ORACLE: A rigorous metric and method to explore all near-optimal designs for energy systems

Evren M. Turan[a], Stefano Moret[a], André Bardow[a]

[a]*Energy & Process Systems Engineering, ETH Zürich, Zürich, 8092, Switzerland*

Abstract

Optimization models are fundamental tools for providing quantitative insights to decision-makers. However, models, objectives, and constraints do not capture all real-world factors accurately. Thus, instead of the single optimal solution, real-world stakeholders are often interested in the near-optimal space – solutions that lie within a specified margin of the optimal objective value. Solutions in the near-optimal space can then be assessed regarding desirable non-modeled or qualitative aspects. The near-optimal space is usually explored by so-called Modelling to Generate Alternatives (MGA) methods. However, current MGA approaches mainly employ heuristics, which do not measure or guarantee convergence.

We propose a method called ORACLE, which guarantees generation and exploration on the *entire near-optimal* space by exploiting convexity. ORACLE iteratively approximates the near-optimal space by introducing a metric that both measures convergence and suggests exploration directions. Once the approximations are refined to a desired tolerance, any near-optimal designs can be generated with negligible computational effort.

We compare our approach with existing methods on a sector-coupled energy system model of Switzerland. ORACLE is the only method able to guarantee convergence within a desired tolerance. Additionally, we show that heuristic MGA methods miss large areas of the near-optimal space, potentially skewing decision-making by leaving viable options for the energy transition off the table.

Keywords: Energy modelling, Decision support, Near-optimal, MGA

Email address: abardow@ethz.ch (André Bardow)

1. Introduction

The transition away from fossil resources necessitates massive shifts in the energy, chemicals, and fuels industries. However, decision-making in large-scale systems is inherently complex due to the intricate interdependencies among various decisions – for example, in an energy system, the cost-effectiveness of low-carbon power-to-hydrogen technologies depends on not only their costs but also the availability of green electricity or alternative generation and storage technologies [1, 2].

A common strategy to manage this complexity is to develop models that describe an abstracted version of a system, and run these models to provide insights to decision-makers. These models are often optimization-based techno-economic models that describe the technical, quantifiable requirements of the system (such as affordability, planning, and operation requirements) and find the best decisions to satisfy these concerns while minimising a specified objective, most commonly the total system cost. Thus, given a set of input parameters that make up a scenario, these models return the least-cost solution for that scenario. Energy system and integrated assessment models [3–8] are examples of such models that have been used since the 1970s to aid stakeholders in their decision-making.

However, providing only a single cost-optimal solution is both limiting and naïve [9]. Least-cost optimization is effective for identifying technically feasible scenarios while providing an estimate of the required costs. However, due to uncertainty of the future, simplifying assumptions, and idealisations, we can be confident that the cost-optimal solution is not the "real life" cost-optimal solution [9, 10]. Indeed, hard-to-model social and political factors can render the least-cost scenario untenable [10].

A particularly promising approach to handle these concern is to identify *near-optimal* solutions – that is, solutions that are similar in cost but otherwise differ [11–13]. The intuition is that, regardless of other stakeholder interests, excessively expensive decisions are not of interest. Once the near-optimal solution space has been defined, decisions can be analysed while taking un-modelled factors and trade-offs into account.

To find near-optimal designs, researchers have proposed a class of methods called Modelling to Generate Alternatives (MGA) [11–17]. MGA methods mainly employ heuristic strategies to explore the near-optimal space by repeated re-optimization of the system model. This exploration usually has two goals: (i) to identify maximally different solutions and (ii) to ensure a

diversity of solutions within the entire near-optimal space so that the exploration does not yield a bias towards/against specific combinations. However, as system models are typically large and computationally expensive, MGA methods are usually run for a specified, low number of iterations. Consequently, recent reviews have highlighted that the returned solutions rarely have guarantees of coverage, diversity, or of being unbiasedly distributed within the near-optimal space [13].

A recent variant, Modelling All Alternatives (MAA) [14, 17], attempts to address these issues by monitoring convergence of the volume of the space; however, due to computational complexity, MAA is limited to very low-dimensional problems ($\lesssim 8$ exploratory variables).

In this paper, we propose an approach, Optimization for the Rigorous Analysis of the Complete Landscape of Alternatives (ORACLE), to overcome these challenges by efficiently approximating the entire near-optimal space of a convex optimization model. Our method exploits convexity to iteratively refine outer and inner approximations of the near-optimal region. For this purpose, we introduce an interpretable convergence metric that measures how much of the space is left to explore. Once a desired approximation tolerance is achieved, we can sample near-optimal solutions at negligible computational cost. The sampling can be tailored to meet specific criteria, e.g., to be maximally diverse or uniformly distributed in the space. Such properties are difficult to guarantee when using the original system model due to computational restrictions.

Furthermore, our proposed convergence metric quantifies how much of the space is left to explore in interpretable units, and can be applied to monitor the convergence of existing and future MGA methods. We compare the ORACLE approach with existing methods on a large-scale sector-coupled energy systems model of Switzerland and show that significant areas of the near-optimal space are not explored by existing MGA methods, resulting in important near-optimal decisions being overlooked. In contrast, with similar computational effort per iteration, our method guarantees finding the entire near-optimal space. Crucially, while presented here for an energy system model, our proposed approach applies to any convex optimization model.

The paper is organised as follows: in Section 2, we briefly review existing MGA methods and establish the necessary mathematical background. In Section 3, we detail our proposed algorithm, which is then numerically demonstrated in Section 4, before concluding the paper with a brief discussion and outlook in Section 5.

2. Background and formal definitions

2.1. Optimal and near-optimal designs

The operation and design of an energy system can be written as an optimization problem, where the objective is to minimize the total system's cost subject to the constraints that define the system. To manage the computational complexity, a widely used approach is to set up the model as a large linear program (LP) [6–8] as these can be solved reliably and efficiently. An LP can always be written in the form:

$$v^* = \min_x \; c^T x \tag{1a}$$

$$s.t. \;\; Ax \leq b \tag{1b}$$

$$Fx = d \tag{1c}$$

where $v^* \in \mathbb{R}$ is the optimum objective value, $x \in \mathbb{R}^{n_x}$ is the decision vector, and $A \in \mathbb{R}^{n_i \times n_x}$, $b \in \mathbb{R}^{n_i}$, $F \in \mathbb{R}^{n_e \times n_x}$, $d \in \mathbb{R}^{n_e}$, $c \in \mathbb{R}^{n_x}$ are model parameters that define the n_i inequality constraints, n_e equality constraints, and the system costs respectively. As convention, we use the asterisk to denote optimality, e.g. x^* is an optimum solution with cost v^*. The optimization variable contains n_x entries (typically $\gg 10^5$) relating to both design/capacity decisions (e.g., how much hydro power to install) and operational decisions (e.g., how should the installed hydro power be used in each time step of the model). These energy system models are typically very large, and depending on the granularity of the spatial and temporal resolution, can take minutes to hours to solve, e.g., see Limpens et al. [8], Bröchin et al. [18], and the references therein.

Let $\epsilon\% \geq 0$ be the maximum allowable percentage deviation from the optimum value. The near-optimal space, $\mathcal{X}_\epsilon$, is defined as all points that satisfy the original constraints, and are within the budget defined by ϵ:

$$\mathcal{X}_\epsilon = \left\{ x \;\middle|\; Ax \leq b,\; Fx = d,\; c^T x \leq v^*(1+\epsilon) \right\} \tag{2}$$

As the decision vector x contains many entries, we typically do not want to explore the entire decision space; rather, we want to focus on the space of some key decisions. For example, for energy systems, we may only want to explore design decisions or the total annual use of a technology instead of hourly operating decisions. Let the decisions to explore be denoted by $z \in \mathbb{R}^{n_z}$, and the corresponding near-optimal region be $\mathcal{Z}_\epsilon$:

$$\mathcal{Z}_\epsilon = \left\{ z \;\middle|\; Sx = z,\; Ax \leq b,\; Fx = d,\; c^T x \leq v^*(1+\epsilon) \right\} \tag{3}$$

where $S \in \mathbb{R}^{n_z \times n_x}$ is a linear projection into the exploratory variable space. In the rest of the paper, $\mathcal{Z}_\epsilon$ will be referred to as the near-optimal space.

We assume that the upper and lower bounds of z are available (these can be determined from the upper and lower bounds of x). Furthermore, we assume that (i) for any z within these bounds the equality constraint $Fx = d$ can be satisfied and (ii) that $\mathcal{Z}_\epsilon$ has a non-empty interior, i.e. that the near-optimal space has an "inside". These assumptions are not restrictive, as they will be naturally satisfied for a reasonably defined study.

2.2. Modelling to generate alternatives

As systems models are typically very large, and not always available in a manipulable form, it is not tractable to find $\mathcal{Z}_\epsilon$ by projecting the constraints of (3) onto $\mathbb{R}^{n_z}$. Instead, researchers typically approximate $\mathcal{Z}_\epsilon$ by finding a point set of near-optimal designs, using Modelling to Generate Alternatives (MGA) approaches. MGA involves solving m optimization problems to get a matrix of points $\mathcal{Z}_{points} = \left[z_1^T, \ldots, z_m^T\right]$. As the vast majority of MGA papers are used with LPs, we do not consider methods that specialize on other problem classes. Most MGA algorithms solve the following problem at each iteration:

$$\min_{x,z} \; w_k^T z \tag{4a}$$
$$s.t. \; Sx = z \tag{4b}$$
$$Ax \leq b \tag{4c}$$
$$Fx = d \tag{4d}$$
$$c^T x \leq v^*(1+\epsilon) \tag{4e}$$

where $w_k \in \mathbb{R}^{n_z}$ is the k^{th} vector of weights chosen by the MGA algorithm and $W_{points} = \left[w_1^T, \ldots, w_m^T\right]$ is the dataset of weights used. Most MGA algorithms only differ in how W_{points} are chosen.

