickly discarded because solving these problems is computationally expensive.

$$\text{Maximize} \sum_{i=0}^{k-1} Distance(\mathbf{x^k}, \mathbf{x^i}) \tag{2.13}$$

2.3. Modeling for All Alternatives

Modeling for All Alternatives (MAA) tries to find all alternatives within the near-optimal solution space [26] as its convex hull. MAA improves on the one-at-a-time approach of generating alternatives that each MGA method employs by finding all possible alternatives in the near-optimal solution space. The method works by finding all the points on the edge of the feasible near-optimal solution space. This can be found by solving (2.7). Similarly to MGA, where weight vectors guide the generation of alternatives in different directions of the solution space, MAA also employs a weight vector $\mathbf{w^k}$ to explore the boundaries of the near-optimal region. In the context of MAA, $\mathbf{w^k}$ is a unit vector that specifies the search direction to locate points on the convex hull of the feasible space. While MGA typically varies the weight vector iteratively to find diverse alternatives one at a time, MAA uses the geometry of the solution space, specifically the face-normal vectors of the convex hull, to systematically guide $\mathbf{w^k}$ toward unexplored boundary regions. The method works by first finding the optimal solution of the problem and additionally d arbitrary points on the edge of the near-optimal solution space, where d is the problem's dimensionality, such that a convex hull can be formed. Of this hull, the face-normal vectors of the half-space are calculated. The vector on the biggest face of the convex hull is used as the weight vector in the new iteration. This is all subject to the near-optimal solution space as described in (2.3). This process is iterated until the volume of the convex hull converges. That ensures that all points of the convex hull will eventually be found and thus all possible alternatives have been generated.

In practice, the volume of this space is calculated using the quickhull algorithm [2]. When this volume converges to a value and does not increase anymore, the algorithm stops. Using the alternatives found, the polyhedron of the near-optimal solution space can be sampled to generate a representative dataset of the near-optimal solution space. This sampling is done by drawing simplexes using the same quickhull algorithm. These simplexes are a triangle in 2D, a tetrahedron in 3D, etc. They are then sampled based on the fraction of the volume of the simplex and the number of alternatives required, which makes it possible to sample the space evenly.

Using this approach has the advantage of guaranteeing that all possible alternatives are considered within the solution space. The biggest disadvantage is that this approach does not scale very well into higher dimensions. Calculating the convex hull of spaces in more than 10 dimensions is computationally expensive. Known methods that calculate the convex hull either fail or do not finish in a reasonable amount of time.

2.4. Co-Evolution

Another method to improve the one-at-a-time approach used in standard MGA is finding multiple alternatives in parallel [12]. In this approach, evolutionary algorithms co-evolve to find the optimal value of the problem. The main idea of co-evolution is to partition the alternatives into P different sets consisting of K alternatives. Where each partition $p \in P$ has a different set of K alternatives. Each set P has a target value as to how much the alternative in that set may differ from the original objective function. In other words, each set of alternatives has a target slack to which it will evolve. It is discussed that it is possible to use multiple metaheuristics to guide the evolution. To the extent of my knowledge, only the firefly algorithm has been used as a metaheuristic for this method of generating alternatives [12, 14]. Using the firefly algorithm improves the runtime of finding alternatives. However, the co-evolution of multiple evolutionary algorithms increases the computational burden of the model.

2.5. Generating Alternatives for Multi-Objective Problems

The methods for finding alternatives are not limited to singular alternatives. Methods that generate alternative Pareto fronts are also considered within the field [16]. These new fronts are generated not by introducing a new objective function, but by constraining different variables to take on values that are different than the original solution, while still adhering to the near-optimal solution space. This constraint takes on two distinct approaches. The first approach to finding an alternative Pareto front is to completely exclude certain variables from being invested in. In other words, constraining those variables to be zero in the alternatives. For example, if the original model invests in fossil fuels, an alternative Pareto front can be generated by excluding any investments in fossil fuels altogether. The second is to restrict the capital of certain variables. This introduces a new constraint that limits the

amount invested in those variables. Continuing on the previous example, instead of excluding investments in fossil fuels, a maximum investment cap is used. In both cases, these new constraints, together with the near-optimal constraint, are added to the original model. This updated model is solved, and a new Pareto front is found.

2.6. Summary

In this chapter, we explored the importance and methodology of generating near-optimal alternatives in energy system modeling. While traditional optimization focuses on identifying a single optimal solution, this approach can be too narrow for real-world decision-making, where uncertainty, conflicting objectives, and stakeholder preferences must be considered. Near-optimal alternatives expand the solution space, offering decision-makers a more nuanced and resilient foundation for strategic planning.

We introduced the formal structure of the original linear programming model and how a near-optimal solution space is defined through the introduction of slack variables. A simple illustrative example helped visualize the concepts, showing how additional alternatives can exist close to the optimal one but differ significantly in structure.

A range of methodologies for generating alternatives was then reviewed. Classic MGA techniques such as Hop-Skip-Jump and Spores offer intuitive and computationally manageable ways to explore the solution space, though they have limitations in coverage. Randomized methods like Min/Max Variables and Random Vector sampling improve coverage and allow for domain-informed exploration, while distance-based metrics introduce a focus on maximizing dissimilarity. Modeling for All Alternatives attempts to systematically capture the full set of feasible alternatives, though at a higher computational cost. Parallel Optimization tries to improve the runtime by increasing the computational cost. Lastly, generating alternatives for Multi-Objective Optimization extends the usage of MGA to apply to multi-objective models.

Each method presents trade-offs between the generated alternatives. No single approach is universally superior; the best choice depends on the problem context, model complexity, and the decision-making environment. Collectively, these techniques empower modelers to provide richer and more actionable insights, ultimately supporting more robust and informed decisions in the transition to sustainable energy systems.

3

Challenges and Metrics of Evaluating Near-Optimal Alternatives

This chapter examines the challenges of evaluating and comparing alternatives produced by near-optimal alternative generation methods, with a focus on their quality and usefulness for decision-making. Current practice often relies on domain experts to interpret and select alternatives, which can be time-consuming and subjective. Ideally, decision-makers (such as policymakers) should be supported by systems that do not only generate alternatives but also prioritize them in a way that highlights the most meaningful, diverse, and robust options. For this to happen, we need reliable and interpretable metrics that can help evaluate the alternatives, both individually and as sets.

To achieve reliable and interpretable metrics, this chapter first discusses the key challenges that should be considered when comparing different alternatives. Based on these requirements, the chapter lists a short survey of existing metrics from the literature used to evaluate either individual alternatives or sets of alternatives.

3.1. Challenges

Methods that generate near-optimal alternatives typically produce sets of alternatives, rather than single alternatives. These sets form the basis for exploring the coverage of the near-optimal solution space and robustness of alternatives within the near-optimal space. However, several key challenges arise in generating, evaluating, and comparing these sets effectively.

3.1.1. Ranking of Alternatives

Each set of alternatives may contain multiple feasible alternatives that differ in structure or composition. A central challenge is to prioritize and rank these alternatives according to how well they align with the decision-making criteria of policymakers. Effective ranking reduces the time and cognitive effort required for decision-making by highlighting the most relevant options first. It is inefficient, for example, to present a clearly inferior alternative ahead of a demonstrably superior one.

3.1.2. Comparison between Sets

In addition to evaluating individual alternatives, it is often necessary to compare entire sets of alternatives generated by different methods or under varying assumptions. A critical aspect of this comparison is how well each set covers the near-optimal solution space. Sets that cover a broader range of distinct and meaningful alternatives are generally more valuable than sets composed of highly similar alternatives. For instance, a set where all alternatives disproportionately reduce investment in one technology at the expense of another, without exploring other viable trade-offs, may fail to provide useful insights. A set of alternatives that only considers the trade-offs between fossil fuels and renewables, without exploring the trade-offs that nuclear energy brings to the table, fails to provide insights into the option of nuclear energy planning.

3.1.3. Computational Efficiency

Generating diverse, high-quality sets of near-optimal alternatives, especially for large multi-objective LP problems, can be computationally expensive to solve. Efficient algorithms that can handle the dimensionality of energy systems models while producing meaningful alternatives in acceptable time frames are essential for widespread application. For example, in a national energy system optimization model with thousands of variables and constraints, solving for a single optimal solution may already take several hours. Generating a representative set of diverse near-optimal alternatives can multiply this cost substantially. Without efficient algorithms, the computational burden becomes prohibitive for real-time policy support or iterative stakeholder engagement.

3.2. Comparison Metrics

Alternative generation methods serve different purposes, ranging from supporting policymaking decisions to identifying optimal alternatives under uncertainty or model misspecification. Consequently, various comparison approaches have emerged. Despite their importance, most of the literature does not employ formal metrics to evaluate alternative sets. Instead, domain experts typically assess alternatives qualitatively and draw conclusions based on their assessment.

Next to the analysis of domain experts, two distinct metrics are commonly used to evaluate the outputs of different alternative-generating methods: convex hull [18] and shadow pricing [29]. The convex hull metric evaluates the overall set of alternatives produced by a method, assessing their collective coverage. In contrast, shadow pricing evaluates individual alternatives, focusing on the robustness of each alternative. Both approaches offer valuable insights and are in detail discussed in this section.

3.2.1. Convex hull

Comparing convex hulls [18] is a metric introduced to compare the output of two methods that generate near-optimal alternatives, by comparing the two different output sets. The convex hull is calculated by taking all the proposed alternatives created by a method and comparing the volume of the convex hull of these proposed alternatives. A method that produces a larger convex hull is considered to have found a more diverse set of alternatives, as there are more possible trade-offs considered between the two sets. This is shown by Pedersen et al. [26], where many alternatives are generated within the convex hull. So the larger the convex hull, the more alternatives one can consider.

For example, consider a two-dimensional decision problem with objectives like cost and environmental impact. If a method generates five alternatives that are tightly clustered in one corner of the objective space, while a different method produces five alternatives spread out more broadly across the space, the convex hull of the alternatives of the latter method encloses a larger area when compared to the former. This allows the latter method to give more diverse trade-offs and flexibility when it comes to decision-making.

The metric has some limitations. To generate a convex hull, there have to be at least $d+1$ representative alternatives, where d is the number of dimensions of the problem. An alternative is representative when it is not a convex combination of other alternatives. When there are not $d + 1$ representative alternatives, the volume of the high-dimensional solution space will be zero, and the metric gives no information about the method. Lastly, calculating the convex hull for high dimensions is already a computationally expensive problem, so this method does not lend itself to quick comparisons between different methods.

3.2.2. Shadow Pricing

In contrast to comparing convex hulls, shadow pricing [29] is a metric that compares individual alternatives. Shadow pricing does not compare the method that produces the alternatives. These shadow prices compare the price deficit, the difference in the price calculated by the model and the actual price, of different alternatives that can occur when the model is slightly off. The core idea is to prefer the alternatives that stay feasible, even when there are slight perturbations in the model. This is done by calculating the dual vector of each alternative based on the original model. This dual vector is element-wise minimized; this is called the shadow price of the alternative. The alternative that has the lowest shadow price is the preferred alternative. In other words, given some small changes in the real world regarding the constraints, shadow pricing estimates the impact of these changes on each alternative.

For example, decision variable x was modeled with the constraint $x \cdot a \leq 1000$, where a in this example represents the price of oil, modeled based on observations in the real world. For this example, assume a is set to 10. The method would prefer an alternative with $x = 90$ over an alternative with $x = 95$. This preference is guided by the shadow price associated with each constraint. The shadow price can be interpreted as a lower bound on the objective function's sensitivity to that constraint, essentially quantifying how tight or "active" the constraint is. A lower shadow price implies more slack in the constraint, indicating that the alternative is less sensitive to perturbations in the parameter a, and thus more robust.

A key limitation of shadow pricing is that it evaluates constraint violations relative to the original optimization model. By design, the optimal solution to this model will have a high shadow price, since the shadow price reflects the marginal value of relaxing constraints. As the most "optimal" solution tends to push constraints to their limits. This leads to the optimal solution often appearing the least robust under this metric, even though it performs best in terms of the original objective.

This highlights an important aspect of shadow pricing: it does not reward alternatives that strictly optimize performance, but rather those that are resilient to small changes in constraints. In this sense, the metric is more about identifying robustness than optimality. As the authors have emphasized, shadow pricing is best suited for filtering through a set of already high-quality alternatives to identify those that maintain performance under perturbation.

3.3. Summary

This chapter has explored the key challenges and existing metrics associated with evaluating near-optimal alternatives in decision-making contexts. While the generation of alternatives has received increasing attention, comparatively little focus has been placed on how to assess the quality and utility of these alternatives, either individually or as sets.

Three major challenges were identified:

- Ranking of Alternatives, which deals with prioritizing alternatives in a way that aligns with decision-making criteria.
- Comparison between Sets, which emphasizes the need to assess the coverage and representativeness of sets of alternatives.
- Computational Efficiency, which highlights the need for scalable algorithms that can support timely and iterative decision processes, especially in large-scale models.

To address these challenges, two prominent comparison metrics were reviewed:

- Convex Hull measures the coverage of a set of alternatives by evaluating the volume of the solution space they span. A larger convex hull implies a broader range of meaningful trade-offs possible within the alternatives. However, the method struggles with high-dimensional problems and requires a sufficient number of convex independent alternatives to produce informative results.
- Shadow Pricing focuses on the robustness of individual alternatives by analyzing their sensitivity to small perturbations in model parameters. Alternatives with a lower shadow price are considered more robust. This leads to the metric inherently penalizing cost-efficient alternatives over robust ones.

Together, these metrics highlight different dimensions of quality in near-optimal alternatives. In the next chapter, these metrics are analyzed, and shortcomings based on the decision relevance of the alternatives will be discussed.

4

Analyzing the Metrics that Compare Near-Optimal Alternatives

The previous chapter evaluated a range of methods using established metrics like convex hull area and shadow pricing. These metrics offer valuable insights but tend to emphasize either coverage of the full near-optimal solution space or the robustness of the alternatives themselves. This chapter analyzes current metrics and establishes a disconnect between the theoretical foundation of the metrics and the quality of the alternatives in decision-making contexts.

The lack of explicit relation of the theory to the quality of the alternatives motivates a more principled approach. This chapter analyses the known metrics and Modeling for Generating Alternatives (MGA) methods used in the literature and identifies their shortcomings based on decision relevance. To combat these shortcomings, this thesis introduces a novel concept of evaluating near-optimal alternatives with concepts from multi-objective optimization theory, specifically the Pareto front. Rather than treating each decision variable of the alternatives as a decision parameter, they are instead treated as their implicit objective function. Where the alternatives can be structured by a dominance relation based on trade-offs between the decision variables. This new perspective enables a deeper understanding of what makes one alternative more relevant than any other.

4.1. Evaluating Existing Metrics and Methods for Near-Optimal Alternatives

To understand how near-optimal alternatives can be meaningfully evaluated and compared, we conduct a dual analysis, where the idea is to evaluate existing metrics used to compare the different MGA methods, while also comparing MGA techniques used to generate those alternatives. This two-sided approach is necessary because the effectiveness of a metric is only meaningful in the context of the alternatives it evaluates, and the usefulness of a method depends not just on how many alternatives it finds, but on the kind of alternatives it generates.

To understand the strengths and limitations of existing metrics used to evaluate near-optimal alternatives, this section compares all MGA techniques discussed in section 2.2 using a controlled, illustrative problem. This comparison focuses exclusively on methods that share a common structural foundation: they all generate alternatives by iteratively modifying the objective function between runs. This shared framework allows for consistent comparison of how each technique explores the near-optimal solution space.

Although Modeling for All Alternatives (MAA) discussed in section 2.3, also relies on iteratively updating the objective function, it is excluded from this analysis for two key reasons. First, while MAA is well-suited to small models, it becomes computationally infeasible in higher dimensions, often failing beyond 10 decision variables. Second, MAA does not require the user to specify the number of alternatives in advance; instead, it exhaustively enumerates all corner points of the near-optimal region, producing

a set of alternatives that includes the whole near-optimal solution space. This makes it fundamentally incompatible with other MGA methods, which allow for only a predetermined number of alternatives. As such, while MAA is a useful tool for exhaustive problem characterization, its unique assumptions and scalability issues make it unsuitable for side-by-side comparison with more general-purpose MGA methods. Altogether, this pushes the analysis of MAA beyond the scope of this thesis.

4.1.1. Two-dimensional illustrative problem

To analyze the behaviour of the different methods when generating alternatives, a two-dimensional illustrative problem is created. This problem is created to show the differences between each method. The optimization problem used is defined in (4.1), with constraints that define a convex feasible region. The goal is to minimize a simple linear cost function, while exploring how different methods traverse the space of near-optimal solutions.

$$\begin{aligned} \text{Minimize } & x + 3y \\ \text{subject to } & 0 \leq x \leq 10 \\ & 0 \leq y \leq 30 \\ & 6x + y - 13 \geq 0 \\ & x + y - 3.5 \geq 0 \\ & 0.4x + y - 2 \geq 0 \\ & 3.5x + y - 8.5 \geq 0 \\ & -3x + y + 15.9 \geq 0 \\ & -3x + y + 2 \geq 0 \end{aligned} \tag{4.1}$$

Its optimal solution is the point $x = 5, y = 0$, with optimal solution value 5. Together with a slack of 1.3, this gives the following near-optimal solution space, see (4.2). This space is bounded and characterized by key corner points, labeled $\mathbf{a_1}$ to $\mathbf{a_8}$ in Figure 4.1. These points serve as a reference for comparing how well each method covers the decision-relevant space. The optimal solution to this model is the solution $\mathbf{a_3}$, which does not have to be generated as an alternative and is only shown when generated as an alternative.

$$\begin{aligned} \text{subject to } & 0 \leq x \leq 10 \\ & 0 \leq y \leq 30 \\ & 6x + y - 13 \geq 0 \\ & x + y - 3.5 \geq 0 \\ & 0.4x + y - 2 \geq 0 \\ & 3.5x + y - 8.5 \geq 0 \\ & -3x + y + 15.9 \geq 0 \\ & -3x + y + 2 \geq 0 \\ & x + 3y \leq (1 + 1.3) \cdot 5 \end{aligned} \tag{4.2}$$

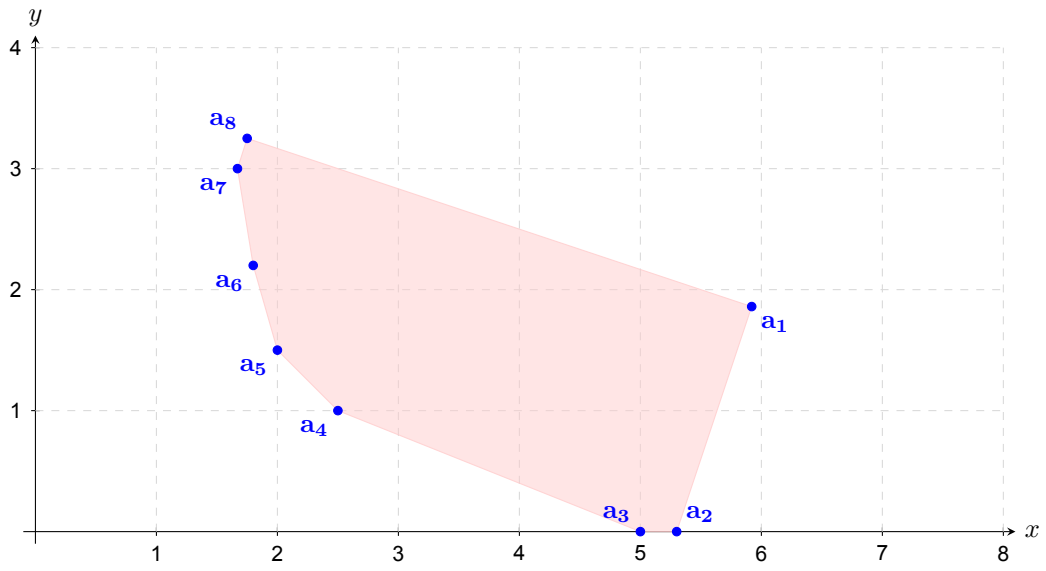


Figure 4.1: The corner points and convex hull of the near-optimal solution space.

It is important to note that the Min/Max Variables and Random Vector methods incorporate stochastic elements in their generation processes. As a result, repeated runs can yield slightly different sets of alternatives and, consequently, variations in the volume of the resulting convex hull. While this variability exists, the examples presented here are representative of the typical behavior observed across multiple runs. The patterns and insights drawn from these observations remain consistent and support the broader conclusions of this analysis.

All five MGA techniques were applied to the problem to generate both a set of alternatives with 5 and 20 iterations, respectively. With only 5 iterations, the results show which regions the methods explore when searching alternatives. While the set of 20 alternatives is chosen to show which parts of the near-optimal solution space each method does not explore, even if the number of alternatives allows for the total search space to be fully explored with an ideal method. The generated alternatives in the set of alternatives for a set size of 5 are visualized in Figure 4.2 to Figure 4.6. Table 4.1 shows how many times each alternative was generated. The resulting set of alternatives with a set size of 20 and their convex hulls are visualized in Figure 4.7 to Figure 4.11. These plots also reveal how different techniques explore the near-optimal solution space or where they fail to do so. The Table 4.2 shows how many times each method found each alternative, and Table 4.3 shows how long each method took to find these alternatives when using the hardware as discussed in Appendix A.

Table 4.1: Number of times each reference point was generated by each method when generating 5 alternatives.

Method	$\mathbf{a_1}$	$\mathbf{a_2}$	$\mathbf{a_3}$	$\mathbf{a_4}$	$\mathbf{a_5}$	$\mathbf{a_6}$	$\mathbf{a_7}$	$\mathbf{a_8}$	set size
Max-Distances	1	1	0	0	0	0	0	3	3
HSJ	0	0	0	4	0	0	1	0	2
Spores	0	0	0	0	1	2	2	0	3
Min/Max Variables	3	0	0	2	0	0	0	0	2
Random Vector	0	2	0	0	0	0	2	1	3

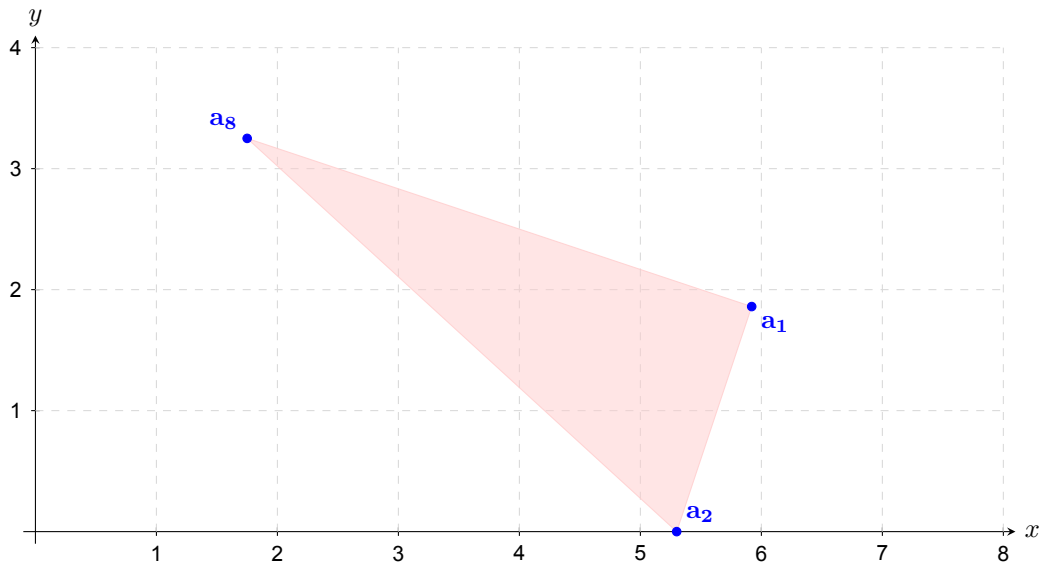


Figure 4.2: The corner points and convex hull found by using the Max-Distance method when generating 5 alternatives.

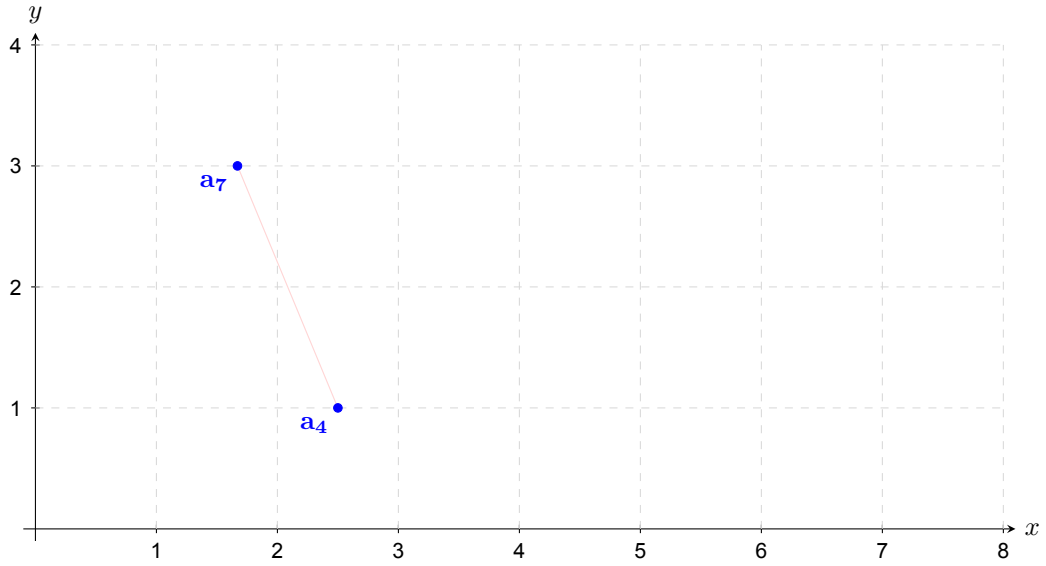


Figure 4.3: The corner points and convex hull found by using the HSJ method when generating 5 alternatives.

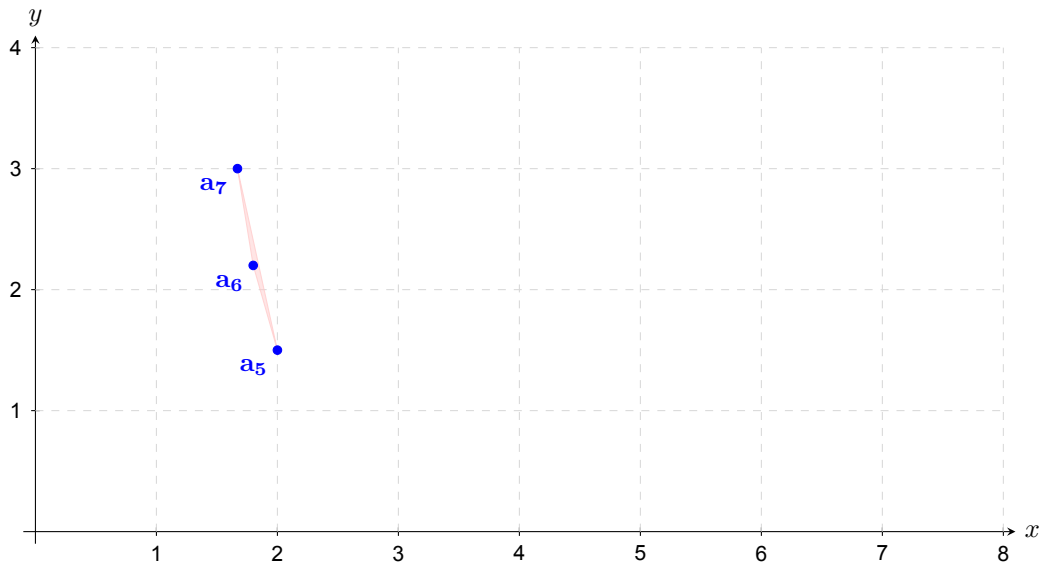


Figure 4.4: The corner points and convex hull found by using the Spores method when generating 5 alternatives.

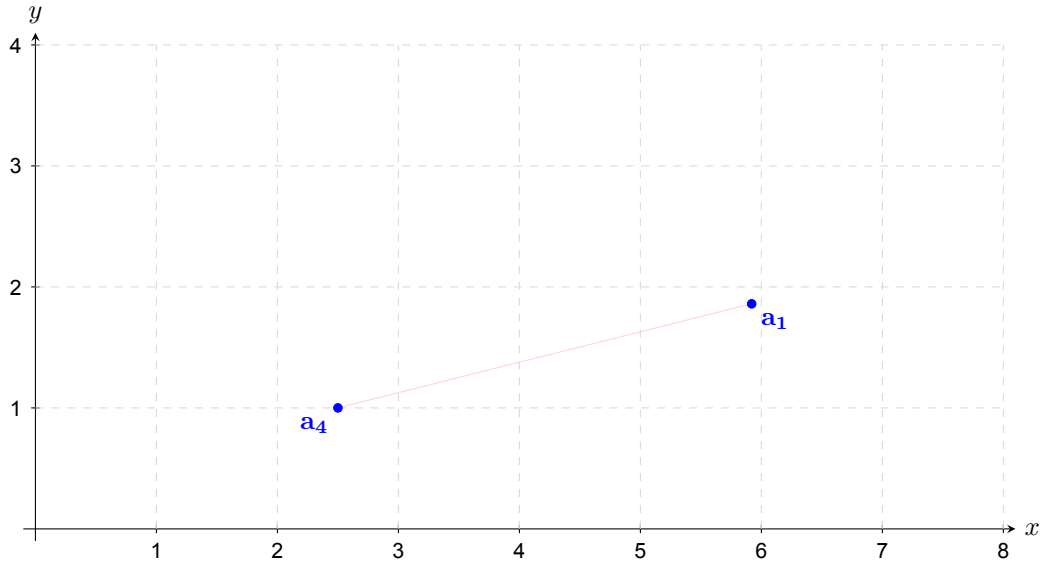


Figure 4.5: The corner points and convex hull found by using the Min/Max Variables method when generating 5 alternatives.

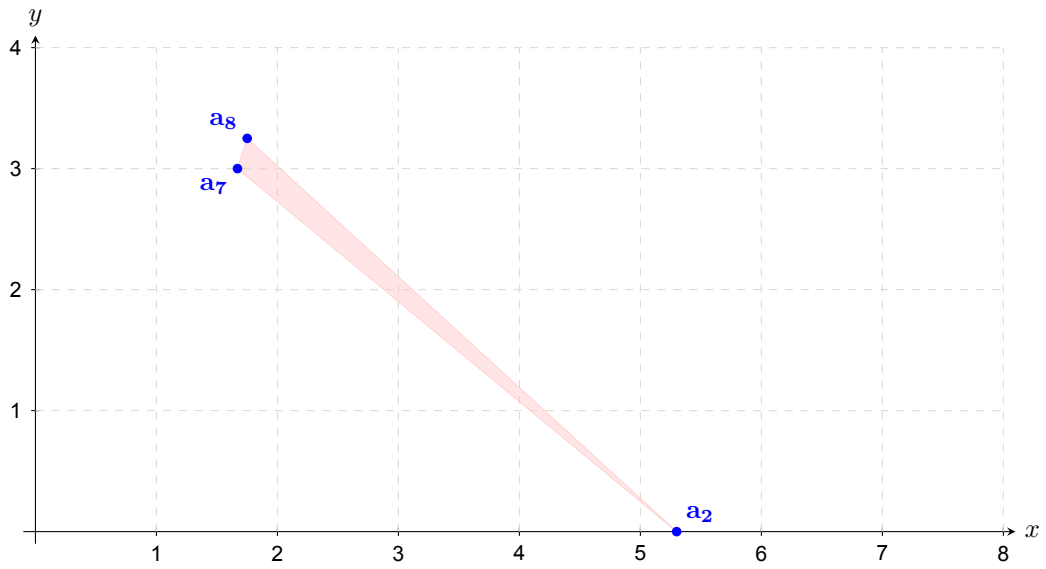


Figure 4.6: The corner points and convex hull found by using the Random Vector method when generating 5 alternatives.

Table 4.2: Number of times each reference point was generated by each method when generating 20 alternatives.

Method	$\mathbf{a_1}$	$\mathbf{a_2}$	$\mathbf{a_3}$	$\mathbf{a_4}$	$\mathbf{a_5}$	$\mathbf{a_6}$	$\mathbf{a_7}$	$\mathbf{a_8}$	set size
Max-Distances	1	9	0	0	0	0	0	10	3
HSJ	0	0	0	19	0	0	1	0	2
Spores	0	0	0	0	12	6	2	0	3
Min/Max Variables	2	4	1	0	3	0	0	10	5
Random Vector	6	4	0	1	1	0	2	6	6

Table 4.3: Total runtime of each method when generating 20 alternatives.

Method	**Runtime (s)**
Max-Distances	0.321
HSJ	0.067
Spores	0.058
Min/Max Variables	0.064
Random Vector	0.061

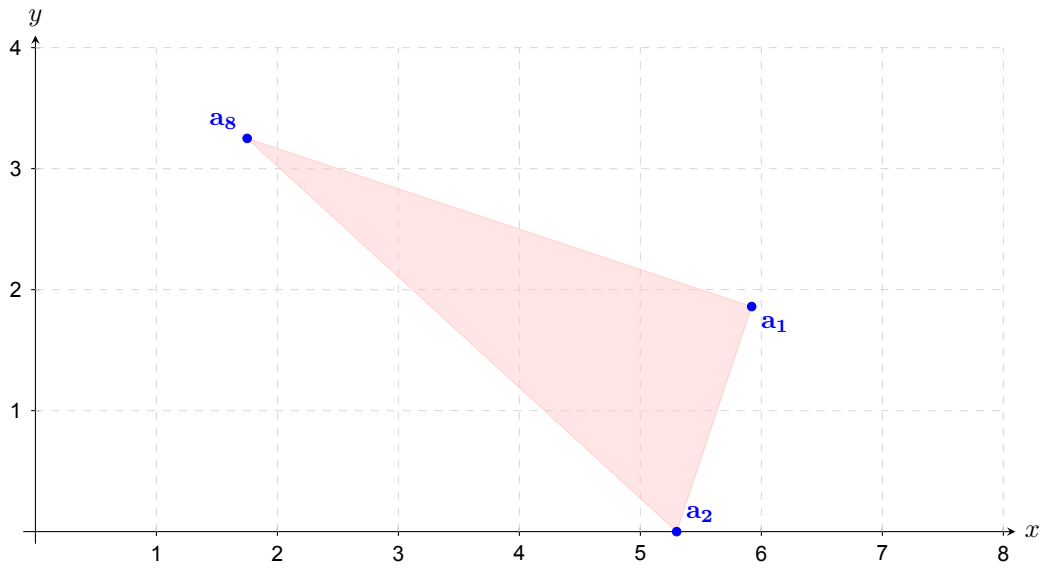


Figure 4.7: The corner points and convex hull found by using the Max-Distance method when generating 20 alternatives.

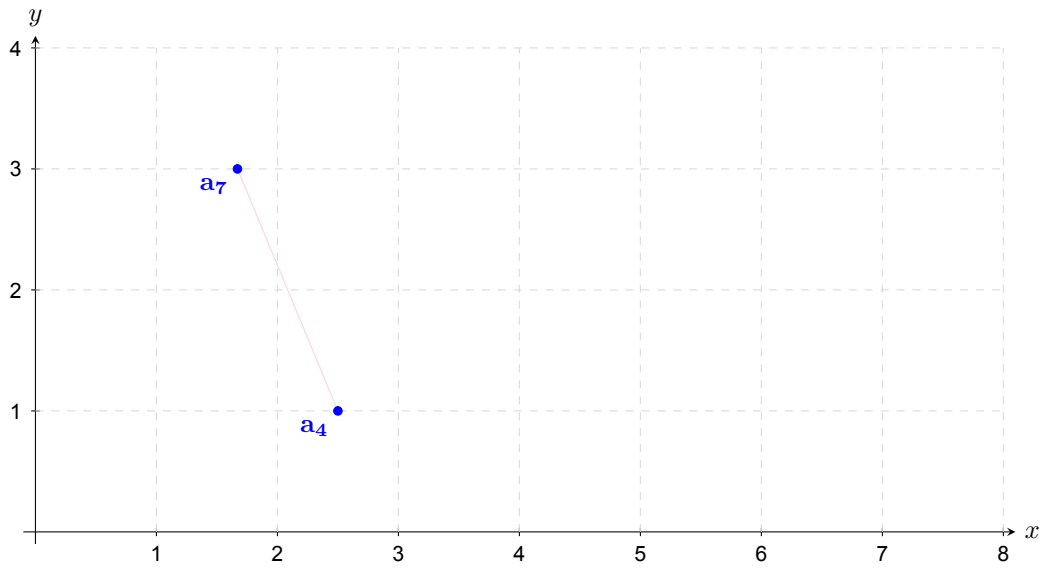


Figure 4.8: The corner points and convex hull found by using the HSJ method when generating 20 alternatives.

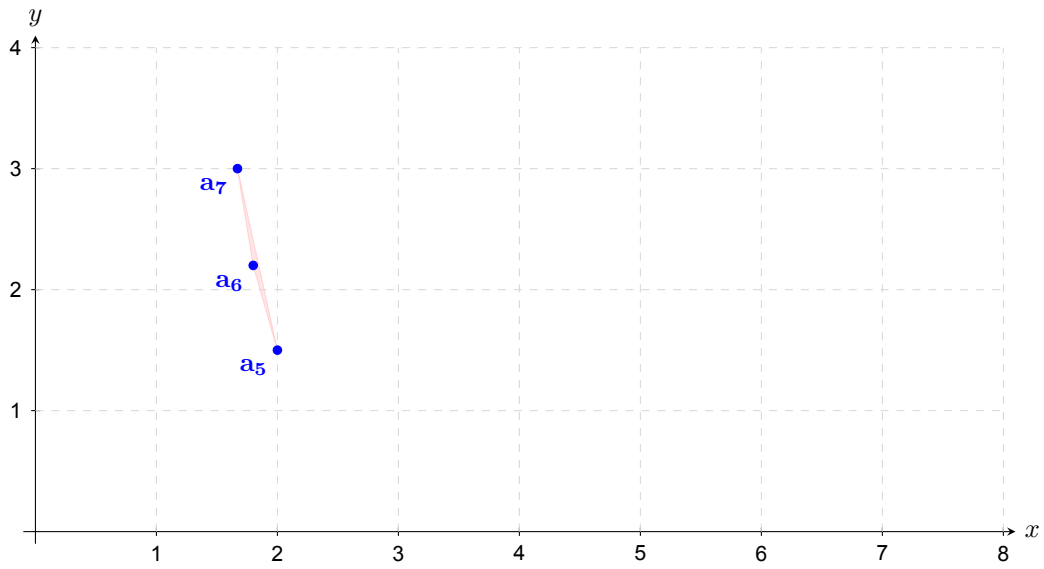


Figure 4.9: The corner points and convex hull found by using the Spores method when generating 20 alternatives.

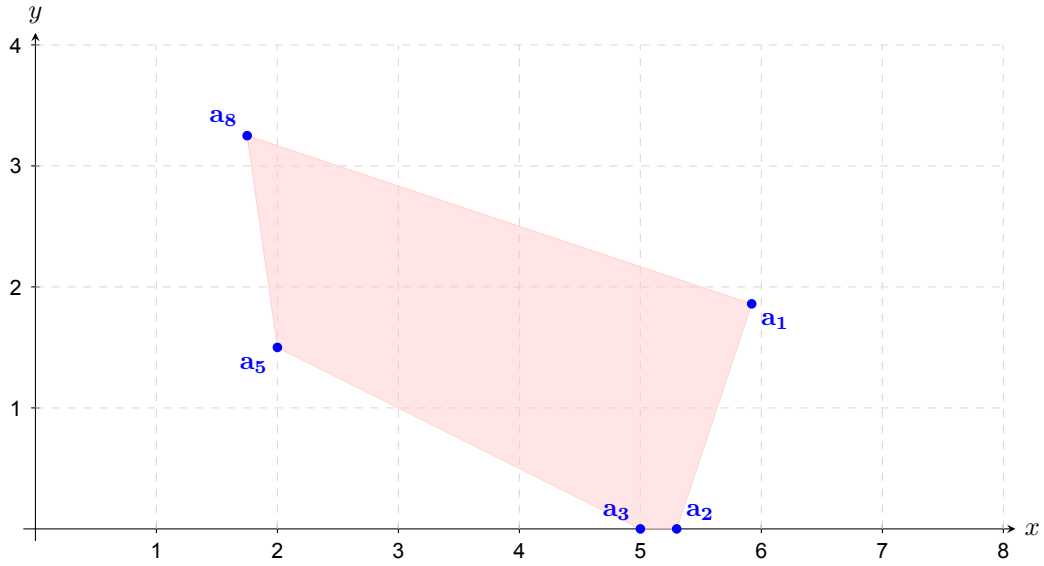


Figure 4.10: The corner points and convex hull found by using the Min/Max Variables method when generating 20 alternatives.