As exact MGA implementations differ between papers, we describe prototypical MGA approaches below, with important aspects of these approaches summarised in Table 1. Further details of MGA algorithms can be found in Appendix A.1 or in the recent review of Lau et al. [13].

Hop-Skip-Jump (HSJ): [11–13, 16, 19, 20]

HSJ is the earliest method known to us in the MGA literature. At each iteration, HSJ heuristically defines w_k with the aim of finding a z_k that

Table 1: A comparison of MGA algorithms in the literature with a selection of references describing their use and implementation.

Method	Parallelizable	Guarantees	References
HSJ (Hop-Skip-Jump)	No	-	[11–13, 16, 19, 20]
Random	Yes	Finds all vertices of $\mathcal{Z}_\epsilon$ with unlimited iterations	[13, 21]
VMM (Variable min/max)	Yes	Finds the minimum and maximum usage for each decision variable in finite iterations.[1]	[13, 22–25]
ERG (Efficient Random Generation)	Yes	-	[9, 26, 27]
SPORES (Spatially explicit practically optimal alternatives)	Partially [2]	Variations that first perform VMM inherit its guarantees.	[28–30]
MAA (Modelling All Alternatives)	No	Converges to $\mathcal{Z}_\epsilon$ with unlimited iterations.	[13, 14, 17]
Manhatten MGA	No	Finds a point cloud and distance such that there is no additional point in $\mathcal{Z}_\epsilon$ more than this distance from the point cloud.	[15]
ORACLE (Optimization for the Rigorous Analysis of the Complete Landscape of Alternatives)	Partially[3]	Converges to within a desired tolerance of $\mathcal{Z}_\epsilon$ in a finite number of iterations.	This work.

[1] Some authors only explore either the maximum and/or minimum usage. If implemented to swap to Random after finding the min/max usages, then the guarantees of Random apply.
[2] Search directions are generated sequentially in the main part of the algorithm, but multiple sequences can be generated in parallel.
[3] Calculation of the convergence metric has to be done sequentially, however many trial and feasible points can be generated in parallel.

is far from existing points in $\mathcal{Z}_{points}$. In HSJ, entries of w_k are chosen as the number of times the corresponding variable has a non-zero value in $\mathcal{Z}_{points}$. A variant, HSJ-rel, allocates entries as the sum of each entry in $\mathcal{Z}$ normalized by each variable's maximum attainable value. HSJ and HSJ-rel are inherently sequential, and hence MGA iterations cannot be computed in parallel. No convergence guarantees have been established for either variation. Additionally, prior authors have noted that HSJ tends to explore the space poorly [13].

Random (vector): [13, 21]
w_k is chosen randomly from a uniform distribution, with entries between -1 and 1. Although this approach is very simple, it has two important properties:

1. In high-dimensional space, any two random vectors are likely to be near-orthogonal [31]. Thus, different parts of the space are naturally explored, and at first, each new weight is likely to be near-orthogonal to *all* previous weights.
2. As the number of iterations goes to infinity, this approach is guaranteed to explore every vertex of the near-optimal space.

Variable Min/Max (VMM): [13, 22–25]
VMM *systematically* assigns w's with only one non-zero entry of either 1 or -1 to find the maximal and/or minimal possible usage of each technology within the near-optimal budget. Hence, at most $2n_z$ points will be found. After these points, the algorithm can either terminate or proceed to follow another MGA approach, e.g. Random, as in Lau et al. [13], if more points are desired.

Efficient Random Generation (ERG): [9, 26, 27]
ERG *randomly* assigns w_k entries of -1, 0, or 1. Given infinite iterations, this method will recover the extreme solution bounds of VMM; however, unlike Random, it is not guaranteed to explore every vertex in the limit.

Spatially explicit practically optimal alternatives (SPORES) [28–30]
SPORES originates as a hybrid of HSJ-rel and VMM that was originally proposed for spatially resolved models [29]. Later variations have hybridised VMM with other weighting schemes [28, 30]. In these variations, the search direction is given by a weighted linear combination of

the weight from VMM and the other schemes. Like HSJ-rel, these other schemes are sequential, meaning that an incumbent search direction is based on earlier search directions. If multiple sequences are generated, these sequences can be run in parallel. No convergence guarantees specific to SPORES have been established. If exploration is first performed using only VMM, and then the hybridisation is introduced (as in de Tomás-Pascual et al. [30]), SPORES inherits VMM's guarantees.

Modelling all alternatives (MAA): [13, 14, 17, 32]
Unlike the other MGA approaches, MAA is a region-based approach. MAA first generates an initial point cloud with VMM. After this phase, MAA iteratively forms the convex hull of $\mathcal{Z}_{points}$ and tries to expand this convex hull by selecting w_k to be the normal of one of the facets of the hull. This iteration continues until a convergence criterion based on the volume of the convex hull is met. As discussed in Section 2.3, this approach is limited to a low number of exploratory variables due to the computational effort of forming the convex hull.

Manhattan MGA [15]
To the author's knowledge, Manhattan MGA is the only MGA algorithm (for continuous models) that does not use the standard MGA problem (4). Instead, the objective is to maximize the Manhattan distance between the new point and all previous points. This requires formulating an MILP with the energy system model and is thus computationally impractical for large models.

2.3. Use of the convex hull in MGA

The underlying idea of MAA is forming the convex hull of $\mathcal{Z}_{points}$ to explore the near-optimal region. The convex hull of a set of points is the smallest polytope that contains all the points. Since the model is convex, the convex hull of $\mathcal{Z}_{points}$ forms the polytope, $\mathcal{I}$, which lies entirely within the near-optimal region, i.e. $\mathcal{I} \subseteq \mathcal{Z}_\epsilon$. Thus, $\mathcal{I}$ is an inner approximation, as all points in $\mathcal{I}$ are guaranteed to be near-optimal. Authors have proposed using the convex hull both to guide the exploration of $\mathcal{Z}_\epsilon$ (MAA), and as a post-processing step to generate further points and perform further analysis [33].

MAA requires forming the halfspace representation of the convex hull from the m points in $\mathcal{Z}_{points}$. For more than a few dimensions, computing

this representation is computationally intractable due to the curse of dimensionality: for state-of-the-art methods, the execution time grows $\mathcal{O}(m^{\lfloor n_z/2 \rfloor})$ [34]. Thus, practically MAA-based methods are only applicable for $n_z \lesssim 8$, and even then can be computationally prohibitive when the number of points is large.

In summary, most MGA methods explore $\mathcal{Z}_{points}$ by finding a fixed, predetermined number of points without measuring convergence or coverage of the space. MAA and Manhattan can measure their coverage of the space[1]; however, both have computational limitations. Thus, no existing method can map the near-optimal space of a large-scale convex optimization model without restricting the exploration to only a few variables.

3. Methodology

3.1. The ORACLE algorithm

To overcome the limitations discussed in Section 2.3, we propose a method to map near-optimal spaces, called ORACLE: "Optimization for the Rigorous Analysis of the Complete Landscape of alternatives in Energy systems"[2]. The underlying idea of ORACLE is to leverage convexity to *adaptively* construct approximations of the near-optimal space while measuring the convergence of these approximations to within a desired tolerance.

ORACLE, shown in Figure 1, iteratively shrinks an outer approximation, $\mathcal{O}$, while iteratively expanding an inner approximation, $\mathcal{I}$, until they converge to within some distance, tol. Importantly, this distance is the *maximum possible* error incurred when generating near-optimal points, as the true near-optimal region is sandwiched between the two approximations. Thus after generation, the regions can be used, e.g. to generate many near-optimal points, while guaranteeing that the maximum error is below a desired tolerance. Additionally, as shown in Section 4, this metric can also be used to measure the convergence of other MGA approaches.

The key steps of ORACLE are outlined below, with numbering corresponding to the steps shown in Figure 1. Further details are provided in

[1]The original MAA method [17] uses a heuristic based on relative improvement, but a later version of the method [32] introduces a valid convergence metric. See Section 4.3 for futher details.

[2]The source code of ORACLE is available at: `place-holder-until-publication.`

the subsequent sections, while the pseudocode and potential variations are in Appendix A.3.

Step 1: Given the cost-optimal design, and additional system information (e.g., other known designs, maximum capacities), using convexity, we form initial inner and outer approximations ($\mathcal{I}$, $\mathcal{O}$) of the entire near-optimal region.

Step 2: To monitor convergence, and to guide exploration of the near-optimal space, we find a trial point, $z^*_{\mathcal{O}}$, which is the furthest point in the outer approximation from the inner approximation. This distance ($d_{\mathcal{OI}}$) is directly interpretable as the maximum error of the approximations. If this distance is below a desired tolerance, we proceed to Step 5; otherwise, we proceed to Step 3.

Step 3: We then explore in the direction of the trial point by finding the nearest point in the near-optimal region, z^*_f, to $z^*_{\mathcal{O}}$. This requires solving the energy systems model with a distance-minimizing objective.

Step 4: We then grow the inner approximation, $\mathcal{I}$, by including the new point, z^*_f, in its data. The outer approximation is shrunk, cutting off a portion of $\mathcal{O}$ which contains $z^*_{\mathcal{O}}$.

Step 5: The refinement of the approximations stops once the distance between the approximations ($d_{\mathcal{OI}}$) falls below a desired tolerance. At this point, near-optimal designs can be generated or other analyses performed with negligible computational effort using only the inner and outer approximations.

3.2. Mathematical formulation

This section provides in-depth details on the mathematical formulations and the steps of ORACLE.

3.2.1. Preliminaries of inner and outer approximations

To form an inner approximation, $\mathcal{I} \subseteq \mathcal{Z}_\epsilon$, we define the convex hull implicitly as the convex combination of all points in $\mathcal{Z}_{points}$:

$$\mathcal{I} = \left\{ z \;\middle|\; \lambda^T \mathcal{Z}_{points} = z,\ \sum_{i=1}^{m} \lambda_i = 1,\ \lambda \geq 0 \right\} \tag{5}$$

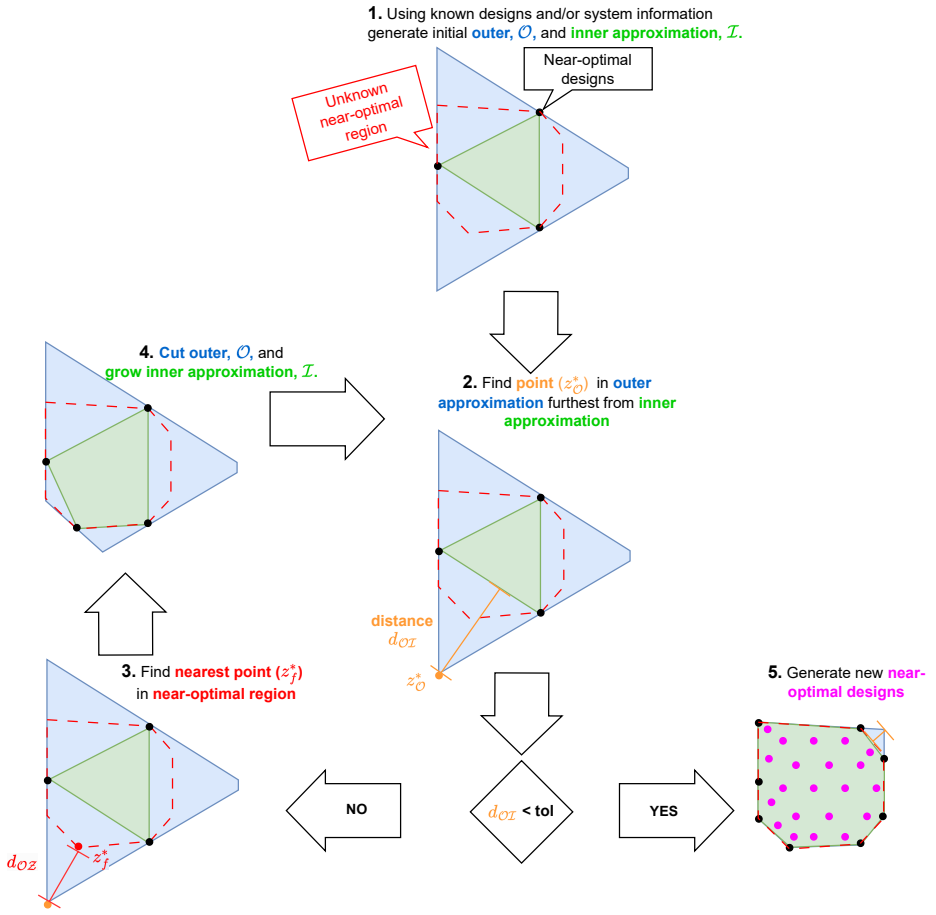


Figure 1: The key steps of the ORACLE algorithm. ORACLE is initialized by constructing an initial outer and inner approximation ($\mathcal{O}$ and $\mathcal{I}$) using known near-optimal points and system information. In Step 2, the main loop of ORACLE begins by finding the point in $\mathcal{O}$ furthest from $\mathcal{I}$. If the distance is below a desired tolerance, the algorithm terminates, and the resulting polytope(s) can be used to generate near-optimal designs at negligible computational effort. Otherwise, ORACLE locates the nearest feasible point to the outer point (Step 3), and uses this solution to grow the inner approximation, and (if possible) cut the outer approximation (Step 4), before going back to Step 2.

with $\lambda \in \mathbb{R}^m$. Thus, for a point z to be in $\mathcal{I}$, one must be able to find a λ satisfying the constraints of (5). Unlike explicitly constructing the convex hull, verifying whether a point satisfies (5) (or constraining a point to lie in $\mathcal{I}$) is an LP, which remains computationally tractable even for many high-dimensional points. While this formulation of the convex hull was used by Lau et al. [33] for post-processing of MGA solutions, this is the first time it has been proposed to explore the space to the best of the authors' knowledge.

In addition, we define an outer approximation of the near-optimal space, $\mathcal{Z}_\epsilon \subset \mathcal{O}$, by defining a set of linear inequality constraints that are true for any $z \in \mathcal{Z}_\epsilon$:

$$\mathcal{O} = \left\{ z \mid y_{O_i}^T (z - z_{O_i}) \leq 0, \ \ \forall i = 1, \ldots, n_{\mathcal{O}} \right\} \tag{6}$$

where $y_{O_i} \in \mathbb{R}^{n_z}$ and $z_{O_i} \in \mathbb{R}^{n_z}$ are parameters – respectively the ith normal and point of a hyperplane. A simple way to form an outer approximation is to note that, due to convexity, the search direction w_i used in the standard MGA problem (4) is the normal of a supporting hyperplane of $\mathcal{Z}_\epsilon$ through z_i [35].

In the following, we provide further details on the steps that make up ORACLE.

3.2.2. Step 1: Defining the initial regions

The inner approximation is always defined as the convex combination of known points, as in (5). Initially, the cost-optimal solution will be known, so the inner approximation will always have at least one point. Any other known designs can be included in the definition.

The initial outer approximation is defined by (linear) inequalities that are known to be satisfied for any point in the true near-optimal region. The simplest example of such constraints are the upper and lower bounds of z, (e.g. the first element may have bounds $0 \leq z_1 \leq 5$). Another simple constraint that can be included is that an under-approximation of the system cost $\underline{v}$, should satisfy the near-optimality constraint[3]:

$$\underline{c} z = \underline{v} \leq v^*(1 + \epsilon) \tag{7}$$

where $\underline{c} \in \mathbb{R}^{n_z}$ is some parameter that under-approximates the contribution of z to the total system cost. For example, often the choice $\underline{c} = Sc^T$ is valid

[3]If $\underline{v}$ is an under-approximation, then any point in $\mathcal{Z}_\epsilon$ is guaranteed to satisfy (7). Thus, to be part of the over-approximation, $\underline{v}$ must be an under-approximation.

as this approximates the total system cost as only the cost of using decisions z. Although this approximation is unlikely to be tight, it can be useful if an exploratory variable is unbounded (or has a very high bound), by ensuring that the outer approximation is bounded. For simplicity, we assume that the initial outer approximation is compact (it is bounded and includes its boundary).

3.2.3. Step 2: Finding a trial point in the outer approximation

The next step is to find a trial point in the outer approximation that is not in the inner approximation, and ideally outside the near-optimal region. In ORACLE, we find the point in the outer approximation that is furthest from the inner approximation. The intuition is that either the trial point is in the near-optimal region, and hence the inner approximation should grow greatly (as this point was far from it), or the trial point is outside the near-optimal region, and the outer approximation should shrink. This refinement is done in the following step.

To generate this trial point, we maximise the minimum distance between $\mathcal{I}$ and $\mathcal{O}$:

$$d_{\mathcal{IO}} = \max_{z_O \in \mathcal{O}} \min_{z_I \in \mathcal{I}} \|z_O - z_I\|_p \tag{8}$$

where p determines the norm used to measure distance. This problem returns $d_{\mathcal{IO}}$, the max-min distance between $\mathcal{I}$ and $\mathcal{O}$, the trial point, z_O^*. As the near-optimal region lies between the outer and inner approximation, $d_{\mathcal{IO}}$ is an upper bound on the distance to the near-optimal region. Thus, it is interpretable as the maximum error of using either $\mathcal{I}$ or $\mathcal{O}$ to approximate the near-optimal region $\mathcal{Z}_\epsilon$.

Problem (8) is a bilevel optimization problem and cannot be provided as is to most solvers. By picking a p such that the lower level is convex, e.g. 1, 2, or ∞[4], problem (8) can be reformulated to yield a single-level problem.

We choose the infinity norm ($p = \infty$) as this measures the maximum error due to a single element of z, making the metric easy to interpret. With this choice, problem (8) is a bilevel LP, which we reformulate into a single-level mixed integer linear program (MILP, see Appendix A.2 for details on the reformulation), which can be solved using off-the-shelf solvers.