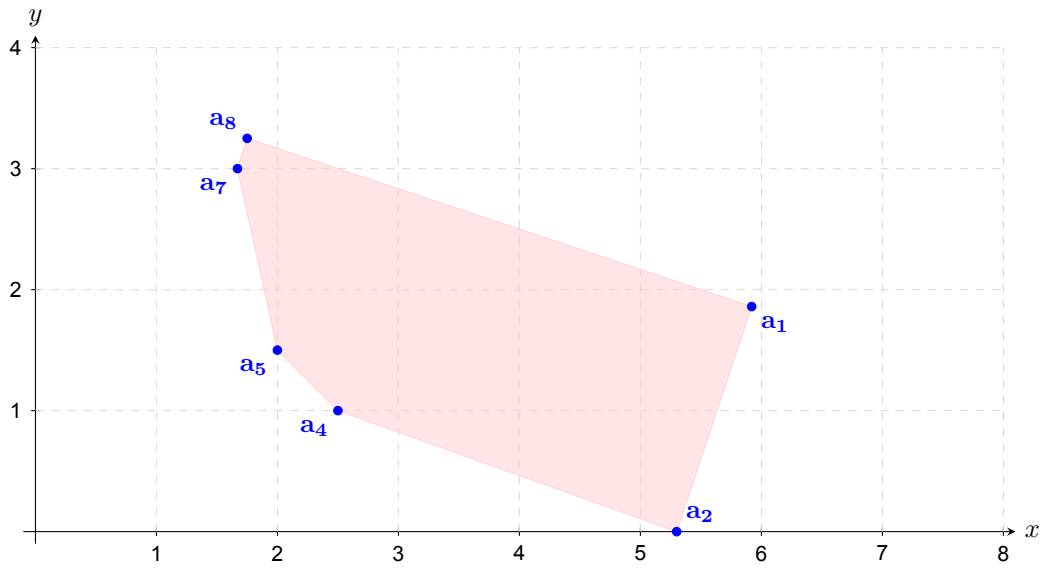


Figure 4.11: The corner points and convex hull found by using the Random Vector method when generating 20 alternatives.

An important observation is that certain deterministic methods, namely Max-Distance, Spores, and HSJ, exhibit stagnation in the generation of alternatives. Specifically, these methods fail to identify new alternatives beyond the fifth iteration, as indicated by the fact that the cardinality of the set of alternatives remains constant between iterations 5 and 20. This behavior suggests that such methods can become confined to a limited subset of the near-optimal solution space.

A similar, though less pronounced, phenomenon can be observed in stochastic search methods. In these cases, the largest increases in the number of discovered alternatives occur during the initial iterations, while subsequent iterations yield diminishing returns. This trend arises from the structure of the near-optimal solution space: once an alternative has been identified, the region it occupies is effectively covered, thereby reducing the probability that a randomly generated weight vector will yield a new alternative.

Nevertheless, stochastic approaches differ from deterministic ones in a crucial respect. Due to their inherent randomness, they retain the ability to occasionally discover new regions of the solution space, even after repeated encounters with previously identified alternatives. As a result, they avoid permanent stagnation.

It should also be noted, however, that this distinction introduces a disadvantage for stochastic methods. Since deterministic methods ultimately converge to a stable set of alternatives, their execution can be terminated as soon as convergence is detected, thereby avoiding unnecessary computation. By contrast, stochastic methods do not exhibit such a clear stopping criterion, as there is always a non-zero probability of uncovering additional alternatives, regardless of how many iterations have already been performed.

Another notable point concerns computational efficiency. As reported in Table 4.3, the Max-Distance method is significantly slower than the others, with a total runtime exceeding 0.3 seconds, nearly an order of magnitude longer than all other techniques. While this runtime might be acceptable in this relatively simple 2D example, it highlights a critical limitation. If the runtime of this method stays an order of magnitude higher for more complex models, Max-Distance is not fast enough to compete with the other methods of generating alternatives.

4.1.2. Convex hull

The convex hull of the alternatives generated by each method provides insight into how well each technique explores the near-optimal solution space. From the results shown in Figures 4.2 to 4.6, when only considering 5 iterations, we observe not that much difference in the size of the convex hull, with one notable exception:

- The Max-Distance method clearly has the biggest convex hull of all the methods.
- HSJ, Spores, Min/Max Variables, and Random Vector all have a relatively small convex hull.

From the results shown in Figures 4.7 to 4.11, when considering 20 iterations we observe notable differences in coverage:

- Random Vector and Min/Max Variables methods demonstrate the most effective exploration. Both span a large portion of the feasible near-optimal region, successfully identifying most alternatives of the near-optimal solution space.
- The Max-Distance approach performs moderately well. While it avoids clustering in a small area, it tends to get trapped bouncing between extreme points, leading to redundant alternatives in the extreme points of the near-optimal solution space and reduced overall spread.
- HSJ and Spores perform the worst in terms of coverage. Their generated alternatives are tightly clustered, missing significant portions of the near-optimal solution space. These methods fail to cover a large part of the near-optimal solution space.

In summary, when only considering 5 iterations, Max-Distance performs best. When considering 20 iterations, stochastic heuristics (Random Vector, Min/Max Variables) offer better convex coverage, while deterministic heuristics (HSJ, Spores, and Max-Distance) often struggle to escape regions of found alternatives.

4.1.3. Shadow pricing

Shadow pricing evaluates the robustness of alternatives by examining their proximity to constraint boundaries, as reflected by the dual values. From this perspective, alternatives further from active constraints are considered more robust. In this example the most robust alternatives are $\mathbf{a_1}$ followed by $\mathbf{a_2}$ and $\mathbf{a_8}$.

When considering 5 iterations, the results show that:

- Max-Distance identifies only the most robust alternatives, including all of the most robust alternatives, which exhibit the lowest shadow prices and lie furthest from multiple constraint boundaries.
- Min/Max Variables and Random Vector also identify robust alternatives, but their search does not find all alternatives that are considered to be the most robust.
- HSJ and Spores produce alternatives with consistently high shadow prices. This suggests that the alternatives they generate lie close to active constraints and lack robustness.

When considering 20 iterations, the results show that:

- Max-Distance identifies only the most robust alternatives, including all of the most robust alternatives, which exhibit the lowest shadow prices and lie furthest from multiple constraint boundaries.
- Min/Max Variables and Random Vector also identify robust alternatives, but their search includes a broader set of alternatives, some of which have higher shadow prices and thus lower robustness.
- HSJ and Spores produce alternatives with consistently high shadow prices. This suggests that the alternatives they generate lie close to active constraints and lack robustness.

In summary, while Max-Distance prioritizes robustness by targeting alternatives distant from constraints, Min/Max Variables and Random Vector identify robust alternatives, but do not focus on finding them. HSJ and Spores, however, are less effective in identifying robust alternatives due to their narrow search focus.

4.1.4. Knowledge Gap

While quantitative metrics such as convex hull and shadow pricing provide useful insights into the performance of MGA techniques, they do not fully capture the nuances that influence method effectiveness in real-world applications. Despite performing the worst on both spread and robustness metrics in both cases of 5 and 20 alternatives, HSJ and Spores are considered state-of-the-art methods when generating alternatives [11, 20, 21, 30]. This indicates a gap between what current metrics capture and

what decision makers use to help decision-making. Possible explanations may include interpretability, perceived alignment with user preferences, or decision-relevant alternative selection of these methods.

This gap reinforces the need for a broader evaluation framework. Metrics like convex hull and shadow pricing, while useful, are not sufficient on their own to assess decision-relevance in alternative generation, as is supported by what decision makers actually use. This gap motivates the exploration of multi-objective evaluation techniques in the following section, which aims to compare MGA methods by accounting for trade-offs between the decision variables of the model.

4.2. Multi-Objective Optimization as a Framework for Evaluating Alternatives

A clear theoretical framework for evaluating the decision-making value of alternatives is still missing. This thesis introduces a new perspective: interpreting the evaluation of near-optimal alternatives through the lens of multi-objective optimization (MOO). While MOO has been widely used in real-world applications [13]. It has not been used to evaluate sets of alternatives produced by near-optimal generation methods.

The key insight is: even if a model is formally optimized using a single objective function (e.g., cost), the decision variables themselves often reflect implicit trade-offs between competing priorities. This allows us to interpret each decision variable as its own implicit objective. Under this view, each near-optimal alternative represents a particular trade-off between the different decision variables.

From this perspective, we can borrow a central concept from multi-objective optimization: the Pareto front. An alternative is Pareto-optimal (or non-dominated) if no objective can be improved without worsening another. Analogously, in the context of near-optimal alternatives, an alternative is decision-relevant (or non-dominated) if it represents a distinct trade-off between its decision variables.

This interpretation leads to an important realization: current evaluation metrics fail to distinguish between alternatives that are truly decision-relevant and those that are not. An alternative that increases investment across all technologies, for instance, may appear “diverse” in convex hull space, yet offer no strategic value to a policymaker.

To the best of my knowledge, this thesis is the first attempt to apply multi-objective optimization theory, not to generate alternatives, but to evaluate their structural and strategic usefulness. The next section illustrates this concept through a two-dimensional example and motivates the need for a new evaluation metric based on dominance relations among alternatives.

4.3. Filtering for Decision-Relevant Alternatives

If this reasoning is applied to the two-dimensional illustrative example, we can directly inspect and compare specific alternatives to highlight the power of the multi-objective perspective.

Let us consider three alternatives: $\mathbf{a_1}$, $\mathbf{a_5}$, and $\mathbf{a_7}$. The two decision variables, x and y, are interpreted as proxies for competing objectives, e.g., investments in renewable and fossil infrastructure. Under this interpretation, each alternative corresponds to a vector in the objective space, with component magnitudes representing the level of investment in each “objective.”

$$\begin{aligned}\mathbf{a_1} &: (x, y) = (5.92,\ 1.86)\\ \mathbf{a_5} &: (x, y) = (2,\ 1.5)\\ \mathbf{a_7} &: (x, y) = (1.67,\ 3.0)\end{aligned}$$

Visualized as bars or vectors, we observe:

- $\mathbf{a_5}$ requires 2 units of renewable and 1.5 units of fossil investments.
- $\mathbf{a_7}$ emphasizes fossil over renewable, with 1.67 and 3.0 units, respectively.
- $\mathbf{a_1}$ has significantly higher values on both axes: 5.92 and 1.86.

Now, applying the logic of dominance from multi-objective optimization:

- Between $\mathbf{a_5}$ and $\mathbf{a_7}$, neither dominates the other. $\mathbf{a_5}$ uses less fossil input, while $\mathbf{a_7}$ uses less renewable. Each represents a different trade-off, both valid depending on decision-maker preferences.
- However, when comparing $\mathbf{a_1}$ with $\mathbf{a_5}$, we see that $\mathbf{a_1}$ has higher values on both x and y. That is, it requires more of both renewable and fossil investments compared to $\mathbf{a_5}$, while yielding a near-optimal cost.

This implies that $\mathbf{a_1}$ is dominated by $\mathbf{a_5}$, there are no trade-offs to justify choosing $\mathbf{a_1}$, as $\mathbf{a_5}$ is strictly better on both decision variables.

This small example underscores a crucial point:

> *Some alternatives, although feasible and near-optimal in terms of cost, are not decision-relevant because they are structurally worse than other alternatives across all decision variables.*

Current metrics like convex hull area or even shadow pricing would still consider $\mathbf{a_1}$ a valid and even desirable addition to a set of alternatives because it expands the geometric coverage of the set and is distant from most constraint boundaries. But from a decision-making standpoint, it is not useful; it adds redundancy and potentially misleads stakeholders.

Thus, the multi-objective lens helps filter out dominated alternatives and identify those that represent meaningful trade-offs, improving both the quality and interpretability of the set of alternatives. When the problem is analyzed using dominance-based criteria, alternatives that are non-dominated are preferred. These alternatives are shown in Figure 4.12. When comparing current methods to these decision-relevant alternatives, an important observation emerges: some alternatives that are undervalued by traditional metrics, but favored by domain experts, perform well under the multi-objective perspective. For instance, the state-of-the-art methods HSJ and Spores, which are not prioritized by existing quantitative metrics due to convex hull or robustness, are revealed to be preferred when evaluated in terms of dominance and trade-off relevance. This observation reinforces the limitations of conventional metrics in isolating decision-relevant alternatives and highlights the value of incorporating a multi-objective perspective.

This insight forms the foundation of the dominance metric introduced in chapter 5, and motivates the development of the method in chapter 6, which explicitly searches for non-dominated and thus decision-relevant alternatives.

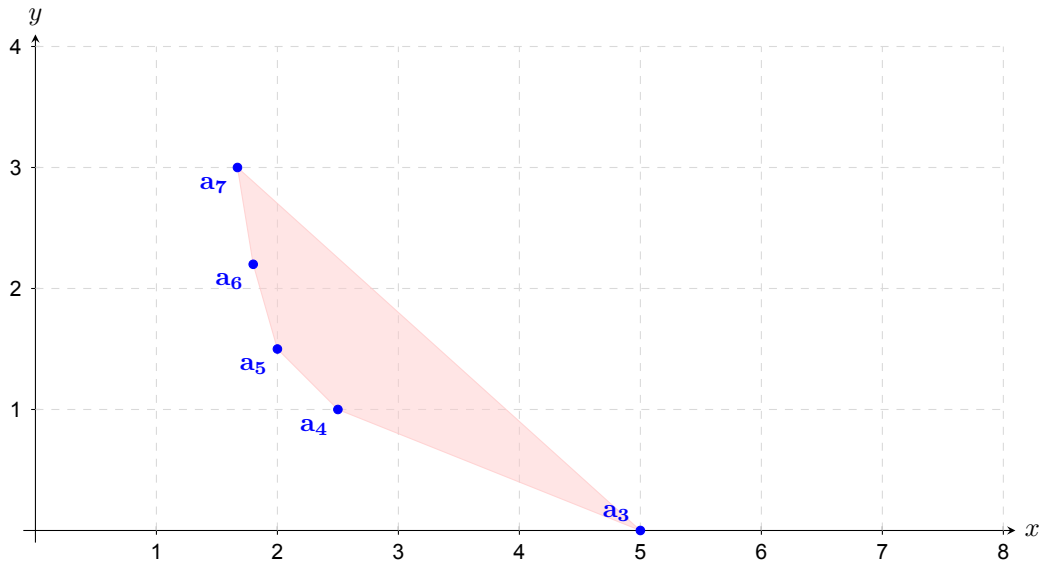


Figure 4.12: The corner points and convex hull found when analyzing for non-dominated alternatives

4.4. Summary

This chapter investigates different approaches for generating near-optimal alternatives in optimization problems. It evaluates several established methods, the stochastic techniques (Random Vector and Min/Max Variables), and the deterministic methods (HSJ, Spores, and Max-Distance). The analysis reveals that stochastic methods generally produce a wider spread of alternatives, effectively covering more of the near-optimal solution space and offering greater coverage. In contrast, deterministic methods often generate alternatives that are more specialized and less diverse.

Even though the stochastic methods seem to outperform the deterministic methods in most metrics, HSJ and Spores are still preferred by experts. This discrepancy between the observations and usage in the real world indicates a gap in current understanding. To combat this gap, the chapter introduces a multi-objective view that identifies alternatives that are decision-relevant. This perspective allows for a deeper understanding of the trade-offs between alternatives and claims that:

> *Some alternatives, although feasible and near-optimal in terms of cost, are not decision-relevant because they are structurally worse than other alternatives across all decision variables.*

The rest of the thesis investigates this claim to see if a multi-objective perspective can reliably distinguish between truly valuable alternatives and those that are merely mathematically feasible. By applying this perspective, the research aims to identify which alternatives provide meaningful trade-offs for decision-makers and which can be excluded without loss of strategic value.

5

Dominance

Decision-making often involves evaluating a set of alternatives. In such contexts, it is essential to distinguish between genuinely valuable alternatives and those that offer no meaningful improvement. This chapter introduces a dominance-based framework to formally assess and compare alternatives.

This chapter begins by defining the notion of dominance between alternatives, establishing the foundation for identifying non-dominated alternatives. The chapter extends this concept by introducing strict dominance, dominance equivalence, and dominance regions, all supported by different visualizations. These concepts enable us to move beyond binary comparisons and a metric for evaluating sets of alternatives, which we will call dominance.

Finally, we apply this new dominance metric to the evaluation of alternative-generation methods. By defining the dominant set and introducing a layered dominance ranking, we create a quantitative basis for comparing methods in terms of the quality and variety of the alternatives they produce. This enables a more meaningful comparison than traditional geometric metrics such as spread or convex coverage.

5.1. Dominance

To address the challenge of quantitatively comparing methods for identifying alternatives, this thesis introduces a new concept to alternative generation, called dominance, which is defined and used as a basis for comparison between different alternatives.

Definition 5.1.1 (Dominance)**.** Let $\mathbf{k}$ and $\mathbf{l}$ be vectors containing the variable assignments of two alternatives. Let k_i refer to the variable value of alternative $\mathbf{k}$ with index i and l_i to the variable value of alternative $\mathbf{l}$ with index i. Let c_i be the coefficient of the variable i in the objective function. We say that $\mathbf{k}$ dominates $\mathbf{l}$, denoted $\mathbf{k} \succeq \mathbf{l}$, if:

$$\forall i \; c_i k_i \leq c_i l_i \tag{5.1}$$

For clarity, this can be equivalently written case-by-case as:

$$\forall i, \begin{cases} k_i \leq l_i & \text{if } c_i > 0, \\ k_i \geq l_i & \text{if } c_i < 0, \\ \text{true} & \text{if } c_i = 0. \end{cases} \tag{5.2}$$

To establish the practical relevance of the dominance relation, it is necessary to demonstrate its implications for the evaluation of alternatives with respect to the objective function. The following lemma formalizes this connection by showing that if one alternative dominates another according to the given definition, then it yields no worse objective function values across all criteria. This result provides a

theoretical justification for employing dominance as a quantitative basis for comparing alternatives.

Lemma 5.1.1. $\mathbf{k} \succeq \mathbf{l} \rightarrow f(\mathbf{k}) \leq f(\mathbf{l})$

Proof. Recall the definition of the linear objective functions

$$f(\mathbf{x}) = \sum_i c_i x_i \tag{5.3}$$

Also it is given that $\mathbf{k} \succeq \mathbf{l}$ thus:

$$\forall i \; c_i k_i \leq c_i l_i \tag{5.4}$$

Sum both sides over all i, we get:

$$\sum_i c_i k_i \leq \sum_i c_i l_i \tag{5.5}$$

By the definition of $f(\mathbf{x})$ in (5.3), this is equivalent to:

$$f(\mathbf{k}) \leq f(\mathbf{l}) \tag{5.6}$$

Thus, we can conclude that:

$$\mathbf{k} \succeq \mathbf{l} \; \rightarrow \; f(\mathbf{k}) \leq f(\mathbf{l}) \tag{5.7}$$

□

While the dominance relation provides a means of comparing alternatives, it is important to characterize other conditions with which two alternatives can be compared. In case two alternatives dominate each other, the notion of equivalence is introduced.

Definition 5.1.2 (Equivalence)**.** Let $\mathbf{k}$ and $\mathbf{l}$ be two alternatives. Alternatives $\mathbf{k}$ and $\mathbf{l}$ are equivalent with respect to dominance, denoted $\mathbf{k} \sim \mathbf{l}$, if and only if:

$$\mathbf{k} \succeq \mathbf{l} \; \wedge \; \mathbf{l} \succeq \mathbf{k} \tag{5.8}$$

To distinguish cases in which one alternative is not only not worse but strictly better than another, we introduce the notion of strict dominance.

Definition 5.1.3 (Strict Dominance)**.** Let $\mathbf{k}$ and $\mathbf{l}$ be two alternatives. Alternative $\mathbf{k}$ strictly dominates $\mathbf{l}$, denoted $\mathbf{k} \succ \mathbf{l}$, if and only if:

$$\mathbf{k} \succeq \mathbf{l} \; \wedge \; \mathbf{l} \not\succeq \mathbf{k} \tag{5.9}$$

That is, $\mathbf{k}$ strictly dominates $\mathbf{l}$ if it dominates $\mathbf{l}$ according to the previously defined relation, and the two alternatives are not identical in their variable assignments. This definition ensures that strict dominance captures genuine improvement in at least one dimension, while maintaining non-inferiority across all others.

In some cases, two alternatives may not dominate one another in either direction. To capture this situation, we define the notion of dominance equivalence.

Definition 5.1.4 (Incomparable)**.** Let $\mathbf{k}$ and $\mathbf{l}$ be two alternatives. Alternatives $\mathbf{k}$ and $\mathbf{l}$ are incomparable with respect to dominance, denoted $\mathbf{k} \mid \mathbf{l}$, if and only if:

$$\mathbf{k} \not\succeq \mathbf{l} \; \wedge \; \mathbf{l} \not\succeq \mathbf{k} \tag{5.10}$$

That is, $\mathbf{k}$ and $\mathbf{l}$ are dominance-incomparable if neither alternative dominates the other under the dominance relation previously defined. This indicates that the alternatives are incomparable in terms of dominance and suggests that each may offer trade-offs that prevent a clear ordering according to the given objective functions.

In addition to comparing pairs of alternatives, it is often useful to understand the extent of an alternative's dominance, that is, the set of alternatives it dominates. This leads to the concept of a dominance region, which captures the full scope of alternatives that are no better than a given alternative under the dominance relation. The dominance region can be thought of as a dominance "footprint" in the space of all feasible alternatives.

Definition 5.1.5 (Dominance Region)**.** Given an alternative $\mathbf{k}$, the dominance region of $\mathbf{k}$, denoted $\mathcal{D}(\mathbf{k})$, is the region of near-optimal solution space $\mathcal{Q}$ where all alternatives $\mathbf{l}$ that lie in this region $\mathcal{Q}$ are dominated by $\mathbf{k}$ such that $\mathbf{k} \succeq \mathbf{l}$, i.e., the region of all near-optimal solution space dominated by $\mathbf{k}$:

$$\mathcal{D}(\mathbf{k}) = \{\mathbf{l} \in \mathcal{Q} \mid \mathbf{k} \succeq \mathbf{l}\}. \tag{5.11}$$

5.2. Visualization of Dominance

To provide an intuitive understanding of the dominance relation and its variants, we now turn to a geometric interpretation in two dimensions. The corresponding visualizations illustrate how dominance, strict dominance, and dominance equivalence manifest when alternatives are represented as points in a two-dimensional variable space.

In Figure 5.1, a fixed alternative $\mathbf{a_1}$ is shown along with its dominance region. This example assumes the objective function to be minimized is $f(x, y) = x + y$, and thus, the dominance region consists of points that lie to the upper right of $\mathbf{a_1}$ in the coordinate space (i.e., with greater or equal values in both dimensions). This region reflects the structure imposed by the dominance relation, whereby any point lying within it is considered no better than $\mathbf{a_1}$ in terms of dominance and thus objective functions. In this example, one can see that $\mathbf{a_1} \succ \mathbf{a_2}$, $\mathbf{a_1} \succ \mathbf{a_7}$, and $\mathbf{a_1} \succ \mathbf{a_6}$.

Figure 5.2 illustrates the concept of dominance incomparability. Two alternatives, $\mathbf{a_1}$ and $\mathbf{a_5}$, are shown along with their respective dominance regions. The figure makes clear that neither alternative is fully contained within the other: neither dominates nor is dominated by the other. This mutual lack of dominance aligns with the formal definition of incomparability in dominance, denoted $\mathbf{a_1} \mid \mathbf{a_5}$.

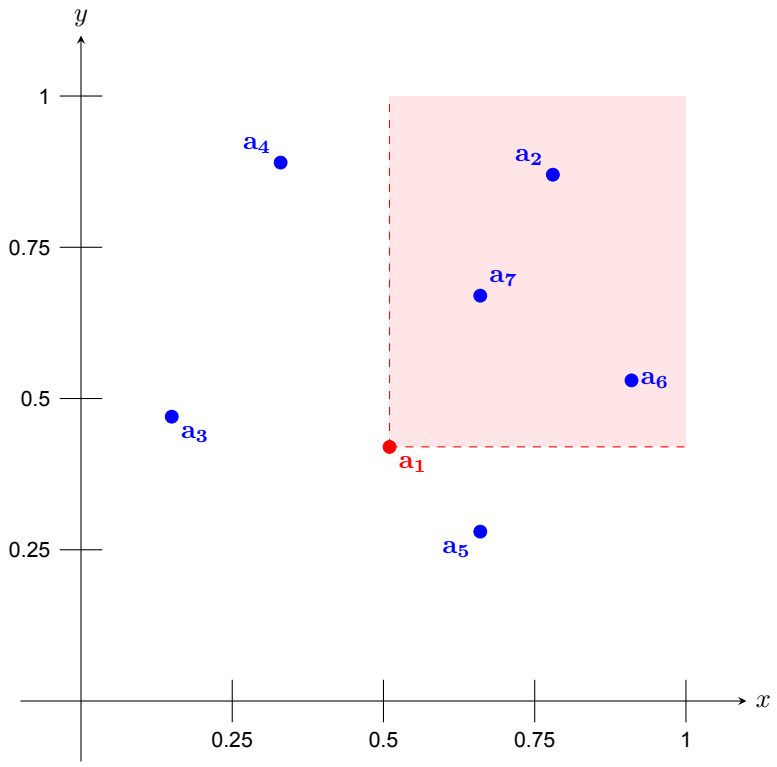


Figure 5.1: Dominance region of $\mathbf{a_1}$

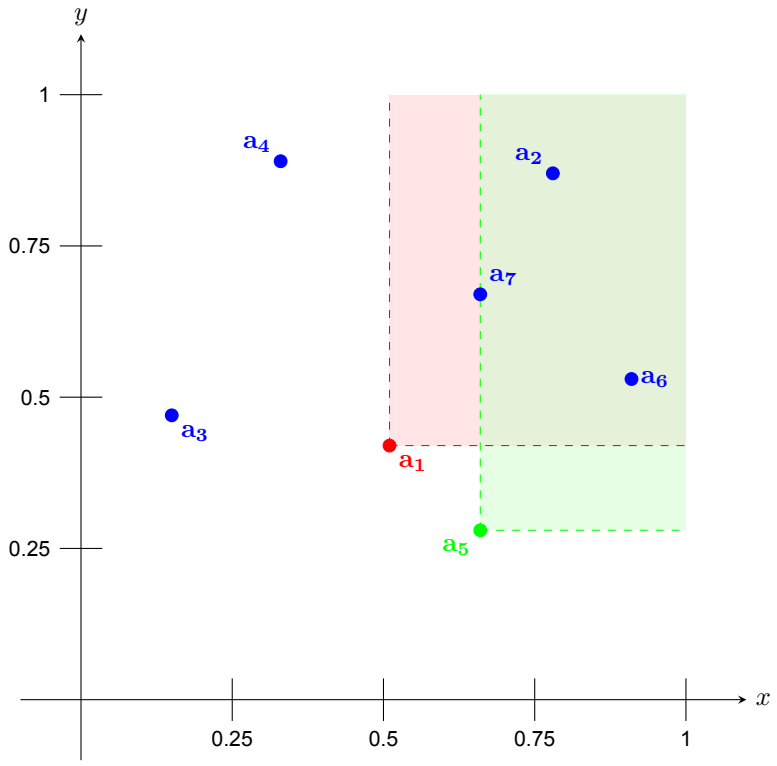


Figure 5.2: Dominance incomparability of $\mathbf{a_1}$ and $\mathbf{a_5}$

5.3. Dominant Set

In addition to evaluating individual alternatives, it is often useful to assess the quality of an entire set of alternatives generated by a given method. To this end, we introduce the concept of the dominant set.

Definition 5.3.1 (Dominant Set)**.** Given a set of alternatives S, the dominant set of S is defined as the subset of alternatives in S that are not dominated by any other alternative in S. That is,

$$Dmn(S) = \{\mathbf{k} \in S \mid \nexists \mathbf{l} \in S \text{ such that } \mathbf{l} \succ \mathbf{k}\} \tag{5.12}$$

In Figure 5.3, the dominant set and its combined dominance region are shown. In this example, three alternatives: $\mathbf{a_1}$, $\mathbf{a_3}$, and $\mathbf{a_5}$ are non-dominated. These alternatives are shown along with the regions that they dominate, emphasizing their relative advantage over the other alternatives in the set. The shaded region represents the subset of the space for which the corresponding alternative offers performance that is no worse in any objective and strictly better in at least one.

Figure 5.3 clearly shows that all other alternatives lie within this dominated region, indicating that they are strictly worse than one of $\mathbf{a_1}$, $\mathbf{a_3}$, or $\mathbf{a_5}$ and therefore excluded from the dominant set. This visual structure reinforces the definition of the dominant set as the collection of alternatives that are not strictly dominated by any other within the set.

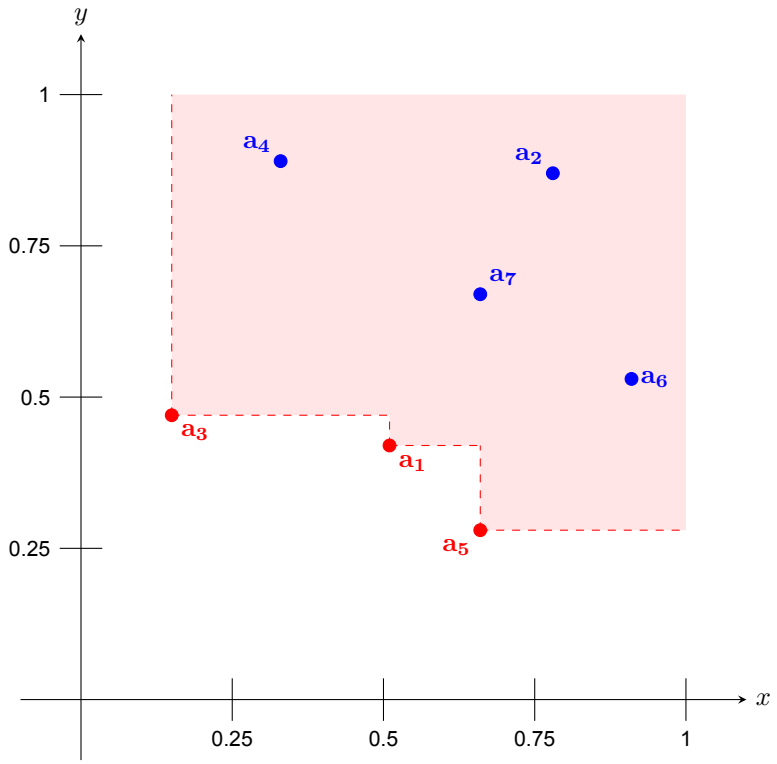


Figure 5.3: The dominated set (red) and the region that they dominate with dominated alternatives (blue)

5.4. Dominance Ranking

The dominance-based metric introduced in this chapter not only allows for a clear comparison between individual alternatives. It also opens the door for more advanced analysis and the ranking of alternatives. This allows for a deeper evaluation of the entire alternative generating methods. The framework of dominance is extended to hierarchically rank the alternatives based on the degree to which they are dominated.

While the current dominance definition provides a binary classification, alternatives are either dominated or non-dominated. More information can be derived when ranking them in successive layers of

dominance. This process results in a dominance ranking, where alternatives are grouped by the extent to which they are dominated.

Definition 5.4.1 (Dominance ranking)**.** Given a set of alternatives S, dominance ranking is defined on levels as follows:
Let D_1 be the dominant set S.
Let D_2 be the set of all alternatives $S \setminus D_1$ that is dominated by at least one alternatives in D_1.
Let D_3 be the set of all alternatives $S \setminus (D_1 \cup D_2)$ that is dominated by at least one alternative in D_2.
Etc.

In this process D_0 is defined as:

$$D_0 = \emptyset \tag{5.13}$$

This process continues iteratively defining D_k for $k > 0, k \in \mathbb{N}$ as:

$$D_k(S) = \left\{ \mathbf{k} \in S \setminus \bigcup_{1}^{k-1} D_i \;\middle|\; \exists \mathbf{l} \in D_{k-1} \text{ such that } \mathbf{l} \succ \mathbf{k} \;\wedge\; \nexists \mathbf{m} \in D_{k+1} \text{ such that } \mathbf{m} \succ \mathbf{k} \right\}. \tag{5.14}$$

This results in ranking each alternative in some D_k until $S \setminus (D_0 \cup ... \cup D_k)$ is an empty set.

This layered structure provides a more nuanced evaluation of alternative quality than the dominant set alone. For example, in comparing methods that generate alternatives, one may not only assess the size of the dominant set (i.e., D_1), but also examine the distribution of alternatives across dominance levels. A method that generates more alternatives in the higher tiers (e.g., $D_1 \cup D_2$) may be preferred over one whose alternatives mostly occupy lower-ranked levels.

This layered structure is also illustrated visually. In the corresponding Figure 5.4, alternatives are plotted in a two-dimensional objective space, and grouped by their dominance level. The first level, D_1, contains the non-dominated alternatives, which form the outer frontier; no other alternative in the set dominates any of them. These are typically positioned toward the lower-left corner in minimization problems, representing optimal trade-offs.

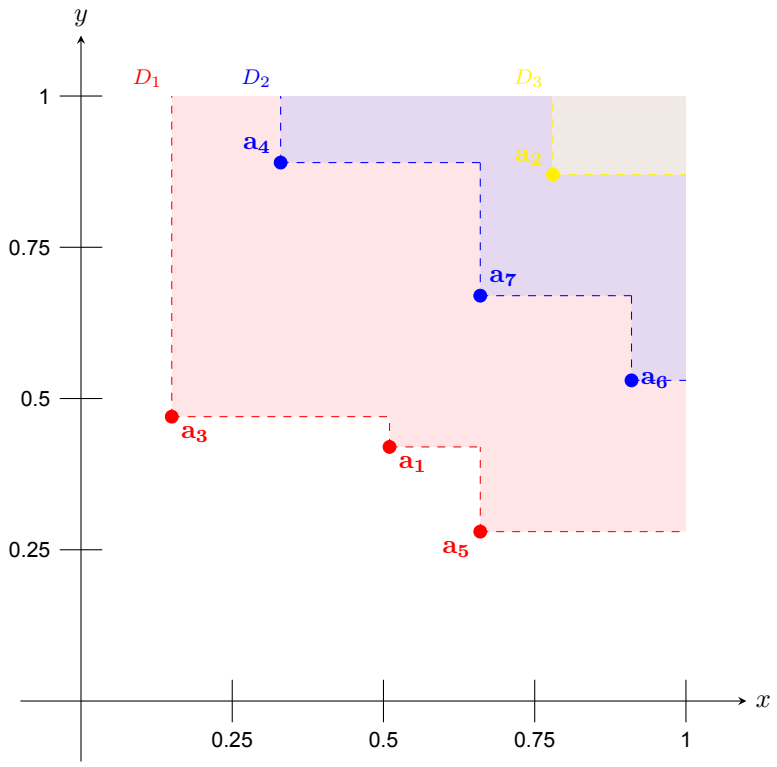


Figure 5.4: Ranking of the alternatives

5.5. Comparing Methods using Dominance

The concept of a dominant set allows for a principled comparison of methods based on the quality of the alternatives they produce. Let method x produce a set of representative alternatives X, and method y produce a set of representative alternatives Y. We say that method x is preferred over method y if in $Dmn(X \cup Y)$ there are more alternatives from set X then from set Y, i.e.

$$\left|Dmn(X \cup Y) \cap X\right| \geq \left|Dmn(X \cup Y) \cap Y\right|. \tag{5.15}$$

This criterion focuses on the number of distinct, non-dominated alternatives, those that are not strictly worse than any others within the set. In doing so, it measures the decision-making value of the generated alternative sets. A method that contributes more non-dominated alternatives offers a broader and potentially more useful range of choices, enhancing the decision space with options that are clearly not inferior to others.

A crucial reason why the dominant set should only include representative alternatives and not all generated alternatives is to avoid infinite cardinality. If convex combinations of dominant alternatives are allowed, then infinite alternatives can be generated, this is because the property of a convex problem is that any convex combination between two alternatives is also a valid alternative. This is a problem because it is impossible to numerically compare the sizes of two dominant sets if both sets have infinite alternatives. Thus, before starting a comparison between two dominant sets, the convex combinations should first be removed from those sets.

For example, suppose method x generates the alternatives $X = \{\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}\}$, and method y generates the alternatives $Y = \{\mathbf{a_4}, \mathbf{a_5}, \mathbf{a_6}, \mathbf{a_7}\}$, as visualized in Figure 5.5. Note that $\mathbf{a_7}$ is a convex combination of $\mathbf{a_4}$, $\mathbf{a_5}$, and $\mathbf{a_6}$ it should be removed from set Y. Now, consider the dominant set of these alternatives:

$$Dmn(X \cup Y) = \{\mathbf{a_1}, \mathbf{a_3}, \mathbf{a_5}\}, \tag{5.16}$$

where $\mathbf{a_1}, \mathbf{a_3} \in X$ and $\mathbf{a_5} \in Y$. In this case, we have:

$$\left|Dmn(X \cup Y) \cap X\right| = 2 \quad \text{and} \quad \left|Dmn(X \cup Y) \cap Y\right| = 1. \tag{5.17}$$

According to the criterion, since more elements in the dominant set come from X than from Y, method x is preferred over method y. In this thesis, this criterion is used to compare methods that generate alternatives.

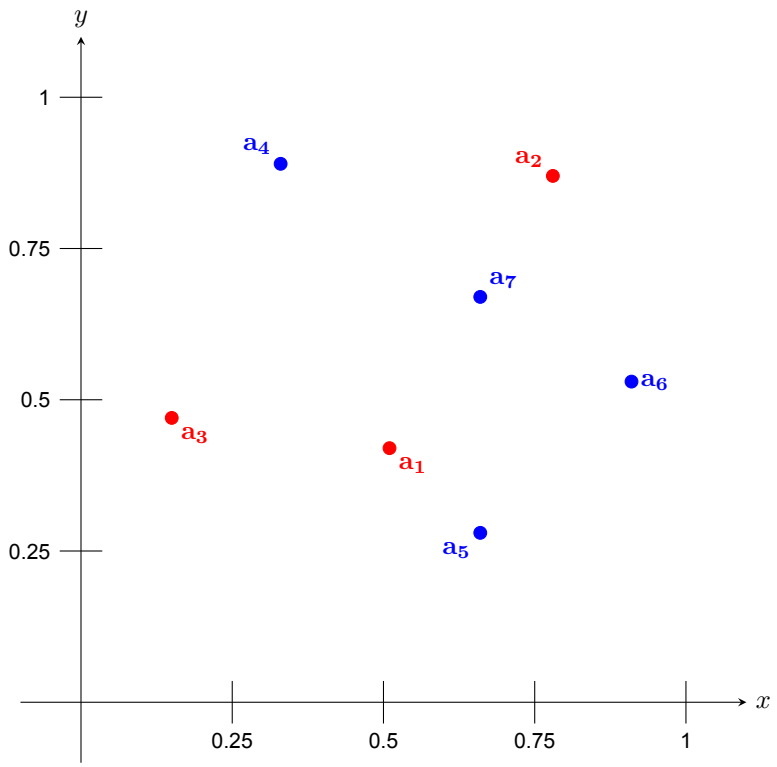


Figure 5.5: Visualization of alternatives from method x in red and method y in blue

5.6. Extension to Dominant Set and Dominance Ranking

In this section, we explore an extension to the concept of the dominance set and dominance ranking. Specifically, we consider how convex combinations of alternatives can be leveraged to refine the dominance set, allowing for the exclusion of more alternatives than the current definition permits. While the existing framework restricts the use of convex combinations when forming sets of alternatives, it does not impose the same limitations when comparing these sets. By explicitly incorporating convex combinations into the comparison process, we can achieve a more discriminating dominance ranking that better captures the relative strengths of alternative options.

Definition 5.6.1 (Convex Dominant Set)**.** Given a set of alternatives S, the dominant set of S is defined as the subset of alternatives in S that are not dominated by any convex combination of alternatives in S. That is,

$$Dmn^c(S) = \{\mathbf{k} \in S \mid \nexists \mathbf{l} \in conv(S) \text{ such that } \mathbf{l} \succ \mathbf{k}\}, \tag{5.18}$$

where $conv(S)$ denotes the convex hull of S, i.e., the set of all convex combinations of elements in S.

In Figure 5.6, the convex dominant set and its combined dominance region are shown. In this example, two alternatives: $\mathbf{a_3}$ and $\mathbf{a_5}$ are non-dominated. Which is different compared to just the dominant set, which also included $\mathbf{a_1}$. These alternatives are shown along with the regions that they dominate, emphasizing their relative advantage over the other alternatives in the set. The shaded region represents the subset of the space that is dominated by either $\mathbf{a_3}$, $\mathbf{a_5}$, or a convex combination of $\mathbf{a_3}$ and $\mathbf{a_5}$. To further extend the use of the convex dominant set, the alternatives can also be ranked based on their convex dominance. Which leads to Figure 5.7.