[4]Also known as the maximum norm, as $\|z_O - z_I\|_\infty = \max_{i=1,\dots,n_z} |z_{O_i} - z_{I_i}|$.

3.2.4. Step 3: Finding the closest feasible point to the trial point

Given a trial point, we need to check if it is feasible. To do so, we use the system model to find the closest feasible point, using the same distance metric as in problem (8). Selecting $p = \infty$, yields the problem:

$$\min_{\delta, z_f, x} \ \|\delta\|_\infty \tag{9a}$$

$$z^*_{\mathcal{O}} - z_f = \delta \tag{9b}$$

$$Sx = z_f \tag{9c}$$

$$Ax \leq b \tag{9d}$$

$$Fx = d \tag{9e}$$

$$c^T x \leq v^*(1+\epsilon) \tag{9f}$$

where δ is the difference between between z_f and $z^*_{\mathcal{O}}$, and z^*_f is the optimal solution. Reformulating the infinity norm as linear inequalities results in an LP of similar size to solving the original systems model (1) or the standard MGA problem (4).

If the trial point $z^*_{\mathcal{O}}$ is infeasible, we can both refine the inner approximation (using the nearest feasible point) and generate a cut that will be a supporting hyperplane of the true near-optimal region (as z^*_f lies on the boundary). If we directly checked the feasibility of the trial point $z^*_{\mathcal{O}}$[5], then (i) the inner approximation would only be refined if $z^*_{\mathcal{O}}$ proved feasible, and (ii) if $z^*_{\mathcal{O}}$ were infeasible the resulting cut might be positioned far from the boundary of the near-optimal region, limiting its effectiveness.

3.2.5. Step 4: Refining the outer and inner approximations

At each iteration, we can grow the inner approximation by including the closest feasible point, z^*_f, in the set of points defining the polytope (5). In contrast, if the trial point is feasible, $z^*_f = z^*_{\mathcal{O}}$, then we cannot refine the outer approximation at this iteration as this point "confirms" the current outer approximation. When the trial point is infeasible, the outer approximation can be refined by adding a valid inequality to equation (6).

If z^*_f is a vertex of the constrained region, then infinitely many supporting hyperplanes exist at that point. Our goal is to find a *strictly separating hyperplane*, i.e. one that satisfies $y^T_{O_i} z^*_{\mathcal{O}} > y^T_{O_i} z^*_f$. This requirement ensures

[5] i.e. by using the constraint $Sx = z^*_{\mathcal{O}}$.

that whenever an infeasible point is found, the trial point, and thus part of the outer-approximation $\mathcal{O}$, is cut off by introducing the new constraint into $\mathcal{O}$. This property guarantees convergence of $\mathcal{O}$, as the initial $\mathcal{O}$ has finite volume (being compact) and each iteration removes a nonzero volume from it.

A strictly separating hyperplane is given by the dual variables of equation (9b) – see Appendix A.4 for further details. We additionally cut $\mathcal{O}$ by introducing a linear under-approximation of the total system cost at z_f^*. This cut requires an additional minimization of the total cost with the constraint $Sx = z_f^*$, but can accelerate convergence to the near-optimal region (see Appendix A.4.2).

3.2.6. Step 5: Sampling from the approximations

Once the approximations have converged to a desired tolerance, one can sample from either the inner ($\mathcal{I}$) or outer approximations ($\mathcal{O}$) with considerably less computational effort than it takes to solve the original energy system model due to the dramatic decrease in problem size ($n_z \ll n_x$). The sampling can be performed very flexibly, and allows for the potential use of (i) any MGA method with the $\mathcal{I}$ or $\mathcal{O}$ as constraints instead of the original model as constraints, (ii) Markov chain Monte Carlo methods such as Hit-and-Run and its later variations [36–38] to find points *uniformly distributed* in the near-optimal region or (iii) any other optimization problem to find points with certain characteristics. For illustration, we find *maximally distant points* in Section 4.4.

3.3. Related work and limitations

To the authors' knowledge, the idea of an outer approximation has only appeared once in the MGA literature [32], where the volume of $\mathcal{O}$ was used to normalise the volume of the convex hull to monitor convergence of MAA. Similarly to finding the convex hull, calculating the volume rapidly becomes computationally intractable [34], and is only practical for $n_z \lesssim 8$.

In contrast, ORACLE requires an MILP to find the trial point in the outer approximation in problem (8). The computational cost of solving an MILP is often strongly determined by the number of integer variables in the problem. The proposed reformulation uses $2n_z + m$ integer variables, where m is the number of points used to define the inner approximation. Thus, for problems that require many points for convergence, the MILP may become computationally costly. Methods for reformulating and solving

bilevel LPs are an active area of research [39–41]. We have not compared the computational effort of these different formulations, nor in the potential use of heuristic methods to solve the MILP resulting from problem (8).

ORACLE adaptively approximates a convex polytope. This geometric problem has been studied in other contexts, including the compression of large systems of linear inequalities, determining reachable sets, multi-objective optimization and scheduling [42–46]. The major differences of ORACLE to these works are (i) the computation and formulation of the proposed distance metric to find a trial point, (ii) the target polytope is indirectly defined by a very large optimization problem where the problem data is challenging to manipulate, and (iii) the application to large-scale energy systems. We note that the approach proposed in Sung and Maravelias [46] for finding the attainable region of a scheduling problem combines elements of ORACLE and MAA, in that a variant of problem (8) is solved with $p = 2$, and the search direction restricted to be a normal of the facets of the inner approximation. This requires forming the convex hull and swapping between half-space and vertex representations, as in MAA, which rapidly becomes computationally intractable [34].

4. Numerical experiments

4.1. Model and experimental set up

We use the EnergyScope TD model of Switzerland [8] to demonstrate the performance of the ORACLE algorithm and to benchmark it against several existing MGA approaches: Random, VMM, HSJ, SPORES, and ERG. We exclude MAA and Manhattan MGA from the comparison due to their computational limitations. Additionally, we evaluate our proposed metric against the volume-based metric found in the MGA literature [13, 17, 32]. All optimization problems are implemented in AMPL [47] and solved with Gurobi version 12 [48]. When solving LPs, we use Gurobi's homogeneous barrier algorithm method without crossover and set the convergence tolerance to 1e-6. For MILPs, we use a relative and absolute MIP gap of 0.1 and 0.05, 32 threads, and impose a time limit of 600 seconds. Volumes are calculated with Quickhull [34] using the "QJ" setting.

EnergyScope TD is a sector-coupled energy system model that optimizes a greenfield representation of Switzerland in 2050. The model is spatially aggregated into a single node and resolved hourly using twelve typical days.

We selected EnergyScope TD for this study as it is a national-scale sector-coupled model that is computationally efficient (evaluations of ≤ 60 seconds), making it suitable for extensive comparisons across MGA methods.

In our numerical results, we consider the following six installed capacities as exploratory variables: photovoltaics (PV), (onshore) wind, nuclear, natural gas combined cycle (gas), with and without carbon capture and storage (CCS), and coal power plants (with CCS)[6]. We emphasize that this list does not contain all technologies in the model. A small list of technologies is necessary for the computation of volumes in the method comparison. All exploratory variables are measured in GW, and all MGA methods are run until they converge within 0.1 GW, calculated by equation (8), or reach 200 iterations. A deviation of 0.1 GW is typically insignificant in the context of strategic decision-making for a national energy system; therefore, this tolerance is relatively stringent, ensuring that the near-optimal space has been rigorously explored.

This section is arranged as follows: we compare ORACLE with other MGA methods, using first our proposed metric, equation (8), and then by volume. We then highlight the benefit of our method by using the converged approximation to identify designs that are missed by the other MGA methods.

4.2. Comparison of MGA convergence

Although most MGA algorithms do not construct outer and inner approximations, they generate data that we use to construct these approximations (see Section 3.2). We initialize the approximations of all MGA algorithms using Step 1 of ORACLE.

The methods (Figure 2) show a stark contrast. ORACLE consistently outperforms all others and converges rapidly. ORACLE reaches a 1 GW tolerance in just 30 iterations, and converges to the desired tolerance (0.1 GW) within 144 iterations. None of the other methods even reach the 1 GW threshold within 200 iterations. The convergence of ORACLE is "noisy" compared to the other methods due to the numerical tolerances used when solving problem (8), which becomes visible when the distance becomes small.

HSJ shows minimal improvement, maintaining a nearly constant distance throughout. Similarly, SPORES initially improves rapidly before reaching a

[6]Although coal power plants do not appear in a cost-optimal system, they can be built in the near-optimal space.