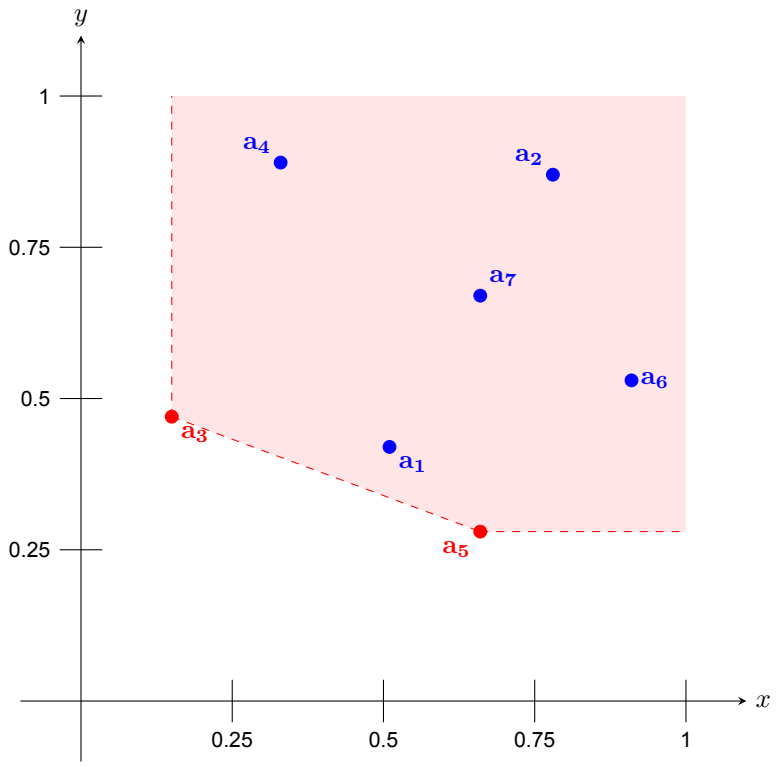


Figure 5.6: The dominated set (red) and the region that they dominate with dominated alternatives (blue)

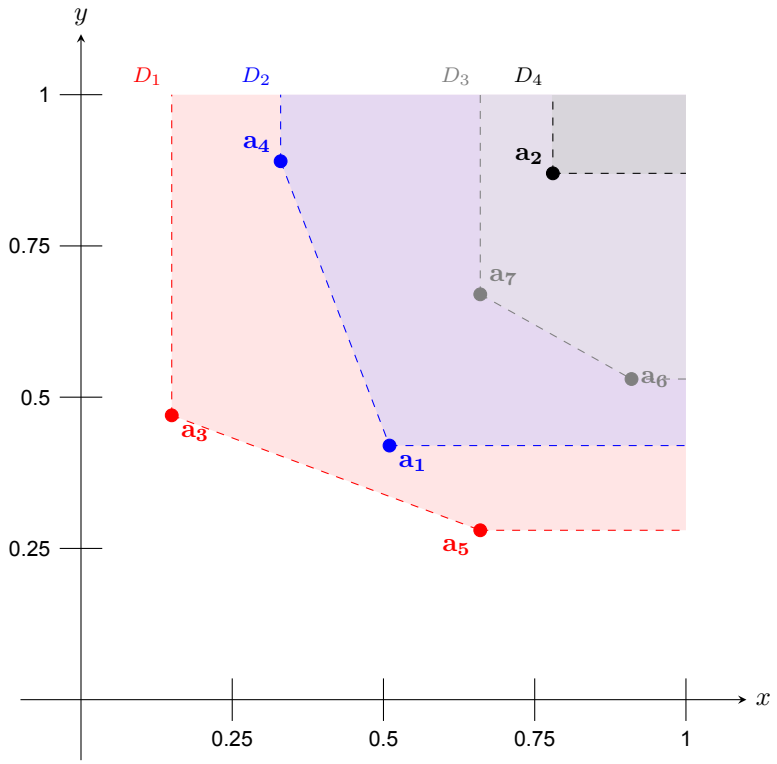


Figure 5.7: Ranking of the alternatives when using a convex dominant set

5.7. Summary

The dominance-based framework presented in this chapter provides a principled methodology for evaluating alternatives and sets of alternatives. By defining formal dominance relations and the dominant set, it becomes possible to distinguish between alternatives not only in terms of absolute performance but also in terms of their relative position within the decision space.

In the context of evaluating alternative-generation methods, the dominant set offers a meaningful criterion for assessing the usefulness of a set of alternatives. Methods that produce a higher proportion of non-dominated alternatives provide greater decision-making value. The layered dominance ranking further refines this assessment by identifying alternatives that, while dominated, remain closer to the non-dominated frontier.

Lastly, this chapter extends the concepts of dominance set and ranking by incorporating convex combinations of alternatives. By considering the convex hull of a set, more alternatives can be excluded than under the standard definition, yielding a convex dominant set. This refined set identifies alternatives that are not dominated by any convex combination of others.

Overall, the dominance framework introduced here offers a robust foundation for both analyzing individual alternatives and comparing the performance of alternative generation techniques.

6

Directionally Weighted Variables

This chapter introduces a new algorithm called Directionally Weighted Variables, which is a heuristic that generates near-optimal alternatives that are likely to be dominating. This Directionally Weighted Variables builds on and refines the Min/Max Variable method. By incorporating structural insights from the problem's formulation and the structure of the near-optimal solution space, the algorithm restricts the weight generation to alternatives that are more likely to be within the dominant set of all near-optimal alternatives.

6.1. Motivation

Traditionally, in MGA methods, alternatives are generated by solving a series of weighted optimization problems, where the weight vector is updated for each alternative. In the case of the Min/Max variable method, for each alternative, the values of the weights are sampled from the set: $-1, 0, 1$. This approach ensures a diverse sampling of the near-optimal space, but it lacks a mechanism to focus on regions that are more likely to contain dominating alternatives.

A key observation is that not all decision variables contribute the same way to the solution space with respect to dominance. In particular, there are three different options possible for any decision variable i, each influencing the objective function according to the sign of c_i:

1. **Positive:** $c_i > 0$.
2. **Negative:** $c_i < 0$.
3. **Zero:** $c_i = 0$, i.e., the variable does not appear in the objective.

Based on this observation, the possible values that are sampled when generating the weights for a weight vector can be reduced when the goal is to find non-dominated alternatives.

This can be more intuitively explained with an example. This example illustrates why the sign of the objective coefficient should influence the selection of weights. When comparing two alternatives that differ only by the sign of a single weight w_i, and where $c_i > 0$, assigning $w_i = 1$ pushes the decision variable x_i toward its lower bound, while $w_i = -1$ pushes it toward its upper bound. Because lower values of x_i yield better objective values when $c_i > 0$, the alternative with $w_i = 1$ dominates the one with $w_i = -1$. This suggests that exploring weights aligned with the sign of c_i is more likely to yield dominant alternatives. While this example considers a simplified case, it demonstrates the benefit of restricting weight values based on objective coefficients. This motivates the creation of a heuristic that searches the near-optimal solution space for alternatives that are more promising in terms of dominance.

6.2. Implementation

The Directionally Weighted Variables generate near-optimal alternatives by solving a series of weighted optimization problems. The key distinction of this method lies in how the weight vector is constructed: rather than being sampled uniformly from a fixed set, weights are selectively drawn based on the

structure of the objective function. This method is based on the Min/Max Variables method, which only considers possible values for the weights in the weight vector to be in the set $\{-1, 0, 1\}$. Limiting the possible weight values based on the coefficient in the objective function.

For each decision variable x_i, the algorithm considers the sign of its corresponding objective coefficient c_i and restricts the set of possible weights accordingly:

- If $c_i > 0$, the variable contributes positively to the objective. To prioritize minimizing this contribution, w_i is sampled from $\{0, 1\}$.
- If $c_i < 0$, the variable contributes negatively to the objective. To reduce its (negative) impact, w_i is sampled from $\{0, -1\}$.
- If $c_i = 0$, the variable does not influence the objective directly, so w_i is sampled from the full set $\{-1, 0, 1\}$.

The method generates in total n weight vectors, where n is the number of alternatives that should be generated. Together with the theory of MGA, the method minimizes the following objective at each iteration k

$$\mathbf{w^k} \cdot \mathbf{x}, \tag{6.1}$$

The whole method is also shown in Algorithm 1

This directional sampling introduces a problem-aware bias into the weight generation process, steering the search toward areas of the near-optimal solution space where dominance is more likely. Once a complete weight vector is generated, it is used to find an alternative in the near-optimal solution space problem. Similar to the other MGA methods, this process is repeated to construct a portfolio of alternatives, with each iteration generating a new weight vector.

The result is a more targeted exploration of the solution space, yielding alternatives that are not only near-optimal but also more decision-informative.

Algorithm 1 Directionally Weighted Variables

Require: Model (original optimization problem), n (number of alternatives), s (slack)
Ensure: Set of alternatives

Solve original model:
  $T \leftarrow$ solve(Model)
Define near-optimal model:
  Add constraint $f(x) \leq (1 + s)T$
Initialize alternative set $\mathcal{P} \leftarrow \{T\}$
for $k = 1$ to $k_{\max}$ **do**
  Generate weight vector w^k:
  for each variable i **do**
    if $c_i > 0$ **then**
      Sample $w_i \in \{0, 1\}$
    else if $c_i < 0$ **then**
      Sample $w_i \in \{0, -1\}$
    else
      Sample $w_i \in \{-1, 0, 1\}$
    end if
  end for
  Define auxiliary objective: minimize $w^k \cdot x$
  Solve near-optimal model with new objective:
    $x^k \leftarrow$ solve(Model)
  Add x^k to set of alternatives $\mathcal{P}$
end for
return $\mathcal{P}$

6.3. Illustrative Example

To illustrate the behavior of the Directionally Weighted Variables, we extend chapter 4 by analyzing the same illustrative example with the Directionally Weighted Variables method described above. This example allows us to demonstrate how directional weight sampling affects the generation of alternatives in a simple, visual setting.

Consider a problem with two decision variables, x and y, with associated objective coefficients $c_1 > 0$ and $c_2 > 0$. According to the algorithm's weight sampling strategy, both x and y adjust the sampled set to $\{0, 1\}$, encouraging the minimization of x and y.

When generating 5 alternatives for this illustrative problem, the resulting alternative set is visualized in Figure 6.1.

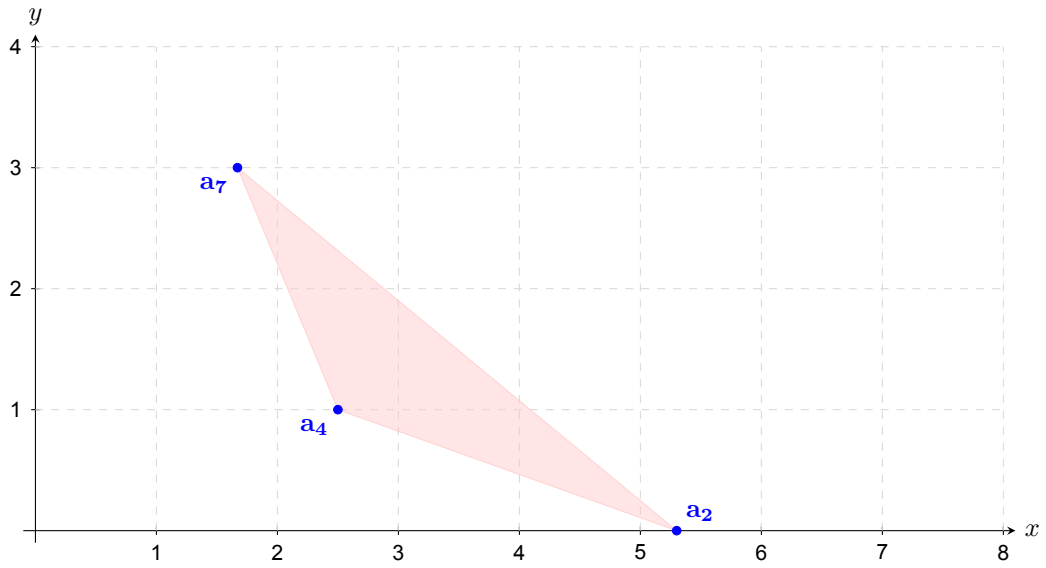


Figure 6.1: The corner points and convex hull found by using the Directionally Weighted Variables method when generating 5 alternatives.

When generating 20 alternatives for this illustrative problem, the resulting alternative set is visualized in Figure 6.2.

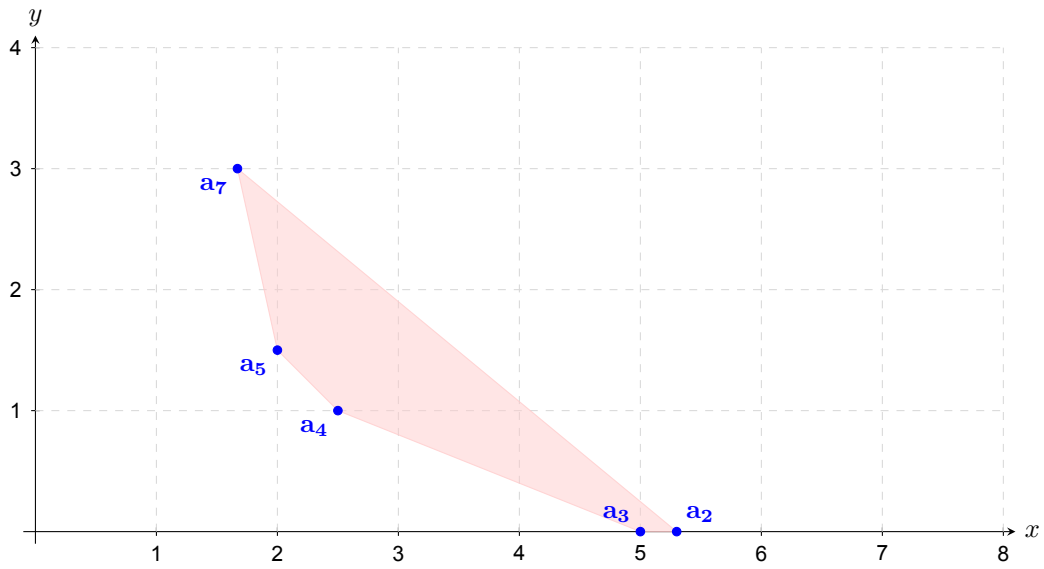


Figure 6.2: The corner points and convex hull found by using the Directionally Weighted Variables method when generating 20 alternatives.

Overall, the results illustrate that while the heuristic does not guarantee only finding alternatives within the dominant set, it finds the most decision-relevant regions of the near-optimal space of this illustrative problem, showing the promise of limiting the weight vectors to explore only certain regions of the near-optimal solution space.

6.4. Summary

This chapter introduced the Directionally Weighted Variables, a heuristic approach designed to generate near-optimal alternatives with a higher likelihood of dominance. Building upon the Min/Max Variable method, this algorithm refines the alternative generation process by using the sign of the objective function coefficients to inform weight vector construction. This directional bias leads to a more focused exploration of the near-optimal solution space, targeting regions that are more decision-relevant.

The illustrative 2D example highlighted how the algorithm guides the search toward more promising areas. While the approach does not guarantee only alternatives within the dominant set, it offers a practical and intuitive enhancement over traditional MGA methods by better aligning the weight generation process with problem structure.

In the following chapters, this algorithm will be evaluated more extensively using larger and more complex models. These experiments will explore the effectiveness and scalability of the Directionally Weighted Variables in high-dimensional settings, further validating its potential for real-world decision support.

7

Setup

This chapter outlines the experimental framework used to evaluate methods for generating high-quality, non-dominated, near-optimal alternatives. The focus is on understanding how different strategies for guiding the search, via weight vector generation, influence the dominance of alternative investment alternatives. This chapter clearly defines the model used for validation, establishes a clear experimental design to test the effectiveness of the proposed method, and discusses the evaluation of the findings.

The Generation Expansion Planning (GEP) model is used as the core optimization problem. Its structure, decision variables, and constraints are well-suited for exploring near-optimality, as multiple investment configurations can yield similar costs but differ significantly in structure. This makes the GEP model particularly valuable for examining trade-offs between decision variables, since these variables reflect the inherent flexibility in long-term capacity planning and highlight how small deviations from the optimal solution can have meaningful implications for system design, policy, and investment strategies.

The chapter is organized into five main sections. First, the GEP model is defined in detail, including all sets, parameters, and decision variables, followed by the mathematical formulation. Second, two model instances are introduced: a full-scale European model and a simplified version used for controlled analysis. Third, the algorithmic workflow is described—how alternatives are generated, filtered, and evaluated using different weight vector strategies. Fourth, this chapter shows how the dominance ranking is calculated. Lastly, the experimental setup is detailed, including solver configuration, uniqueness thresholds, and performance metrics.

7.1. Model

The model that is used to test the idea of domination is the Generation Expansion Planning (GEP) problem. GEP is a model that aims to minimize the economic cost of installing power plants to meet the expected demand of a given planning horizon. The objectives of GEP are fourfold [25]:

- Identify what types of power generation technologies should be added to the grid.
- Determine how much capacity of each generation type should be installed.
- Decide where the power plants should be located within the network.
- Establish when each power plant should be implemented.

There are many ways to implement the GEP model, each tailored to specific needs or contexts. These implementations may vary in planning horizons, network topology, and the inclusion of real-world constraints such as environmental regulations, fuel prices, or transmission limitations. Additionally, models can differ in how they handle uncertainty (e.g., through deterministic or stochastic approaches), as well as in whether they assume centralized or decentralized decision-making. Depending on their focus, the GEP model can incorporate various technical, economic, and environmental factors to more accurately reflect real-world energy systems.

GEP offers decision-makers a strong setting for flexible and robust long-term planning. Because it combines discrete investment choices with complex system constraints, the GEP model naturally exposes a landscape of near-optimal alternatives that reveal markedly different investment structures despite similar costs. This makes GEP well-suited for exploring near-optimal alternatives, which highlight trade-offs in cost, technological diversity, geographic distribution, and environmental impact, supporting more informed and resilient energy policy decisions [27].

This thesis focuses on a simplified, deterministic version of the GEP model. In this version, all investment decisions are made at the start of the planning horizon. As a result, the model does not address the timing of power plant construction, one of the original four GEP objectives. Instead, the goal of this simplified GEP is to minimize both investment and operational costs across the entire planning period.

Investment costs correspond to the expenses associated with building new power plants and directly address the first three goals of the GEP model: technology type, amount, and location. Operational costs represent the expenses required to run the power system once the investments are made. These consist of the cost of energy production, penalties for unmet demand, and any other costs associated with keeping the system functional.

Importantly, the GEP model is by design a doubly minimization problem. First, at the investment level, the model seeks the least-cost portfolio of technologies, capacities, and locations. Second, conditional on this investment portfolio, the operational subproblem minimizes the cost of dispatching the system to meet demand under technical and system constraints. In practice, these two problems are solved jointly in a single integrated optimization, but conceptually, the GEP embeds one minimization (operations) within another (investment), making it a hierarchical problem by nature.

7.1.1. Model Components

The model is defined using three categories: sets, parameters, and decision variables. Each component plays a role in formulating the GEP optimization problem. The sets describe the structure of the energy system (such as locations, technologies, time periods), the parameters define system-specific characteristics (such as demand, cost, and availability), and the variables represent the decisions to be optimized (such as how much to invest or produce).

Sets

The sets Table 7.1 define the core indexing structures used throughout the model. These consist of the locations or nodes (N), available generation technologies (G), and their combinations (NG) that indicate where a technology can be deployed. The model also considers discrete time steps (T) for temporal analysis and a set of bi-directional transmission lines (L) that connect different nodes. In this simplified model, the transmission lines do not lose any energy.

Table 7.1: Model Sets

Symbol	Description	Index
N	Locations (nodes)	n
G	Generation technologies	g
$NG \subseteq N \times G$	Technology-location availability pairs	(n, g)
T	Time steps	t
$L \subseteq N \times N$	Bi-directional transmission lines	$l = (n, n')$

Parameters

The parameters Table 7.2 outline the fixed input values that define system characteristics and constraints. This consist of demand ($D_{n,t}$), technology availability ($A_{n,g,t}$), investment costs $I_{n,g}$, operational costs $V_{n,g}$), technical specifications like unit capacity ($U_{n,g}$) and ramping rate ($R_{n,g}$), as well as transmission constraints (L_l^{exp}, L_l^{imp}). Additionally, the model accounts for the value of lost load (V^{loss}), which penalizes unmet demand.