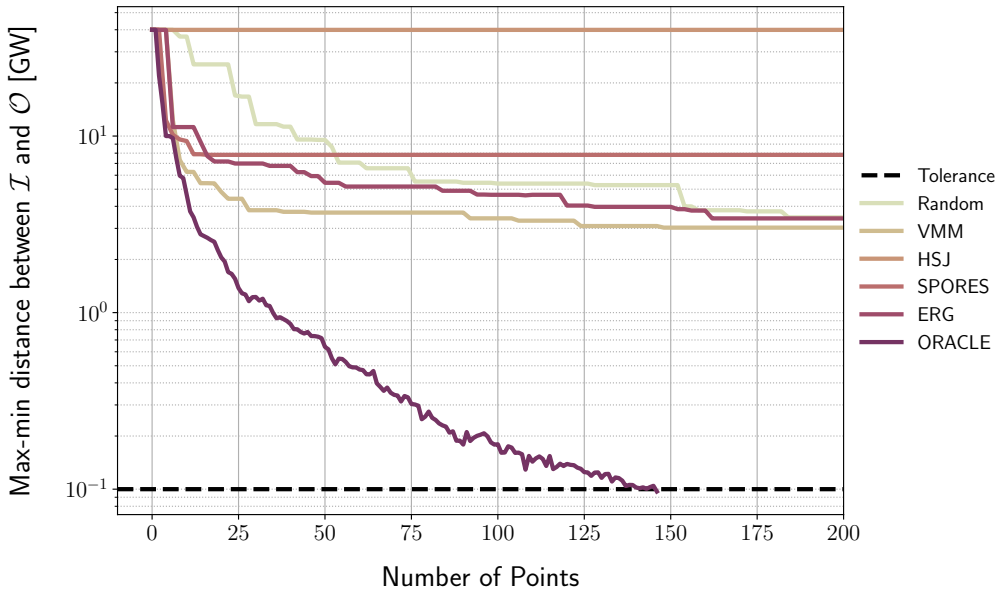


Figure 2: Distance between inner and outer approximations of the near-optimal space for state-of-the-art MGA methods and the ORACLE algorithm. ORACLE's non-monotonic convergence is a numerical artifact, as problem (8) is solved with relative and absolute termination criteria of 10% and 0.05 respectively.

plateau. The maximum distance will only change[7] if a search direction (w_k in problem (4)) is picked that coincides with the direction of maximum distance. Both HSJ and SPORES generate a sequence of search directions that are autocorrelated (the search direction at iteration k depends on previous weights). This autocorrelation may cause HSJ and SPORES to explore only a restricted part of the near-optimal space. Another group (Random, VMM, ERG) exhibits sporadic progress, reducing the maximum distance from 40 GW to around 4.5 GW within 200 iterations. The initial, rapid improvement of VMM is due to the systematic optimization for the "real" range of each decision variable. After this phase, VMM is equivalent to the Random method, which has a chance to select the direction of maximum separation. As the ERG and Random methods are also similar to each other, it is unsurprising that these methods have similar trajectories.

4.3. Comparison of MGA convergence by volume

Volume has been suggested as a convergence metric in the recent MGA literature [13, 17, 32]. When using the difference between the volumes of the inner and outer approximations as a metric, the groupings of the algorithms (Figure 3) are the same as when the proposed metric (Figure 2) is used: ORACLE performs the best followed by Random, VMM, ERG, and Spores with HSJ performing the worst. In fact, the volumes of the inner approximations of HSJ are not shown as they are very small in comparison. Small volumes have been reported for HSJ have been reported in earlier work [13].

Interestingly, the volumes of the inner approximations of Random, VMM, and ERG are similar to the converged volume of ORACLE. However, as the volumes of their outer approximations are much larger, they are unable to show that the inner approximations appear to have converged by volume. This finding implies that the poor performance of Random, VMM, and ERG in Figure 2 is because their outer approximations are not tight. Despite the volumes being similar, the distance between the inner approximation of Random, VMM, and ERG with the converged outer approximation of ORACLE remains above the desired tolerance by a significant margin. Thus, although the volumes of the inner approximations are similar, the inner approximations have not converged to the desired tolerance (see Appendix A.6).

[7] We emphasize that whenever the maximum distance remains constant, this does not mean that the size of the approximations necessarily remains constant (they could be growing or shrinking in other directions).

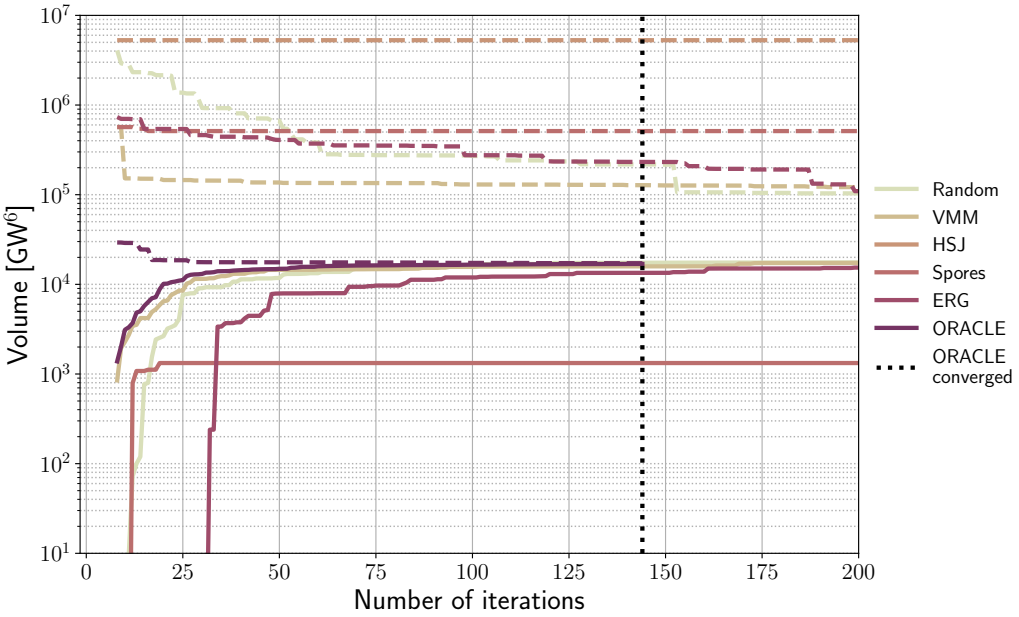


Figure 3: Volumes of the inner (solid lines) and outer (dashed lines) approximations of state-of-the-art MGA methods and of the ORACLE algorithm. ORACLE terminates after 144 iterations, having converged to within 0.1 GW (see Figure 2).

Earlier work [13, 17] only used the change in the volume of the inner approximation as a heuristic to monitor convergence. Figure 3 clearly illustrates the difficulty of using the rate of change, as it becomes small for ORACLE, Random, VMM and ERG after approximately 50 iterations despite the inner approximations having different sizes. Additionally, as the volumes of the inner approximations of HSJ and SPORES are smaller than those of the other algorithms, their volumes change by less, which, by this metric, would be seen as convergence. Thus, the rate of change of volume does not appear suitable for measuring convergence. Schricker et al. [32] proposed monitoring the ratio of the volumes of the inner and outer approximations, which is plotted in Appendix A.5. This metric gives the same final ranking as our proposed distance metric; however, due to computational limitations, volume can only be calculated for a small number of exploratory variables[8].

4.4. What designs are left out by the other MGA methods?

The proposed ORACLE method generates near-optimal regions that enable analyses that would be computationally infeasible using the original system model (defined by problem (1)). To demonstrate this capability, we formulate an optimization problem to answer the question: "What is the most different design that another MGA method did not find?"[9]. This analysis is done by solving the following optimization problem to find the furthest point in the (approximated) near-optimal region to the point cloud generated by another MGA method:

$$\max_{z_{\mathcal{O}},\delta} \ \delta \tag{10a}$$

$$\delta \leq d(z_{\mathcal{O}} - z_{MGA,i}) \qquad \forall i = 0, \ldots, m \tag{10b}$$

$$z_{\mathcal{O}} \in \mathcal{O} \tag{10c}$$

where d is the desired distance function, m is the number of points found by the other MGA methods, and δ is the maximum distance. In this section, we select the Manhattan distance, $d = \|\cdot\|_1$. Compared to optimizing the two

[8] In Lau et al. [13], it is suggested to approximate the volume by calculating the sum of the projected volumes of each pair of elements in z. This estimate is not guaranteed to increase whenever the actual volume increases, and so exacerbates the issues with using the rate of change.

[9] As a further example of flexibility, note that a variation of problem (10) can be used to find the most diverse set of designs within the near-optimal region.

and infinity norm, the Manhattan distance tends to find solutions that differ across multiple dimensions rather than finding solutions that differ greatly in only one dimension.

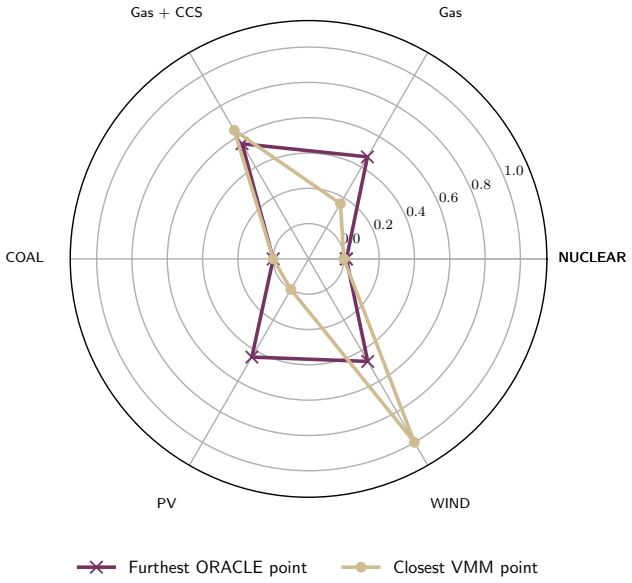
We select VMM and HSJ (run for 200 iterations) as representative examples from the "groups" of Figures 2 and A.5, and identify their potentially incorrect conclusions due to missed designs (Figure 4) . For example, Figure 4 (a) shows a near-optimal design identified by ORACLE that employs moderate photovoltaics, wind, gas, and gas with CCS. However, the closest point found by VMM is one with high wind and no photovoltaics[10]. Thus, one could incorrectly conclude that moderate deployment of photovoltaics, wind and gas is not viable. Figure 4 (b) shows a more dramatic result; due to the poor convergence of HSJ, the closest HSJ point to the identified ORACLE design of high gas, wind and PV is very far away. In this case, one could incorrectly conclude that high deployment of wind implies that gas and photovoltaics should not be deployed, while their co-deployment is actually within the near-optimal space.

This finding highlights that the lack of convergence metrics and exploration guarantees of MGA methods is not solely a theoretical concern, but a practical concern, as it can significantly impact the information provided to stakeholders. Additionally, as the proposed convergence metric, equation 8, directly estimates the maximum error in the results, its usefulness is immediately apparent: even if exploration is performed using a different MGA method, the proposed metric can be used to certify whether the results can be trusted. For example, as ORACLE has converged within 0.1 GW, the near-optimal space has essentially been fully explored.

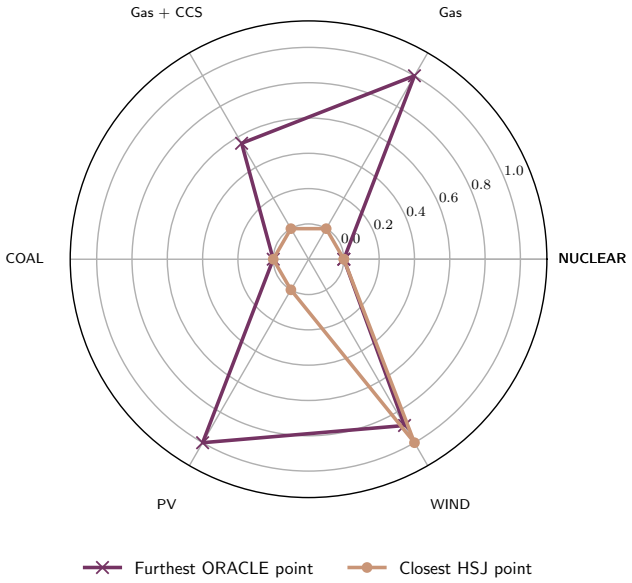
5. Conclusion

To avoid the bias of least-cost solutions [10], numerous researchers have proposed to instead generate *near-optimal solutions* [9, 11, 12, 15, 25]. Once generated, the near-optimal solutions can be analysed based on stakeholder interests, while taking un-modelled decision factors into account. Currently, near-optimal solutions are generated by a class of methods called Modelling

[10]The two designs have a significant difference in installed technologies. As not all electricity generation technologies are explored and energy imports are allowed, the HSJ design can use these to supply the energy demands of the system.



(a) Comparison with VMM



(b) Comparison with HSJ

Figure 4: Identification of strategies not found by the literature MGA methods. Each radial plot identifies a design found by ORACLE and the closest design found by the respective MGA method. The installed capacity of each technology is normalised against its maximum potential capacity.

to Generate Alternatives (MGA). These methods employ heuristics to generate diverse solutions in the near-optimal space. However, the majority of MGA methods do not measure their coverage of the space, have few or no performance guarantees, and normally run for a pre-specified number of iterations before terminating. Recent variants [15, 17] have attempted to address these concerns, but have computational limitations that restrict their usage to either small models or a low number of exploratory variables.

We have proposed a convergence metric that quantifies how much of the space is left to explore. Importantly, this metric can be applied to existing MGA methods, allowing for easy estimation of their convergence. Furthermore, we have developed an approach called ORACLE that leverages convexity alongside the convergence metric to adaptively approximate the entire near-optimal space to within a desired tolerance. Once converged, near-optimal designs can be generated with negligible computational cost. In this step, the design generation can be tailored to have desired characteristics: e.g., to be most diverse, uniformly distributed, and so on.

We compare ORACLE with five other MGA algorithms to explore the 10% near-optimal region of a greenfield, sector-coupled energy model of Switzerland. ORACLE converges within 144 iterations, while the other MGA methods are unable to guarantee their convergence within 200 iterations (Figure 2). We demonstrate the impact of this non-convergence by finding very different near-optimal designs that are missed by the MGA methods (Figure 4). Additionally, we compare our proposed metric with the use of volume, as suggested in the MGA literature. While the rankings of the methods are the same, our proposed method can be calculated for larger dimensions than volume, while additionally providing a direct estimate of the error of the corresponding MGA method in convenient units.

The computational time per iteration of ORACLE is more expensive than that of other MGA methods, as it involves not only evaluating the system model, but also performing another optimization to evaluate the convergence metric. System models with high spatial and temporal resolution are computationally expensive to evaluate; hence, this step is likely to dominate the time per iteration, with the additional time per iteration of ORACLE being negligible in comparison (see Section Appendix A.7). To compensate for the increase in the computational time of using a larger model or exploring more variables, strategies can be employed such as partial parallelization and hybridization of ORACLE with other algorithms (for further details, see Appendix A.3).

In conclusion, while the MGA literature has been rapidly growing [9, 13], insufficient attention has been paid to the quality of exploration of MGA methods. Using the proposed metric, problem (8), we can rigorously measure the potential error of using a particular method, thus allowing for MGA methods to be rigorously compared against each other. Thus, even if exploration is performed using a different MGA method, we recommend using the proposed metric to measure the exploration quality.

Declaration of competing interest

The authors declare the following financial interests and personal relationships that could be considered as potential competing interests: AB has served on review committees for research and development at ExxonMobil and TotalEnergies, companies active in both oil and gas and chemical production. AB holds ownership interests in firms that provide services to industry, some of which may operate in the chemical industry. In particular, AB has ownership interests in Carbon Minds, a company that supplies life cycle assessment (LCA) databases used, among others, in this work to validate the CRYSTAL model. The remaining other authors declare no known competing financial interests or personal relationships that could have influenced the work reported in this paper.

Acknowledgements

TotalEnergies OneTech is acknowledged for partial funding through project CT0072000. S.M. acknowledges support from the Swiss National Science Foundation under Grant No. PZ00P2_202117. We thank Francesco Lombardi for his time and discussions regarding the implementation of SPORES.

References

[1] A. Ajanovic, M. Sayer, R. Haas, On the future relevance of green hydrogen in europe, Applied Energy 358 (2024) 122586.

[2] S. Wurbs, P. Stöcker, J. Gierds, C. Stemmler, M. Fischedick, H.-M. Henning, E. Matthies, K. Pittel, J. Renn, D. U. Sauer, et al., How important will hydrogen be in the energy system of the future?, Series on Energy Systems of the Future (ESYS) (2024). doi:10.48669/esys_2024-7.

[3] W. D. Nordhaus, An optimal transition path for controlling greenhouse gases, Science 258 (1992) 1315–1319.

[4] A. S. Manne, Eta: A model for energy technology assessment, The Bell Journal of Economics (1976) 379–406.

[5] W. D. Nordhaus, Economic growth and climate: the carbon dioxide problem, The American Economic Review 67 (1977) 341–346.

[6] L. G. Fishbone, H. Abilock, Markal, a linear-programming model for energy systems analysis: Technical description of the bnl version, International journal of Energy research 5 (1981) 353–375.

[7] J. Hoersch, F. Hofmann, D. Schlachtberger, T. Brown, Pypsa-eur: An open optimisation model of the european transmission system, Energy Strategy Reviews 22 (2018) 207–215. doi:10.1016/j.esr.2018.08.012. arXiv:1806.01613.

[8] G. Limpens, S. Moret, H. Jeanmart, F. Maréchal, Energyscope td: A novel open-source model for regional energy systems, Applied Energy 255 (2019) 113729.

[9] E. Trutnevyte, Does cost optimization approximate the real-world energy transition?, Energy 106 (2016) 182–193.

[10] G. J. Stigler, The cost of subsistence, Journal of farm economics 27 (1945) 303–314.

[11] E. D. Brill Jr, The use of optimization models in public-sector planning, Management Science 25 (1979) 413–422.

[12] J. F. DeCarolis, Using modeling to generate alternatives (mga) to expand our thinking on energy futures, Energy Economics 33 (2011) 145–152.

[13] M. Lau, N. Patankar, J. D. Jenkins, Measuring exploration: Review and systematic evaluation of modelling to generate alternatives methods in macro-energy systems planning models, Environmental Research: Energy 1 (2024) 045004.

[14] A. Grochowicz, K. van Greevenbroek, F. E. Benth, M. Zeyringer, Intersecting near-optimal spaces: European power systems with more resilience to weather variability, Energy Economics 118 (2023) 106496.

[15] J. Price, I. Keppo, Modelling to generate alternatives: A technique to explore uncertainty in energy-environment-economy models, Applied energy 195 (2017) 356–369.

[16] J. F. DeCarolis, S. Babaee, B. Li, S. Kanungo, Modelling to generate alternatives with an energy system optimization model, Environmental Modelling & Software 79 (2016) 300–310.

[17] T. T. Pedersen, M. Victoria, M. G. Rasmussen, G. B. Andresen, Modeling all alternative solutions for highly renewable energy systems, Energy 234 (2021) 121294.