Table 7.2: Model Parameters

Symbol	Description	Unit	Domain
$D_{n,t}$	Demand node n and time t	MW	$\mathbb{R}_+$
$A_{n,g,t}$	Availability of technology g at node n at time t	1/unit	$[0,1]$
$I_{n,g}$	Annual investment cost for technology g at node n	EUR/MW	$\mathbb{R}_+$
$V_{n,g}$	Hourly operational cost for technology g at node n	EUR/MWh	$\mathbb{R}_+$
$U_{n,g}$	Maximum capacity per unit of technology g at node n	MW/unit	$\mathbb{R}_+$
$R_{n,g}$	Ramping rate for technology g at node n	1/unit	$[0,1]$
L_l^{exp}	Export capacity of transmission line l	MW	$\mathbb{R}_+$
L_l^{imp}	Import capacity of transmission line l	MW	$\mathbb{R}_+$
V^{loss}	Value of lost load	EUR/MWh	$\mathbb{R}_+$

Decision Variables

The decision variables Table 7.3 present the optimization outputs that the model will determine. These consist of total costs, both investment (c^{inv}) and operational (c^{op}), as well as investment decisions ($i_{n,g}$), production levels ($p_{n,g,t}$), transmission flows ($f_{l,t}$), and any unmet demand ($p_{n,t}^{\text{loss}}$). These variables are constrained by the parameters and are optimized to minimize system costs while satisfying technical and demand requirements. The model supports solving an LP relaxation of the problem, changing the domain of investment decision to the positive real numbers.

Table 7.3: Decision Variables

Symbol	Description	Unit	Domain
c^{inv}	Total investment cost	EUR	$\mathbb{R}_+$
c^{op}	Total operational cost	EUR	$\mathbb{R}_+$
$i_{n,g}$	Number of technology units g invested at node n	unit	$\mathbb{Z}_+$ (or $\mathbb{R}_+$ if relaxed)
$p_{n,g,t}$	Production of technology g at node n at time t	MW	$\mathbb{R}_+$
$f_{l,t}$	Flow in transmission line l at time t	MW	$\mathbb{R}$
$p_{n,t}^{\text{loss}}$	Unmet demand at node n and time t	MW	$\mathbb{R}_+$

7.1.2. Full Model Formulation

The complete formulation of the GEP model is structured into four main components: the objective function, the demand balance constraint, the generation capacity constraint, and the ramping constraints. Together, these components capture both the investment and operational aspects of power system planning over a defined time horizon.

Objective Function

The objective of the model is to minimize the total system cost, which consists of two components: investment costs and operational costs, described in

$$\min\left(c^{\text{inv}} + c^{\text{op}}\right). \tag{7.1}$$

Investment costs are incurred upfront when installing generation technologies and are based on the number of units, their capacity, and associated capital costs, as shown in

$$c^{\text{inv}} = \sum_{(n,g)\in NG} I_{n,g} \cdot U_{n,g} \cdot i_{n,g}. \tag{7.2}$$

Operational costs are accumulated over the planning horizon and consist of variable generation costs as well as penalties for any unmet demand, referred to as lost load, as shown in

$$c^{\text{op}} = \sum_{(n,g)\in NG} \sum_{t\in T} V_{n,g} \cdot p_{n,g,t} + \sum_{n\in N} \sum_{t\in T} V^{\text{loss}} \cdot p_{n,t}^{\text{loss}}. \tag{7.3}$$

Demand Balance

The demand balance constraint ensures that, at every node and for each time step, electricity demand is met either through local generation, net imports and exports from connected nodes via transmission lines, or if necessary, by allowing unmet demand (lost load). This constraint enforces energy conservation within the network and serves as the central operational requirement of the model, as described in

$$D_{n,t} = \sum_{g \in G:(n,g) \in NG} p_{n,g,t} + \sum_{n' | (n',n) \in L} f_{(n',n),t} - \sum_{n' | (n,n') \in L} f_{(n,n'),t} + p^{\text{loss}}_{n,t} \quad \forall n \in N, \forall t \in T. \tag{7.4}$$

Generation Capacity

The generation capacity constraint limits the amount of electricity that each technology can produce at a given node and time step. This upper bound depends on the number of units installed, the per-unit capacity, and the availability of the technology, which can vary over time (e.g., due to weather conditions for renewables). This constraint ensures that generation levels remain physically realistic and do not exceed the technically feasible output of the installed infrastructure. The constraint is described in

$$p_{n,g,t} \leq A_{n,g,t} \cdot U_{n,g} \cdot i_{n,g} \quad \forall (n,g) \in NG, \forall t \in T. \tag{7.5}$$

Ramping Constraints

Lastly, ramping constraints limit the rate at which power generation can increase or decrease between consecutive time steps. These constraints reflect the operational flexibility of each technology, ensuring that production changes remain within feasible bounds determined by the ramping rate and installed capacity. By enforcing these dynamics, the model captures important temporal limitations on how quickly different technologies can respond to fluctuations in demand or system conditions. The ramping constraint for ramping up is described in

$$p_{n,g,t} - p_{n,g,t-1} \leq R_{n,g} \cdot U_{n,g} \cdot i_{n,g} \quad \forall (n,g) \in NG, \forall t \in T \setminus \{1\}, \tag{7.6}$$

and ramping down in

$$p_{n,g,t} - p_{n,g,t-1} \geq -R_{n,g} \cdot U_{n,g} \cdot i_{n,g} \quad \forall (n,g) \in NG, \forall t \in T \setminus \{1\}. \tag{7.7}$$

7.2. Model instance

To evaluate the proposed dominance-aware MGA algorithm, we conduct experiments on two variants of a European dataset, containing hourly data for each parameter for a full year. The two variants that are considered in this thesis are:

1. The full European dataset, consisting of a more realistic, high-dimensional planning problem.
2. A simplified European dataset, with a reduced problem nodes and technologies for tractability and controlled analysis.

These two problem instances allow us to validate both the behavior and scalability of the proposed algorithm under different levels of complexity.

7.2.1. Full Model

The full dataset contains 20 different countries, all represented by nodes:

- Austria
- Belgium
- Balkan countries
- Baltic countries
- Czech Republic
- Denmark
- Finland
- France
- Germany
- Ireland
- Italy
- Netherlands
- Norway
- Poland
- Portugal
- Slovakia
- Spain
- Sweden
- Switzerland
- United Kingdom

Each node is connected to at least one other node by transmission lines. Furthermore, the model also supports eight different technologies:

- Coal
- Gas
- Lignite
- Nuclear
- Oil
- Solar energy
- Offshore wind
- Onshore wind

Together, they form 106 decision variables, as not all combinations between the countries and technologies are valid. Experiments on this model test the algorithm's scalability and ability to find dominating alternatives in a high-dimensional space.

7.2.2. Simplified Model

In the simplified setup, the original European dataset is reduced to include only five countries:

- Belgium
- France
- Germany
- Netherlands
- United Kingdom

As well as only two different technologies:

- Gas
- Onshore wind

This results in only nine investment decision variables, as one country-technology pair, the Netherlands-gas pair, is infeasible. The model does not allow for any investments in gas in the Netherlands. The reduced dimensionality allows detailed insight into how weight vectors influence the generation of near-optimal alternatives as well as the dominance of those alternatives.

7.2.3. Focus on Investment Variables

Because the GEP model is, by design, a double minimization problem in both model variants, the experiments only generate alternatives with respect to the decision variables that decide on the investment cost. Alternatives are only generated with respect to investment costs because these decisions represent the long-term, irreversible choices in the GEP problem. Namely, which technology should be built, how much of each should be invested in, and where it should be invested. Operational costs, by contrast, are outcomes of these investment choices rather than strategic decisions. Focusing on the investment costs, therefore, highlights the feasible strategies within the near-optimal solution space. However, the alternatives, which still have both the operational costs and investment costs, are still limited by the addition of the near-optimal constraint when defining the near-optimal solution space.

Similarly, this double minimization alters how domination is evaluated. While the optimal solution minimizes total system cost (investment + operational), dominance is assessed solely on the investment variables, since these are the only dimensions along which alternatives are generated. Consequently, the cost-optimal solution often does not dominate many near-optimal alternatives, as it may rely on lower operational costs rather than strictly lower investment expenditures.

7.2.4. Near-optimal solution space

To formally define the near-optimal solution space of the GEP model, we introduce an additional constraint:

$$c^{inv} + c^{opt} \leq (1 + s)T \tag{7.8}$$

Where c^{inv} represents the investment cost, c^{opt} corresponds to the operating cost, and T denotes the cost in the optimal solution. The parameter s specifies the allowable slack, thereby controlling how much the sum of all costs can deviate from the optimal solution.

Within this constrained near-optimal space, the objective function at iteration k can be expressed as:

$$min(w^k \cdot c^{inv}) \tag{7.9}$$

Where w^k is an iteration-dependent weighting vector at iteration k that depends based the method used for generating alternatives.

7.3. Experimental Workflow

This section outlines the experimental workflow, including the model-solving steps, comparison methods, uniqueness criteria, solver settings, and evaluation framework. The overarching goal of this chapter is to evaluate different strategies for generating high-quality, non-dominated, near-optimal alternatives. In particular, we aim to understand how varying approaches to weight vector generation influence the exploration of the feasible solution space and the dominance of the resulting investment alternatives. To this end, the following workflow establishes a structured and reproducible procedure for alternative generation, comparison, and evaluation.

7.3.1. General Procedure

Each experiment follows this structured process to generate and evaluate near-optimal alternatives:

1. Baseline Solution: Solve the baseline model to obtain the cost-optimal investment and operational plan, save the optimal solution found this way as a viable alternative in the set of alternatives.
2. Introduce Slack: Add a constraint to allow for near-optimality by permitting alternatives within a fixed optimality gap (typically 10%). This introduces flexibility to explore alternative investment plans.
3. Weight Vector Generation: Generate a new weight vector for the investment variables using one of the selected weight update methods.
4. Alternative Generation: Update and resolve the model using the new weight vector by minimizing the dot product of the weight vector and the variable vector. Add the found alternative to the set of alternatives.
5. Repeat: Continue generating alternatives by returning to Step 3 (Weight Vector Generation) until the desired number of alternatives is reached.
6. Time Tracking: Record the time required by each method to generate the full set of alternatives.

This process is repeated for each method described below.

7.3.2. Methods for Weight Vector Generation

The following six methods are compared to explore the space of near-optimal investment alternatives:

- HSJ
- Spores
- Min/Max Variables
- Random Vector
- Max-Distance
- Directionally Weighted Variables

Each method guides the search through the feasible space differently, aiming to produce meaningful alternatives.

7.3.3. Model and Solver Setup

To ensure feasibility and scalability across both small and large model variants, the following configuration is used:

- Solver: Gurobi (exact solver)
- Relaxation: Linear Programming (LP) relaxation is used to reduce runtime
- Optimality Gap (Slack): Set at 10% unless otherwise specified
- Equality Threshold: Set to $1.0e^{-4}$ unless otherwise specified
- Number of Alternatives per Method: Configurable based on the scenario

- Model Variants:
  - Simplified Model: 9 investment variables
  - Full Model: 106 investment variables
- Time Steps: Configurable based on the scenario. Investment costs are scaled according to the length of each time step relative to the expected lifetime of the asset. This ensures that long-lived investments, such as a power plant expected to last 50 years, contribute only the appropriate fraction of their total cost based on the number of time steps (e.g., 1/50th per year) rather than the full upfront cost.

After generating the same number of alternatives for each of the six methods, a combined set of alternatives is constructed. Dominance ranking section 5.4 is used to classify alternatives into the ranking sets ($D_1, D_2, \ldots$). In theory, this ranking should be based on the convex dominant set. However, this full theoretical framework is challenging to implement in practice.

To approximate the intended behavior, simplifications are made. The first simplification is that we assume that all alternatives are single corner points. This is assumed because having (two or more) optimal corner points (and thus the possibility to find an optimal point not in a corner) is extremely rare [6]. This assumption allows for sets not to include any convex combination of two other alternatives, making comparison between sets of alternatives possible. Secondly, due to the difficulty of generating all convex combinations of the alternatives and excluding alternatives dominated by any of such combinations being extremely difficult in high-dimensional space, the experiments will compare the set size of the dominant set and not the convex dominant set.

Due to the use of floating values, two alternatives are considered equal when each decision variable differs by at most this numerical threshold. If an alternative is considered equal to any other, it will not be included in the set of alternatives.

Performance is assessed based on both the quality (e.g., dominance rank) of the alternatives, as well as the runtime of finding the set of alternatives for each method.

7.3.4. Dominance Ranking

To assess the structural quality of the generated alternatives, each alternative is evaluated using dominance. For each experiment, all alternatives generated by each method are considered when determining dominance.

The ranking of alternatives is computed using the Fast Non-Dominated Sorting algorithm introduced in the NSGA-II framework [8]. This method efficiently partitions the set of alternatives into dominance sets:

- D_1: The set of all non-dominated alternatives.
- D_2: The set of alternatives dominated only by those in D_1.
- D_3: The set of alternatives dominated only by those in $D_1 \cup D_2$, and so on.

The use of the fast non-dominated sorting algorithm enables scalability to larger sets of high-dimensional alternatives while preserving the theoretical properties of dominance. It is particularly suitable for comparing the effectiveness of different weight vector generation strategies in producing high-quality alternatives. The method that contributed the most alternatives to the set D_1 is considered to perform best when comparing for dominance.

7.4. Summary

This chapter defines the experimental foundation for evaluating near-optimal alternatives in Generation Expansion Planning problems. The GEP model is described in full, along with its simplified and full-scale variants, each offering a different level of complexity.

A structured workflow is presented for generating alternatives within a predefined cost slack, using various methods for weight vector generation. Each alternative is filtered for uniqueness and evaluated

based on its dominance rank. Only investment variables are considered when assessing structural quality, allowing for a focused analysis on long-term planning decisions.

The chapter also specifies solver settings, LP relaxation assumptions, thresholds used across experiments, and the optimality gap. Together, these elements form a consistent framework for comparing different strategies to explore the near-optimal solution space effectively.

8

Results

The goal of this chapter is twofold. First, it evaluates the performance of the newly proposed Directionally Weighted Variables method, introduced in chapter 6. The chapter tests whether the heuristic delivers on its promise of efficiently guiding the search toward non-dominated alternatives. Second, it benchmarks this method against five established alternatives — HSJ, Spores, Min/Max Variables, Random Vector, and Max-Distance — on the Generation Expansion Planning (GEP) model introduced in chapter 7.

The motivation for this evaluation lies in both method development and practical application. From a methodological perspective, Directionally Weighted Variables is designed to improve the quality of generated alternatives by focusing on dominance, and it is crucial to assess whether it provides measurable advantages over existing approaches. From a practical perspective, applying all methods to a real-world, large-scale optimization problem allows us to compare their strengths and weaknesses in terms of the size of the set of non-dominated alternatives and computational efficiency.

8.1. Baseline Experiments

The first experiments are done as described in section 7.3. For these experiments, each method generates 1000 near-optimal alternatives per scenario. Experiments are performed for different time step configurations: 1, 10, 100, and 500 for the simplified model, and 1, 10, and 100 for the full model. The 500 time step configuration is omitted for the full model due to its significantly increased complexity, which makes such long time horizons computationally impractical. This setup allows us to evaluate the influence of temporal complexity and model size on the performance of each method, as well as their ability to generate dominant alternatives.

8.1.1. Simplified Model

To better understand the fundamental behavior of each method under controlled conditions, this section first evaluates their performance on a simplified GEP model.

Before presenting the results, it is important to outline a few expectations based on the structure of the problem. First, we expect that as the number of time steps increases, each method will be able to generate more unique alternatives. This is because additional time steps create more flexibility in operational cost scheduling, which in turn supports a larger variety of investment costs in near-optimal alternatives.

Second, we anticipate that Directionally Weighted Variables, which is designed to generate non-dominated alternatives, to contribute a high proportion of alternatives it generates to the dominant set. Similarly, as seen in the toy example discussed in chapter 4, it is expected that HSJ and Spores will also contribute a large proportion of their generated alternatives to the dominant set. In contrast, the other methods are expected to contribute a smaller proportion of their generated alternatives to the dominant set.

Third, the deterministic methods (HSJ, Spores, and Max-Distance), which are constrained by their

update rules, are expected to produce fewer alternatives overall.

Fourth, because there are only nine decision variables, HSJ and Directionally Weighted Variables only produce $2^9 = 512$ distinct weight vectors. This means that both methods are expected to generate at most 512 unique alternatives.

Lastly, we expect that each method, except Max-Distance, has similar computational costs, with Max-Distance having a significantly increased runtime, due to its non-linearity in its objective function.

Dominance Rank Distribution

The results of the dominance rank of the simplified model for the four different time step configurations are shown in Table 8.1 through Table 8.4.

Table 8.1: Dominance rank distribution for the simplified model with 1 time step

Method	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10+}
HSJ	2	1	0	0	0	0	0	0	0	0
Spores	11	3	0	0	0	0	0	0	0	0
Max-Distance	128	23	13	12	14	8	5	3	3	3
Min/Max Variables	286	202	118	34	28	5	7	6	5	10
Random Vector	120	212	161	82	43	15	8	5	6	8
Directionally Weighted Variables	609	44	3	0	0	0	0	0	0	0

Table 8.2: Dominance rank distribution for the simplified model with 10 time steps

Method	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10+}
HSJ	7	0	0	0	0	0	0	0	0	0
Spores	13	2	0	0	0	0	0	0	0	0
Max-Distance	70	22	30	25	5	9	6	4	1	8
Min/Max Variables	439	141	82	37	19	13	10	4	7	21
Random Vector	266	194	94	42	17	12	7	3	2	15
Directionally Weighted Variables	813	31	10	3	1	0	0	0	0	1

Table 8.3: Dominance rank distribution for the simplified model with 100 time steps

Method	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10+}
HSJ	173	1	0	0	0	0	0	0	0	0
Spores	45	1	0	0	0	0	0	0	0	0
Max-Distance	168	94	68	15	7	5	5	5	5	29
Min/Max Variables	533	113	52	26	17	5	1	0	0	26
Random Vector	615	74	51	19	11	3	0	0	0	14
Directionally Weighted Variables	877	30	3	1	0	0	0	0	0	1

Table 8.4: Dominance rank distribution for the simplified model with 500 time steps

Method	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10+}
HSJ	775	5	0	0	0	0	0	0	0	0
Spores	57	1	0	0	0	0	0	0	0	0
Max-Distance	89	16	18	55	9	6	1	3	3	18
Min/Max Variables	413	114	55	29	11	9	12	8	3	19
Random Vector	566	35	37	23	10	3	2	2	2	9
Directionally Weighted Variables	718	49	10	1	0	0	0	0	0	0

Runtime Comparison

In this part, the results of the runtime of the simplified model for the four different time step configurations are shown in Table 8.5 and Figure 8.1.

Table 8.5: Runtime (seconds) of each method to generate 1000 alternatives for the simplified model

Time steps	1	10	100	500
HSJ	2.6	4.3	10.0	56.6
Spores	2.7	4.6	13.0	44.7
Max-Distance	44.3	118.6	1105.1	8513.6
Min/Max Variables	2.9	5.6	20.7	234.7
Random Vector	2.8	5.9	25.6	367.6
Directionally Weighted Variables	2.7	5.2	17.4	113.4

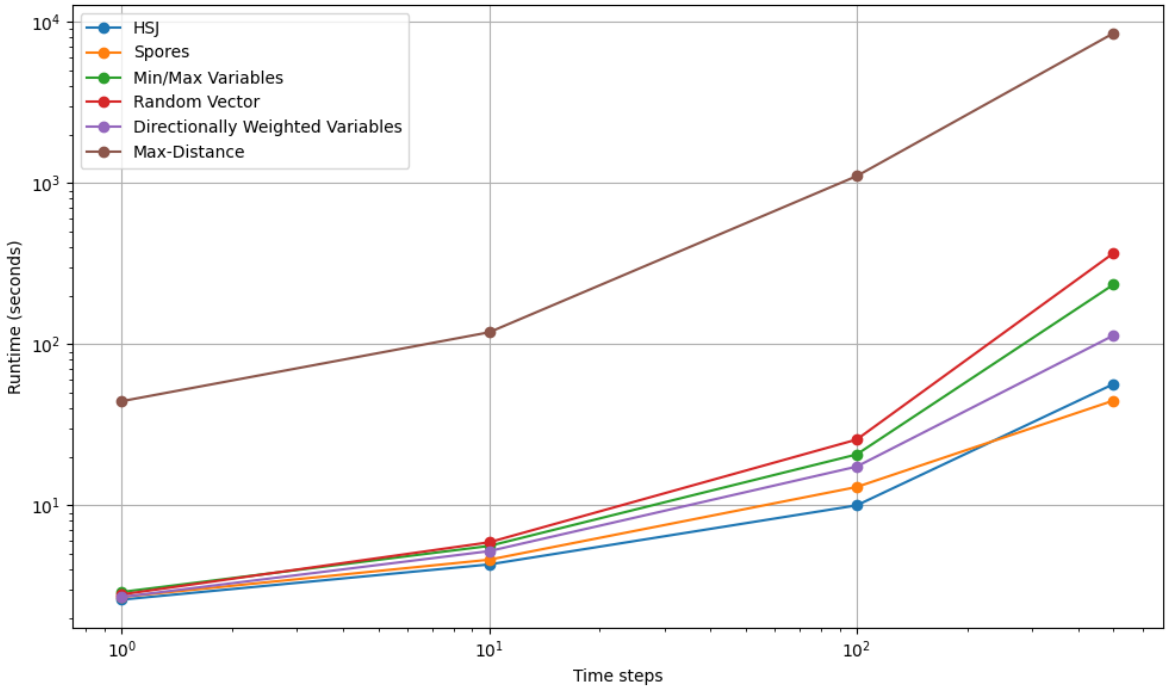


Figure 8.1: Log–log plot of runtime for the simplified model when generating 1000 alternatives under different time-step configurations.