[18] M. Bröchin, B. Pickering, T. Tröndle, S. Pfenninger, Harder, better, faster, stronger: understanding and improving the tractability of large energy system models, Energy, sustainability and society 14 (2024) 27.

[19] E. D. Brill Jr, S.-Y. Chang, L. D. Hopkins, Modeling to generate alternatives: The hsj approach and an illustration using a problem in land use planning, Management Science 28 (1982) 221–235.

[20] S.-Y. Chang, E. D. Brill Jr, L. D. Hopkins, Use of mathematical models to generate alternative solutions to water resources planning problems, Water Resources Research 18 (1982) 58–64.

[21] P. B. Berntsen, E. Trutnevyte, Ensuring diversity of national energy scenarios: Bottom-up energy system model with modeling to generate alternatives, Energy 126 (2017) 886–898.

[22] F. Neumann, T. Brown, The near-optimal feasible space of a renewable power system model, Electric Power Systems Research 190 (2021) 106690.

[23] F. Neumann, T. Brown, Broad ranges of investment configurations for renewable power systems, robust to cost uncertainty and near-optimality, Iscience 26 (2023).

[24] L. Nacken, F. Krebs, T. Fischer, C. Hoffmann, 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.

[25] H. Schwaeppe, M. S. Thams, J. Walter, A. Moser, Finding better alternatives: Shadow prices of near-optimal solutions in energy system optimization modeling, Energy 292 (2024) 130558.

[26] S.-Y. Chang, E. D. Brill Jr, L. D. Hopkins, Efficient random generation of feasible alternatives: a land use example, Journal of Regional Science 22 (1982) 303–314.

[27] J.-P. Sasse, E. Trutnevyte, Distributional trade-offs between regionally equitable and cost-efficient allocation of renewable electricity generation, Applied Energy 254 (2019) 113724.

[28] F. Lombardi, B. Pickering, S. Pfenninger, What is redundant and what is not? computational trade-offs in modelling to generate alternatives for energy infrastructure deployment, Applied Energy 339 (2023) 121002.

[29] F. Lombardi, B. Pickering, E. Colombo, S. Pfenninger, Policy decision support for renewables deployment through spatially explicit practically optimal alternatives, Joule 4 (2020) 2185–2207.

[30] A. de Tomás-Pascual, L. À. Pérez-Sánchez, M. Sierra-Montoya, F. Lombardi, S. Pfenninger-Lee, I. Campos, C. Madrid-López, The optimal is not always the best: Life cycle impacts of near-optimal energy systems, Applied Energy 399 (2025) 126487.

[31] A. Blum, J. Hopcroft, R. Kannan, Foundations of data science, Cambridge University Press, 2020.

[32] H. Schricker, B. Schuler, C. Reinert, N. von der Aßen, Gotta catch'em all: Modeling all discrete alternatives for industrial energy system transitions, arXiv preprint arXiv:2307.10687 (2023).

[33] M. Lau, X. Wang, N. Patankar, J. D. Jenkins, Modelling to generate continuous alternatives: Enabling real-time feasible portfolio generation in convex planning models, arXiv preprint arXiv:2411.16887 (2024).

[34] C. B. Barber, D. P. Dobkin, H. Huhdanpaa, Qhull: Quickhull algorithm for computing the convex hull, Astrophysics Source Code Library (2013) ascl–1304.

[35] S. P. Boyd, L. Vandenberghe, Convex optimization, Cambridge university press, 2004.

[36] H. Berbee, C. Boender, A. Rinnooy Ran, C. Scheffer, R. L. Smith, J. Telgen, Hit-and-run algorithms for the identification of nonredundant linear inequalities, Mathematical Programming 37 (1987) 184–207.

[37] R. L. Smith, Efficient monte carlo procedures for generating points uniformly distributed over bounded regions, Operations Research 32 (1984) 1296–1308.

[38] D. E. Kaufman, R. L. Smith, Direction choice for accelerated convergence in hit-and-run sampling, Operations Research 46 (1998) 84–95.

[39] J. F. Bard, Practical bilevel optimization: algorithms and applications, volume 30, Springer Science & Business Media, 2013.

[40] S. Pineda, H. Bylling, J. M. Morales, Efficiently solving linear bilevel programming problems using off-the-shelf optimization software, Optimization and Engineering 19 (2018) 187–211.

[41] T. Kleinert, M. Schmidt, Computing feasible points of bilevel problems with a penalty alternating direction method, INFORMS Journal on Computing 33 (2021) 198–215.

[42] G. Kamenev, A class of adaptive algorithms for the approximation of convex bodies by polyhedra, Zh. Vychisl. Mat. Mat. Fiz 32 (1992) 136–152.

[43] D. E. McClure, R. A. Vitale, Polygonal approximation of plane convex bodies, J. Math. Anal. Appl 51 (1975) 326–358.

[44] A. Lotov, Generalized reachable sets method in multiple criteria problems, in: Methodology and Software for Interactive Decision Support: Proceedings of the International Workshop Held in Albena, Bulgaria, October 19–23, 1987, Springer, 1989, pp. 65–73.

[45] R. V. Efremov, G. K. Kamenev, Properties of a method for polyhedral approximation of the feasible criterion set in convex multiobjective problems, Annals of Operations Research 166 (2009) 271–279.

[46] C. Sung, C. T. Maravelias, An attainable region approach for production planning of multiproduct processes, AIChE Journal 53 (2007) 1298–1315.

[47] R. Fourer, D. M. Gay, B. W. Kernighan, Ampl: A mathematical programming language, Management Science 36 (1990) 519–554.

[48] Gurobi Optimization, LLC, Gurobi Optimizer Reference Manual, 2024. URL: https://www.gurobi.com.

Appendix A. Appendix

Appendix A.1. MGA algorithms

In this section, we briefly review the implementation of the MGA algorithms used in the comparisons of the paper. Note that as exact implementations do differ between papers, we have implemented typical versions of these algorithms. These algorithms all generate a matrix of points $\mathcal{Z}_{points} = [z_1^T, \ldots, z_m^T]$, where each $z \in \mathbb{R}^{n_z}$ is a vector of the exploratory variables. At each iteration, k, the weighting vector $w_k \in \mathbb{R}^{n_z}$ is specified, and problem (4) is solved to yield solution z_k. The algorithms differ on how w_k is chosen at each iteration.

Hop-Skip-Jump (HSJ): [11–13, 16, 19, 20]

HSJ assigns weights based on how often a technology is used in the previous iterations. For this purpose, an element-wise indicator function, $\mathbf{1}_e$, equation (A.1c) is used to define weights. This function returns a vector with elements that are one if the input element is non-zero, and zero otherwise. At each iteration weights are determined by:

$$w_1 = \mathbf{1}_e(z_0^*) \tag{A.1a}$$

$$w_k = w_{k-1} + \mathbf{1}_e(z_{k-1}^*), \qquad k = 2, \ldots, m \tag{A.1b}$$

$$\mathbf{1}_e(z) = \begin{bmatrix} 1 \ if \ z^{(1)} \neq 0, \ else \ 0 \\ \vdots \\ 1 \ if \ z^{(n_z)} \neq 0, \ else \ 0 \end{bmatrix} \tag{A.1c}$$

HSJ-rel is a variation of HSJ where weights are updated by dividing the solution vector by a vector of the maximal values of each element:

$$w_1 = \frac{z_0^*}{z^{max}} \tag{A.2a}$$

$$w_k = w_{k-1} + \frac{z_{k-1}^*}{z^{max}}, \qquad k = 2, \ldots, m \tag{A.2b}$$

where z^{max} is a vector of the maximum values of the exploratory variables.

Random (vector) MGA [13, 21]

At each iteration, a vector of weights is generated with elements randomly between -1 and 1.

Variable Min/Max (VMM): [13, 22–25]
VMM first determines the maximum and/or minimum possible usage of each technology within the near-optimal budget. A naive implementation requires solving at least $2n_z$ MGA problems; however, as the upper and lower bounds of the variables are normally known, the minimization or maximization of a variable can be skipped if it achieves its lower or upper bound in an earlier iteration. After this phase, we assign weights randomly, as in Random. Note that some other implementations of VMM stop after the initial phase.

Efficient Random Generation (ERG): [9, 26, 27]
At each iteration, a random number of technologies is chosen to be randomly assigned a weight of -1 or 1. All other technologies are assigned a weight of zero.

Spatially explicit practically optimal alternatives (SPORES) [28–30]
SPORES was proposed for spatially resolved models, and so we have adjusted its implementation accordingly. Following feedback from the SPORES authors, we base our implementation (Algorithm 1) on the evolving-average version of SPORES from Lombardi et al. [28], while using the VMM-only initialization as in de Tomás-Pascual et al. [30]. The update step on line 17 of Algorithm 1 uses equations (A.3) to update the search direction.

$$w_j^{(s)} = \frac{\bar{z}^{(s)}}{\left|\bar{z}^{(s)} - z^{*,(s)}\right|}, \qquad j > 1 \tag{A.3a}$$

$$w_1^{(s)} = \frac{z^{*,(s)}}{z^{max}} \tag{A.3b}$$

Following Lombardi et al. [28], and input from the SPORES authors we weight the combination of the VMM and evolving average terms using $\alpha = 1$, and $\beta = 0.5$. These weights ensure that both terms are of similar magnitude, and that the VMM term does not dominate the contribution from the evolving average term.