Discussion

The results of the baseline experiments on the simplified GEP model reveal clear trade-offs between the dominance of the generated alternatives and the computational efficiency of each method.

First, it was expected that, given more time steps, each method would generate more unique alternatives. This is true for the methods Spores and HSJ. However, the other methods produce the most unique alternatives when there are 100 time steps, and produce fewer unique alternatives when the time steps increase to 500 time steps.

Second, across all time step configurations, the alternatives generated with either Directionally Weighted Variables, HSJ, and Spores, consistently are among the alternatives in the dominant set (D_1). This demonstrates their strong ability to identify non-dominated alternatives. This confirms that in lower-dimensional settings, the use of these methods guides the search toward superior regions of the solution space. As expected, the other methods, Max-Distance, Min/Max Variables, and Random Vector, contribute a lower proportion of their alternatives to the dominant set.

Third, as expected, methods such as HSJ, Spores, and Max-Distance generally generate significantly fewer unique alternatives across most configurations. In contrast, Directionally Weighted Variables, Random Vector, and Min/Max Variables produce significantly more unique alternatives across all configurations. An exception occurs in the case of HSJ with 500 time steps, where it produces several unique alternatives, that is comparable to the stochastic methods. Further research on how well the alternatives generated by each method represent the near-optimal solution space should be investigated further.

Fourth, the fact that HSJ and Directionally Weighted Variables were able to generate more than 512 distinct alternatives suggests that some core concept of generating alternatives is missed. This should be investigated further.

Lastly, when considering computational burden, as expected, Max-Distance performs the worst by a significant margin, especially as the number of time steps increases. Surprisingly, HSJ and Spores emerge as the most computationally efficient methods, consistently achieving low runtimes across all scenarios. Maybe even more unexpected is the performance of Directionally Weighted Variables, which, for low dimensions, seems to be computationally more efficient than the other stochastic methods. The lower runtimes of the deterministic methods warrant more research to find out why these methods perform computationally better than others.

In low-dimensional problems, this experiment shows particularly promising results, indicating that the Directionally Weighted Variables method is not only the most effective in guiding the search toward regions with non-dominated alternatives, by generating many non-dominated alternatives, but Directionally Weighted Variables also has lower computation costs compared to other stochastic methods. For exploratory purposes or where runtime is severely constrained, HSJ and Spores present a smaller set of non-dominated alternatives compared to Directionally Weighted Variables.

8.1.2. Full Model

To investigate the difference of each method in a real-world environment, this section evaluates the performance of the full GEP model.

Before presenting the results, it is important to outline a few expectations based on the structure of the problem. First, given the large number of decision variables and the definition of dominance, which includes all these decision variables, it is expected that the vast majority—if not all—–alternatives will fall within the dominant set. This is likely due to the sheer size of the solution space: even considering each investment decision variable as binary (active or inactive), the total number of possible combinations is on the order of 2^{106}, an astronomically large number compared to the 1000 alternatives generated by each method in this experiment. This disparity underscores that only a minuscule fraction of the full decision space can be explored, and thus, most generated alternatives are unlikely to dominate any other, purely by chance.

Second, deterministic methods are generally known to generate fewer unique alternatives compared to stochastic approaches. As such, we anticipate that HSJ, Spores, and Max-Distance methods will yield a smaller number of unique alternatives than the stochastic methods.

Last, based on prior results, HSJ and Spores have demonstrated strong performance in terms of computational efficiency. We expect this trend to continue in the full model. Conversely, Max-Distance is expected to incur the highest computational burden due to its distance-based search mechanism. Among the stochastic methods, we anticipate that Directionally Weighted Variables will continue to outperform Min/Max Variables and Random Vector in both alternative quality and runtime efficiency.

Important to note is that these results do not include Max-Distance, as generating just 1 alternative for 1 time step already took more than an hour.

Dominance Rank Distribution

In this part, the results of the dominance rank of the full model for the four different time step configurations are shown in Table 8.6 through Table 8.8.

Table 8.6: Dominance rank distribution for the full model with 1 time steps

Method	D_1
HSJ	8
Spores	866
Min/Max	1000
Random Vector	1000
Domination Vector	1000

Table 8.7: Dominance rank distribution for the full model with 10 time steps

Method	D_1
HSJ	999
Spores	993
Min/Max	1000
Random Vector	1000
Domination Vector	1000

Table 8.8: Dominance rank distribution for the full model with 100 time steps

Method	D_1
HSJ	893
Spores	1000
Min/Max	1000
Random Vector	1000
Domination Vector	1000

Runtime Comparison

In this part, the results of the runtime of the full model for the four different time step configurations are shown in Table 8.9 and Figure 8.2. Note that Max-Distance did not finish for any configuration.

Table 8.9: Runtime (seconds) of each method to generate 1000 alternatives for the full model

Time steps	1	10	100
HSJ	5.4	35.3	1644.0
Spores	3.5	11.6	116.5
Max-Distance	-	-	-
Min/Max Variables	3.8	22.7	702.6
Random Vector	3.9	22.4	691.6
Directionally Weighted Variables	3.7	30.0	1521.5

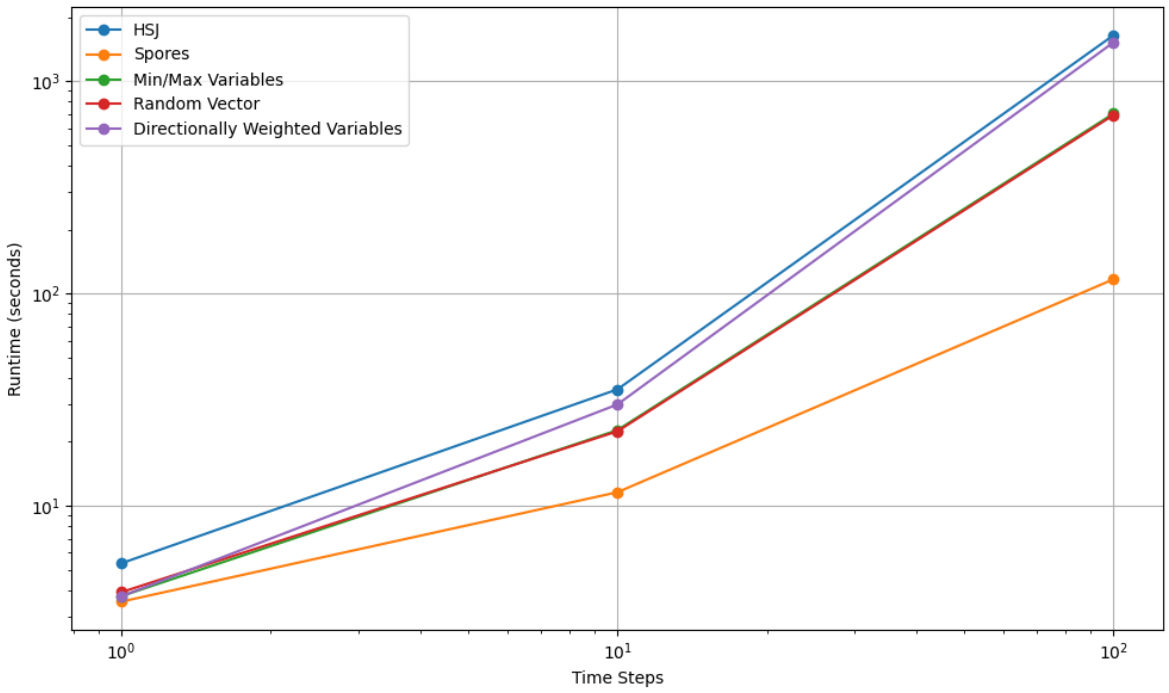


Figure 8.2: Log–log plot of runtime for the full model when generating 1000 alternatives under different time-step configurations

Discussion

The dominance rank distributions for the full GEP model show that all methods consistently produce alternatives exclusively within the dominant set (D_1). This is expected, given the model's complexity with 106 decision variables, 1000 alternatives generated per method are unlikely to sample enough of the near-optimal solution space to find both a pair of alternatives for which one of them dominates the other.

Another interesting observation is that HSJ and Spores do not generate 1000 unique alternatives. This aligns with expectations, as these methods are known for limited exploration of the near-optimal space.

In terms of computational efficiency, Spores remains the most time-efficient method across all tested time steps and is significantly more efficient than any other method. This runtime is followed by the Min/Max Variables and Random Vector approaches, which both take a similar amount of time. While HSJ performs well for smaller models, its runtime increases substantially at more difficult models, indicating potential scalability challenges. The Directionally Weighted Variables method, although effective at producing dominant alternatives, exhibits higher runtimes in the full model. More research is needed to compare why the runtime of Directionally Weighted Vectors performs worse in comparison to other random methods for the full model.

8.1.3. Conclusion

These results show great use of generating alternatives with respect to dominance in low-dimensional settings. These results also highlight a limitation of the dominance metric itself: dominance seems to be a weaker discriminator in high-dimensional spaces. As such, dominance alone may not be sufficient for evaluating the quality of alternatives in complex, high-dimensional problems and should be complemented by other metrics such as coverage of the near-optimal solution space or the robustness of the alternatives.

Overall, these experiments demonstrate that no single method is universally optimal across all scenarios. Directionally Weighted Variables offer the best quality in terms of dominance. Spores are the most efficient computationally, and Min/Max Variables and Random Vector methods provide a balance between exploration and efficiency. The choice of method should thus be guided by the specific priorities of the application—dominance quality or computational budget.

8.2. Runtime Analysis of Alternative Generation Methods

One of the key observations from the previous experiments is that the computational performance of the alternative generation methods varies significantly. This variation can be summarized in three main findings:

1. Spores consistently outperforms all other methods in terms of runtime, both in the simplified and full models.
2. Max-Distance performs the worst, with runtimes so high that generating even a single alternative in the full model took over an hour.
3. HSJ and Directionally Weighted Variables are more efficient in the simplified model, whereas Min/Max Variables and Random Vector methods perform better in the full model.

This section explores the underlying reasons behind these differences in runtime. We propose three hypotheses that may explain the observed disparities:

1. Impact of the Near-Optimal Solution Space: The shape and structure of the near-optimal solution space may influence the difficulty of solving subproblems. To investigate this, we compare runtimes with a similar configuration and only differing optimality gap. If a wider or narrower gap significantly affects runtime, it would suggest that the shape of the near-optimal solution space plays a role.
2. Exploration of Different Regions in the Near-Optimal Solution Space: Some methods may naturally explore regions of the solution space that are computationally easier or harder to solve. To test this, we introduce a variant of the Directionally Weighted Variables method that is specifically designed to target dominated alternatives rather than non-dominated ones. If this inverted search direction leads to different runtimes, it could indicate that certain regions of the near-optimal solution space are inherently more difficult to explore.
3. Role of Objective Function vs. Model Updating: Since all methods differ only in how the objective function is formulated for each iteration, the observed runtime differences may stem from how these objectives interact with the solver's optimization process. To isolate this effect, we rerun experiments in which the model is rebuilt from scratch at each iteration. This allows us to determine whether the speed of some methods (like Spores) is due to the structure of the objective function itself or from the way the model is incrementally updated.

By systematically evaluating these three hypotheses, we aim to better understand the computational characteristics of each method and provide guidance for their practical use in different model settings. In addition to runtime, we also report significant changes in dominance behavior across methods and configurations, offering further insight into how different strategies influence both the quality and computational cost of the generated alternatives.

8.2.1. Impact of Near-Optimal Solution Space

The structure of the near-optimal solution space can play a critical role in determining the efficiency and behavior of alternative generation methods. One hypothesis is that the computational effort required to generate alternatives is influenced by how easily the solver can locate feasible near-optimal alternatives within a given optimality gap.

This leads to the following hypothesis: *Changing the optimality gap and thereby the near-optimal solution space impacts the computational efficiency of each method differently*

To evaluate this hypothesis, the optimality gap is varied, which ensures that the near-optimal solution space is varied as well. The results show how a differing optimality gap influences computational costs for the different methods. This analysis helps reveal whether the shape of the near-optimal space contributes to the observed performance differences among the methods. Whereas in the existing literature the optimality gap is typically restricted to values between 0.05 and 0.2, the present experiment employs a wider range of values, $0.001, 0.01, 0.1,$ and 1, to more thoroughly explore the boundaries of the near-optimal space and to test the patterns that might emerge outside typical values.

In this experiment, for the full model 10 time steps are used, and for the simplified model 100 time steps are used.

The runtime with respect to the optimality gap of both the full model (see Figure 8.3) and simplified model (see Figure 8.4) is shown.

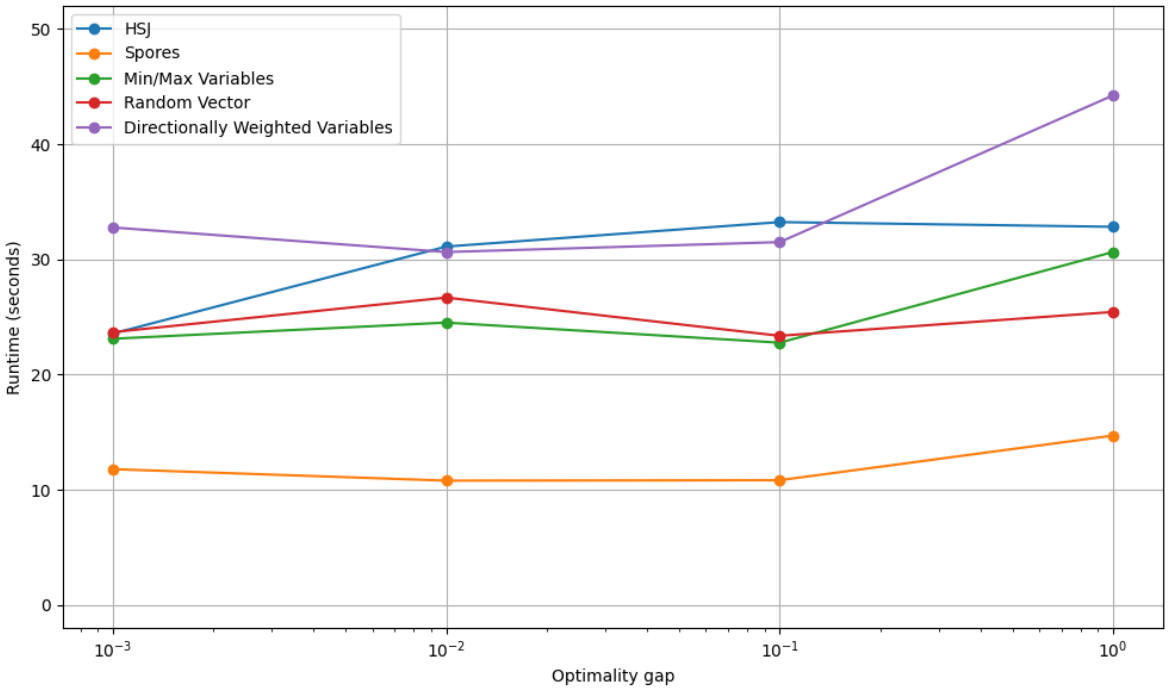


Figure 8.3: Runtime versus log-scaled optimality gap for the full model with 10 time steps

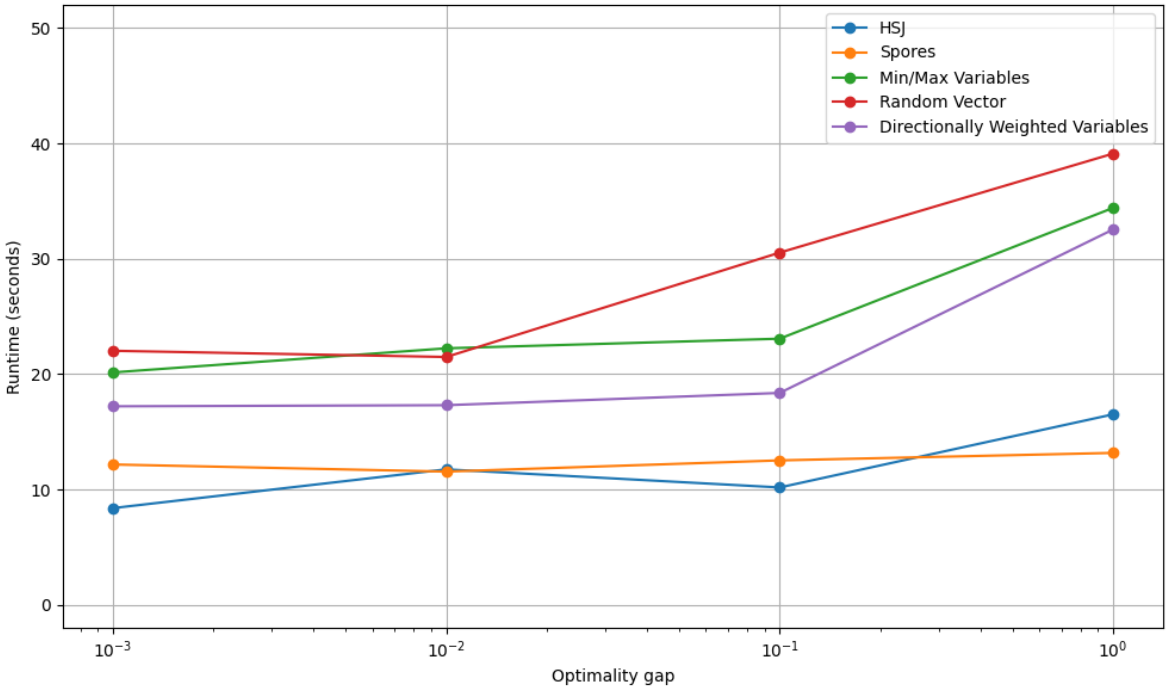


Figure 8.4: Runtime versus log-scaled optimality gap for the simplified model with 100 time steps

The results indicate that the optimality gap exerts a measurable influence on the runtime of each method. For both models, computational efficiency generally increases with a larger optimality gap. However,

its impact does not appear to differ substantially across the differing methods. While a relationship between optimality gap and runtime is evident, this relationship was not quantified numerically. This decision is justified by the fact that the experiments deliberately explored optimality gaps well beyond those typically encountered in practice, and the observed differences in runtime across this extended range remain comparatively modest.

An additional noteworthy observation arises with respect to the size of the dominant set. For the simplified model and for the HSJ method, it seems that an increase in the optimality gap heavily influences the number of alternatives within the dominant set, see Table 8.10. For the full model and for the Spores method, it seems that for an optimality gap of 0.1, it only contributes about half of its generated alternatives compared to all other optimality gaps, see Table 8.11. For all other model and method combinations, there was almost no difference between the differing optimality gaps. Although further investigation would be required to determine the precise cause, this finding highlights that adjustments to the optimality gap can lead to qualitatively different outcomes of the size of the dominant set.

Table 8.10: Alternatives contributed to the dominant set by HSJ in the simplified model

Optimality gap	0.001	0.01	0.1	1
HSJ	2	13	173	918

Table 8.11: Alternatives contributed to the dominant set by Spores in the full model

Optimality gap	0.001	0.01	0.1	1
HSJ	988	488	993	988

8.2.2. Exploration of Different Regions in the Near-Optimal Solution Space

From the previous results, we observed that Min/Max Variables and Random Vector methods perform computationally better than Directionally Weighted Variables and HSJ in the full model, but computationally worse in the simplified model. It is also observed that from all alternatives generated by both HSJ and Directionally Weighted Variables, these generated alternatives are mostly performing well with respect to dominance. In contrast, the alternatives generated by Min/Max Variables and Random Vector have no such guarantee.

This leads to the following hypothesis: *The region explored by Directionally Weighted Variables in the simplified model is relatively easy to explore, while in the full model, the region explored by Directionally Weighted Variables is relatively difficult to explore.*

To test this hypothesis, an altered Variables method is created, which generates all possible weight vectors that can be generated by the Min/Max Variables method, but excludes all the weight vectors that could be generated by the Directionally Weighted Variables method. To confirm the hypothesis, this method should perform computationally worse than Min/Max and Directionally Weighted Variables in the simplified model and better in the full model.

In both model instances, the Min/Max Variables are expected to yield the median results. For the simplified model, the Directionally Weighted Variables should perform the best computationally, since they explore a region that is not computationally intensive. By contrast, in the simplified model, the Altered Variables method should perform the worst, as it deliberately avoids exploring the less computationally demanding region. However, this effect is expected to reverse in the full model, where the Altered Variables approach is anticipated to be computationally more efficient than the Directionally Weighted Variables.

When comparing the runtimes of these methods for the simplified and full models, we can clearly see that this hypothesis is correct, see Table 8.12. With this observation, it can be concluded that most of the runtime differences between the runtime of Min/Max Variables and Directionally Weighted Variables are directly affected by the region of the near-optimal solution space that they explore. This shows that any

runtime advantage gained from either method is purely based on the underlying near-optimal solution space.

Table 8.12: Runtime (s) of each method to find 1000 alternatives

	Simplified model 500 time steps	Full model 100 time steps
Altered Variables method	247.1764578	762.5362692
Min/Max Variables	236.038391	781.6368696
Directionally Weighted Variables	113.7205672	1596.3874411

Another key observation is that this altered method, essentially an inverse of the heuristic used in Directionally Weighted Variables, finds fewer non-dominated alternatives in the simplified model than the Min/Max Variables method, see Table 8.13. This provides further evidence that the regions explored by the Directionally Weighted Variables method are indeed rich in non-dominated alternatives, reinforcing its value in guiding the search process effectively. However, as expected, the dominance metric does not give any insights into the full model, see Table 8.14.

Table 8.13: Dominance rank distribution for the simplified model with 500 time steps

Method	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10+}
HSJ	773	7	0	0	0	0	0	0	0	0
Spores	57	1	0	0	0	0	0	0	0	0
Min/Max Variables	398	108	67	40	21	17	12	11	9	18
Random Vector	588	11	26	21	18	15	5	4	3	11
Directionally Weighted Variables	709	62	21	2	1	0	0	0	0	0
Altered Variables method	353	124	50	49	32	12	12	8	9	20

Table 8.14: Dominance rank distribution for the full model with 100 time steps

Method	D_1
HSJ	893
Spores	1000
Min/Max Variables	1000
Random Vector	1000
Directionally Weighted Variables	1000
Altered Variables method	1000

8.2.3. Role of Objective Function vs. Model Updating

The only difference between the different methods is the updating rule that each of them uses. This raises an important question: *Are these runtime differences primarily driven by the regions of the near-optimal solution space that each method explores, or by how the optimization model is updated between iterations?*

To investigate this, we alter step 4 of the general procedure 7.3.1 where, instead of updating and solving the model, the model is rebuilt from scratch and then solved. Normally, when the model updates the objective function, it retains the optimal variable values obtained by the previous weight vector minimization, starting the search from the previously found alternative. Rebuilding the model from scratch at each iteration removes the previously obtained variable values, in which case the variables have the same starting value in each iteration. If a method like Spores, which is the only method that incrementally changes its weight vector, remains significantly faster even when the model is rebuilt at each iteration, it would suggest that its efficiency stems from the nature of the underlying near-optimal solution space, and not from the way that the weight vector is updated throughout the iterations.

Conversely, if the runtime increases substantially under this approach, the efficiency may largely be attributed to how the solver benefits from incrementally updating the weight vector.

The results when solving with this alteration on the full model are shown in Table 8.15.

Table 8.15: Runtime (s) of each method to find 1000 alternatives (Updating vs. Rebuilding the Solver)

Time steps	1		10		100	
	Updating	Rebuilding	Updating	Rebuilding	Updating	Rebuilding
HSJ	5.4	10.3	35.3	87.8	1644.0	2696.9
Spores	3.5	7.3	11.6	95.2	116.5	2280.1
Min/Max Variables	3.8	7.9	22.7	44.3	702.6	1559.4
Random Vector	3.9	6.7	22.4	44.4	691.6	1579.0
Directionally Weighted Variables	3.7	6.3	30.0	87.6	1521.5	2273.5

The results clearly show that all methods experience a significant increase in computational burden when the model is rebuilt for each alternative. This confirms that reusing the model across iterations offers substantial performance benefits. Spores is particularly affected, with Spores' runtime increasing the most, indicating that Spores' incremental weight updates are computationally efficient when reusing the model across iterations.

These results highlight a promising direction for future work: developing or adapting other methods to better exploit the solver's internal mechanisms, similar to Spores, could lead to significant improvements in computational efficiency.

An observation worth mentioning when rebuilding the model from scratch at each iteration was that HSJ produced significantly fewer unique alternatives compared to when the model was not rebuilt, shown in Table 8.16. This suggests that rebuilding the model influences the solver's behavior in finding certain unique alternatives. The biggest change that happens when a model is rebuilt is that its starting point is always the same, in contrast to what happens when updating the model, in which case each decision variable's starting value is the value of that specific variable in the previously found alternative. We hypothesize that methods that use weight vectors, which can include weights with value 0, will find fewer unique alternatives when rebuilding the model compared to when the model is updated at each time step.

Table 8.16: Unique alternatives generated by HSJ

Time steps	1	10	100
Rebuilt the near-optimal model at each iteration	5	7	6
Updating the model at each iteration	8	999	893

To confirm this suspicion, we will test it on the simplified model. This is done because Directionally Weighted Variables found more than 512 unique alternatives when updating the model. If the model is rebuilt at every iteration, it is expected that Directionally Weighted Variables finds fewer than 512 unique alternatives. The results of the simplified model with 100 time steps are shown in Table 8.17.

Table 8.17: Dominance rank distribution for the simplified model with 100 time steps

Method	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10+}
HSJ	4	0	0	0	0	0	0	0	0	0
Spores	46	0	0	0	0	0	0	0	0	0
Min/Max Variables	434	150	82	49	23	15	8	2	1	2
Random Vector	608	78	44	16	5	2	7	4	0	1
Directionally Weighted Variables	245	19	0	0	0	0	0	0	0	0

As expected, HSJ, Directionally Weighted Variables, and Min/Max Variables all found fewer unique alternatives, with HSJ having the relatively biggest difference and Min/Max Variables the smallest. The other methods had almost no difference in the number of unique alternatives found. This confirms the suspicion that methods using weight values of 0 find more unique alternatives when updating the model, when compared to rebuilding the model.

8.3. Summary

This chapter presented a detailed evaluation of six alternative generation methods — HSJ, Spores, Max-Distance, Min/Max Variables, Random Vector, and Directionally Weighted Vector — applied to both simplified and full models. The assessment focused on alternative dominance and computational efficiency across varying levels of temporal complexity.

Key findings from the baseline experiments are as follows. First, Directionally Weighted Variables consistently generate alternatives in the dominant set. Specifically, in the simplified model where it is clearly the best method for finding non-dominated alternatives. Second, in high dimensionality, the dominance metric does not find any dominated alternatives. Indicating that comparing alternatives with respect to dominance is unreliable in high-dimensional problems. Third, the runtime difference between Directionally Weighted Variables and the Min/Max Variables (which it alters) is shown to be based on the near-optimal solution space in which the methods generate alternatives, showing that both methods are computationally competitive. Fourth, it is also shown that, in low dimensions, the inverse of the Directionally Weighted Variables method generates fewer non-dominated alternatives than the Min/Max Variables, which further supports the claim that the Directionally Weighted Variables method generates alternatives within the dominant set. Lastly, it is shown that there is a big difference between updating the objective function of the model between iterations and rebuilding the model from scratch. Updating the model allows methods with weight values of 0 to find more alternatives, and allows for each method to find alternatives computationally faster.

9

Conclusion

This chapter concludes the thesis by reflecting on the role of dominance in near-optimal alternative generation. It also reflects on the proposed new method, which is based on a heuristic to find non-dominated alternatives.

9.1. Contributions

Traditional optimization approaches, which focus on identifying a single cost-optimal solution, fail to capture the full range of distinct, yet economically viable, alternatives. Most methods that generate alternatives do so blindly within the near-optimal solution space. Recognizing this gap, the thesis reframes the alternative generation problem to prioritize decision relevance rather than mere diversity or proximity to optimality.

This work provides a systematic analysis of several well-established MGA techniques, including Hop-Skip-Jump, Spores, Min/Max Variables, Random Vector, and Max-Distance metrics within energy system optimization. Although these methods differ in their heuristics and implementation strategies, a key insight of the thesis is that they can all be reformulated within a shared framework based on weight vector formulations. Doing so reveals the implicit assumptions, directional biases, and structural limitations of each method, such as their tendencies to explore only certain regions of the near-optimal solution space or produce redundant alternatives. This unifying perspective not only clarifies their conceptual differences, but also enables direct and consistent comparison across methods. This answers RQ1: *What are the conceptual differences between known MGA approaches for generating near-optimal alternatives?*

The thesis proposes a way to leverage the insights gained by conceptually comparing the existing methods by redefining what constitutes a useful or decision-relevant alternative. Existing MGA methods often focus on finding alternatives as different as possible from a reference point or randomly searching the near-optimal solution space. These methods lack a principled mechanism to prioritize alternatives that align with real-world decision-making needs. To bridge this gap, a dominance-based evaluation framework is proposed. By treating each decision variable as an implicit objective, this approach assesses alternatives based on dominance, ensuring that preferred alternatives represent distinct trade-offs between the different decision variables. This answers RQ2: *How can these differences be exploited to reformulate the alternative generation problem in a way that supports decision relevance?*

Building upon the definition of dominance, a new alternative generation method was developed using Directionally Weighted Variables that are aligned with dominance. This method actively searches for non-dominated alternatives within the near-optimal space, those that offer meaningful trade-offs between the decision variables. The thesis validates the proposed metric and generation approach through experiments on the Generation Expansion Planning problem. The results demonstrate that the new method produces a larger share of non-dominated alternatives, while remaining computationally feasible. This establishes the method as both a theoretical contribution and a practical resource

for decision-makers aiming to analyze and manage complex energy planning scenarios. This answers RQ3: *How does the proposed dominance-based method perform compared to existing MGA techniques in terms of decision relevance and computational efficiency?*

9.2. Limitations

While this thesis presents a novel framework for generating and evaluating near-optimal alternatives, several limitations should be acknowledged.

First, the dominance-based evaluation metric, while effective in identifying decision-relevant alternatives in low-dimensional settings, appears less informative in higher dimensions. This limitation can be interpreted in two ways. The first possibility is that dominance does remain meaningful, but the number of generated alternatives is too small to observe dominance relations reliably. The second possibility is that dominance itself may lose practical value as dimensionality increases, but the current experiments do not generate enough alternatives to determine this conclusively. In other words, either dominance works but is under-sampled, or it does not work, but this cannot yet be confirmed due to insufficient coverage of the solution space.

Second, while the Directionally Weighted Variables method aims to improve decision relevance, it relies on weight vector selection strategies that may still be sensitive to sampling bias or model structure. The method's performance in terms of dominance and computational costs may vary depending on the initial conditions, solver behavior, and model formulation.

Third, in the Generation Expansion Planning problem, the coefficients of investment costs are positive, as investment costs always increase the objective functions. Other models within the energy domain might have negative coefficients, and while in theory that should work with Directionally Weighted Variables, it is not applied to such problems in this thesis. These models might give differing results from what is shown in this thesis.

Finally, although this work emphasizes decision relevance, the evaluation remains primarily quantitative. The practical utility of the proposed alternatives for stakeholders or decision-makers has not been empirically validated, and further work is needed to integrate qualitative feedback or domain-specific preferences into the alternative generation and filtering processes.

9.3. Future works

Several promising directions remain for extending and improving the methods presented in this thesis.

First, this thesis focuses on the energy domain. Using near-optimal alternatives and, by extension, dominance within near-optimal alternatives might be interesting in other domains. In future work, one might be able to bridge the gap between the domains and apply dominance to these other domains.

Second, the space searched for alternatives can have a big impact on the alternatives that each method generates. This includes the alternatives in the dominant set. Changing the underlying near-optimal solution space can be achieved by either varying the optimality gap or using a different method that explores different parts of the near-optimal solution space. However, the reason why there are such changes and how these differences can be exploited is not discussed within this thesis. Future work should investigate why and how the underlying near-optimal solution space impacts the performance of different alternative generating methods.

Third, evaluating alternatives based on dominance proved difficult in higher dimensions. One potential improvement is to investigate the application of the convex dominant set, described in section 5.6, when comparing alternatives instead of just the dominant set, described in section 5.3, as done in this thesis. Additionally, generating more alternatives in higher dimensions allows for more rigorous dominance analysis, as more alternatives give clearer insight into the structure of the solution space and reduce the risk of drawing conclusions from sparsely sampled regions.

Fourth, further investigation is needed into the role of updating versus rebuilding the model. In particular, starting the search from a previously found alternative can both reduce the computational costs as well as increase the number of alternatives found. Future work could explore adaptive strategies that

dynamically decide when to update or rebuild the solver based on characteristics of the solution space, the optimality gap, or the specific method employed.

Lastly, another avenue worth exploring is the use of quasi-random sampling methods, such as Sobol Sequences [17] or Latin Hypercube Sampling [19], to more uniformly explore the near-optimal solution space. These quasi-random sampling methods may enable faster convergence and improve space coverage. When using these quasi-random sampling methods, the sequence order should be studied to see the effect it has on the alternatives that are generated.

Together, these directions offer exciting opportunities to build upon the foundations established in this thesis, enhancing the practical applicability, scalability, and theoretical robustness of near-optimal alternative generation in energy system models.

References

[1] Naomi Altman and Martin Krzywinski. “The curse (s) of dimensionality”. In: *Nat Methods* 15.6 (2018), pp. 399–400.

[2] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. “Qhull: Quickhull algorithm for computing the convex hull”. In: *Astrophysics Source Code Library* (2013), ascl–1304.

[3] Philip B Berntsen and Evelina Trutnevyte. “Ensuring diversity of national energy scenarios: Bottom-up energy system model with Modeling to Generate Alternatives”. In: *Energy* 126 (2017), pp. 886–898.

[4] Stephen Boyd. “Convex optimization”. In: *Cambridge UP* (2004).

[5] E Downey Brill Jr, Shoou-Yuh Chang, and Lewis D Hopkins. “Modeling to generate alternatives: The HSJ approach and an illustration using a problem in land use planning”. In: *Management Science* 28.3 (1982), pp. 221–235.

[6] Vašek Chvátal. *Linear programming*. Macmillan, 1983.

[7] Kalyanmoy Deb. “Multi-objective optimisation using evolutionary algorithms: an introduction”. In: *Multi-objective evolutionary optimisation for product design and manufacturing*. Springer, 2011, pp. 3–34.

[8] Kalyanmoy Deb et al. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE transactions on evolutionary computation* 6.2 (2002), pp. 182–197.

[9] Joseph F DeCarolis. “Using modeling to generate alternatives (MGA) to expand our thinking on energy futures”. In: *Energy Economics* 33.2 (2011), pp. 145–152.

[10] Joseph F DeCarolis et al. “Modelling to generate alternatives with an energy system optimization model”. In: *Environmental Modelling & Software* 79 (2016), pp. 300–310.

[11] Katharina Esser et al. “Participatory Modelling to Generate Alternatives for Decarbonising Campus Energy Supply”. In: *Available at SSRN 5065016* (2024).

[12] G Gunalay and Julian Scott Yeomans. “Multicriteria generation of alternatives for engineering optimization problems using population-based metaheuristics: A computational test”. In: *WSEAS Transactions on Computers* 18.31 (2019), pp. 239–247.

[13] Nyoman Gunantara. “A review of multi-objective optimization: Methods and its applications”. In: *Cogent Engineering* 5.1 (2018), p. 1502242.

[14] Raha Imanirad, Xin-She Yang, and Julian Scott Yeomans. “A biologically-inspired metaheuristic procedure for modelling-to-generate-alternatives”. In: *International Journal of Engineering Research and Applications* 3.2 (2013), pp. 1677–1686.

[15] International Energy Agency. *Developing Capacity for Long-Term Energy Policy Planning: A Roadmap*. Tech. rep. Licence: CC BY 4.0. Paris: International Energy Agency, 2024. URL: `https://www.iea.org/reports/developing-capacity-for-long-term-energy-policy-planning-a-roadmap`.

[16] Rui Jing et al. “Exploring the impact space of different technologies using a portfolio constraint based approach for multi-objective optimization of integrated urban energy systems”. In: *Renewable and Sustainable Energy Reviews* 113 (2019), p. 109249.

[17] Stephen Joe and Frances Y Kuo. “Constructing Sobol sequences with better two-dimensional projections”. In: *SIAM Journal on Scientific Computing* 30.5 (2008), pp. 2635–2654.

[18] Michael Lau, Neha Patankar, and Jesse D Jenkins. “Measuring exploration: evaluation of modelling to generate alternatives methods in capacity expansion models”. In: *Environmental Research: Energy* 1.4 (2024), p. 045004.

[19] Wei-Liem Loh. “On Latin hypercube sampling”. In: *The annals of statistics* 24.5 (1996), pp. 2058–2080.

[20] Francesco Lombardi and Stefan Pfenninger. “Human-in-the-loop MGA to generate energy system design options matching stakeholder needs”. In: *PLOS Climate* 4.2 (2025), e0000560.

[21] Francesco Lombardi, Bryn Pickering, and Stefan 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.

[22] Francesco Lombardi et al. “Policy decision support for renewables deployment through spatially explicit practically optimal alternatives”. In: *Joule* 4.10 (2020), pp. 2185–2207.

[23] Lukas Nacken et al. “Integrated renewable energy systems for Germany–A model-based exploration of the decision space”. In: *2019 16th international conference on the European energy market (EEM)*. IEEE. 2019, pp. 1–8.

[24] Fabian Neumann and Tom Brown. “The near-optimal feasible space of a renewable power system model”. In: *Electric Power Systems Research* 190 (2021), p. 106690.

[25] Vishwamitra Oree, Sayed Z Sayed Hassen, and Peter J Fleming. “Generation expansion planning optimisation with renewable energy integration: A review”. In: *Renewable and Sustainable Energy Reviews* 69 (2017), pp. 790–803.

[26] Tim T Pedersen et al. “Modeling all alternative solutions for highly renewable energy systems”. In: *Energy* 234 (2021), p. 121294.

[27] Stefan Pfenninger, Adam Hawkes, and James Keirstead. “Energy systems modeling for twenty-first century energy challenges”. In: *Renewable and sustainable energy reviews* 33 (2014), pp. 74–86.

[28] James Price and Ilkka Keppo. “Modelling to generate alternatives: A technique to explore uncertainty in energy-environment-economy models”. In: *Applied energy* 195 (2017), pp. 356–369.

[29] Henrik Schwaeppe et al. “Finding better alternatives: Shadow prices of near-optimal solutions in energy system optimization modeling”. In: *Energy* 292 (2024), p. 130558.

[30] Aditya Sinha et al. “Diverse decarbonization pathways under near cost-optimal futures”. In: *Nature Communications* 15.1 (2024), p. 8165.

[31] Diana Süsser et al. “Better suited or just more complex? On the fit between user needs and modeller-driven improvements of energy system models”. In: *Energy* 239 (2022), p. 121909.

[32] Evelina Trutnevyte. “Does cost optimization approximate the real-world energy transition?” In: *Energy* 106 (2016), pp. 182–193.

[33] Evelina Trutnevyte et al. “Context-specific energy strategies: coupling energy system visions with feasible implementation scenarios”. In: *Environmental science & technology* 46.17 (2012), pp. 9240–9248.

A

System Specifications

Processor: AMD Ryzen 7 PRO 5850U with Radeon Graphics, 1.90 GHz
Installed RAM: 16.0 GB (15.3 GB usable)
System type: 64-bit Operating System, x64-based processor