Algorithm 1 SPORES implementation

Requires weighting parameters α and β, which we assign values of 1 and 0.5.

$i \leftarrow 1$
while $i \leq n_z$ **do** ▷ Begin each sequence with a VMM weight
  $w_0^{(i)} \leftarrow \mathbf{0}^{n_z}$
  $w_{0,i}^{(i)} \leftarrow -1$
  $w_0^{(i+n_z)} \leftarrow \mathbf{0}^{n_z}$
  $w_{0,i}^{(i+n_z)} \leftarrow 1$
  $i \leftarrow i + 1$
end while
$k \leftarrow 1$ ▷ Counter for MGA iterations
while $k \leq m$ **do**
  $s \leftarrow 1 + (k \bmod 2n_z)$ ▷ Index sequence by s
  $j \leftarrow \left\lfloor 1 + \frac{k}{2n_z} \right\rfloor$ ▷ Floored division
  if $j == 1$ **then**
    $z^{*(s)} \leftarrow \texttt{mga}(w_j^{(s)})$ ▷ Solve (4)
    $\bar{z}^{(s)} \leftarrow z^{*(s)}$
  else
    $\bar{z}^{(s)} \leftarrow \frac{(j-1)\bar{z}^{(s)} + z^{*(s)}}{j}$ ▷ Update moving average
    $w_j^{(s)} \leftarrow \texttt{update}(w_{j-1}^{(s)}, z^{*(s)}, \bar{z}^{(s)})$
    $z^{*(s)} \leftarrow \texttt{mga}(\beta w_0^{(s)} + \alpha w_j^{(s)})$ ▷ Solve (4)
  end if
  $k \leftarrow k + 1$
end while

Appendix A.2. Max-min reformulations

By selecting $p = \infty$ and expanding the norm, problem (8) can be written as the bi-level LP:

$$\min_{t, z_{\mathcal{I}}, z_{\mathcal{O}}} \quad -t \tag{A.4a}$$

$$\text{s.t.} \quad z_{\mathcal{O}} \in \mathcal{O} \tag{A.4b}$$

$$t, z_{\mathcal{I}} \in \arg\min_{t, z_{\mathcal{I}}} \quad \{t \ : \ t \leq z_{\mathcal{O}} - z_{\mathcal{I}} \leq t, \ z_{\mathcal{I}} \in \mathcal{I}\} \tag{A.4c}$$

To solve this bi-level LP, we reformulate it into a single-level MILP using the KKT conditions of the lower level problem. This is a standard reformulation described in the literature [39, 40]; however, for completeness, we briefly describe the reformulation below.

Problem (A.4) can be manipulated into the (simplified) standard form:

$$\min_{x_U, y_L} \quad -c_t^T y_L \tag{A.5a}$$

$$\text{s.t.} \quad A_U x_U \leq b_U \tag{A.5b}$$

$$y_L \in \arg\min_{y_L} \{c_t^T y_L \ : \ B_L y_L + C_L x_U \leq d_L, \ F_L y_L = e_L\} \tag{A.5c}$$

where $x_U \in \mathbb{R}^{n_{x_U}}$ are the upper-level variables, $y_L \in \mathbb{R}^{n_{y_L}}$ are the lower-level variables, the upper-level constraints are written as n_{I_U} inequality constraints with $A_U \in \mathbb{R}^{n_{I_U} \times n_{x_U}}$ and $b_U \in \mathbb{R}^{n_{I_U}}$, and the lower-level constraints are written as n_{I_L} and n_{E_L} inequality and equality constraints with $B_L \in \mathbb{R}^{n_{I_L} \times n_{y_L}}$, $C_L \in \mathbb{R}^{n_{I_L} \times n_{x_U}}$, $d_L \in \mathbb{R}^{n_{I_L}}$, $F_L \in \mathbb{R}^{n_{E_L} \times n_{y_L}}$, and $e_L \in \mathbb{R}^{n_{E_L}}$. $c_t \in \mathbb{R}^{n_{y_L}}$ is a vector used to select the element of y_L corresponding to t. We reformulate problem (A.5) by replacing the lower level problem with its KKT to yield the single-level problem:

$$\min_{x_U, y_L} \quad -c_t y_L \tag{A.6a}$$

$$\text{s.t.} \quad A_U x_U \leq b_U \tag{A.6b}$$

$$B_L y_L + C_L x_U \leq d_L \tag{A.6c}$$

$$F_L y_L = e_L \tag{A.6d}$$

$$c_t + B_L^T \lambda_L + F_L^T \eta_L = 0 \tag{A.6e}$$

$$\lambda_L \geq 0 \tag{A.6f}$$

$$\lambda_{L,i}(B_{L,i} y_L + C_{L,i} x_U - d_{L,i}) = 0 \qquad \forall i = 1 \dots n_{I_L} \tag{A.6g}$$

where $\lambda_L \in \mathbb{R}^{n_{I_L}}$ and $\eta_L \in \mathbb{R}^{n_{E_L}}$ are the dual variables corresponding to the inequality and equality constraints of the lower level problem, equation (A.6e) is the stationary condition, equation (A.6f) is the dual feasibility condition, and equation (A.6g) is the complementary constraint.

Complementary constraints are (i) inherently combinatorial and (ii) difficult to handle directly as they fail to satisfy constraint qualifications assumed by general non-linear solvers. To avoid the second issue, we reformulate equation (A.6g) using a MILP reformulation:

$$\lambda_i \leq u_i M_D \qquad \forall i = 1 \dots n_{I_L} \qquad \text{(A.7a)}$$

$$B_i y_L + C_i x_U - d_i \leq M_P(1 - u_i) \qquad \forall i = 1 \dots n_{I_L} \qquad \text{(A.7b)}$$

where M_P and M_D are sufficiently large constants that bound the primal and dual solutions [39], and $u \in \{0,1\}^{n_{I_L}}$. If $u_i = 0$, the corresponding inequality constraint is inactive, whereas if $u_i = 1$, the inequality is active.

Using information about the problem, we can tighten this reformulation by providing bounds on the sum of u. From Carathéodory's theorem, a maximum of $n_z + 1$ points are needed to interpolate any point in the convex hull. Additionally, a maximum of n_z constraints can be active due to the infinity norm. Similarly, at least one point is needed for the interpolation, and at least one of the norm inequalities must be active.

Appendix A.3. ORACLE pseudocode and variations

The basic ORACLE algorithm is given in Algorithm 2, and described in Section 3. ORACLE can be extended in various ways. Here, we briefly outline some simple, but potentially beneficial variations:

Parallelization: The version of ORACLE in Algorithm 2 is sequential due to the generation of only one trial point at each iteration. However, solution pools of a desired size can be returned when finding the distance between $\mathcal{I}$ and $\mathcal{O}$ in Step 2. These solutions are feasible points found during "normal" optimization or specially selected by the optimization software (e.g., to maximise diversity in the pool). All points in the solution pool can be evaluated in parallel to check for feasibility and to generate cuts. Thus, a partial degree of parallelisation can be easily implemented.

Combination with other MGA algorithms: All MGA approaches using the LP formulation (4) find points on the perimeter of $\mathcal{I}$, and

can also generate constraints for $\mathcal{O}$. Thus, any of these algorithms can be used within ORACLE to generate additional points and constraints. Then, ORACLE will sequentially check if convergence has been achieved (while introducing constraints), while the MGA algorithm is used to generate more points in parallel.

MILP heuristics Step 2 requires solving an MILP, which can be computationally expensive. Thus, to further reduce the computational effort, one could accurately solve problem (8) only every n iterations, and instead use local methods for the intermediate iterations. These local methods are only heuristics; however, they often perform well [39, 41] and will generate valid trial points for Step 3. However, we note that as system models can be very computationally expensive [18], Step 2 of ORACLE may not be the computational bottleneck.

Algorithm 2 ORACLE: Optimization for the Rigorous Analysis of the Complete Landscape of Alternatives
Figure 1 visually depicts the algorithm, with matching colours.

1: $\mathcal{O}, \mathcal{I}, d_{\mathcal{OI}} \leftarrow$ `init_regions` ▷ Step 1
2: **while** $d_{\mathcal{OI}} >$ `tol` **do**
3: $z^*_{\mathcal{O}},\ d_{\mathcal{OI}} \leftarrow$ `furthest_point`$(\mathcal{O}, \mathcal{I})$ ▷ Step 2
4: $z^*_f, d_{\mathcal{OZ}} \leftarrow$ `closest_near_opt`$(z^*_{\mathcal{O}})$ ▷ Step 3
5: $\mathcal{O}, \mathcal{I} \leftarrow$ `refine`$(\mathcal{O}, \mathcal{I}, z^*_f, d_{\mathcal{OZ}})$ ▷ Step 4
6: **end while**
7: $z_{designs} \leftarrow$ `sample`$(\mathcal{O}, \mathcal{I})$ ▷ Step 5

Appendix A.4. Cutting the outer approximation

We describe two approaches to cut the outer approximation (if the trial point is infeasible): the first approach (Appendix A.4.1) ensures the discovery of a strictly separating hyperplane, whereas the second approach (Appendix A.4.2) relies on a local approximation of the total cost, which requires an additional optimization. The rationale of the second approach is that if the trial point was infeasible *only* due to the near-optimality constraint, then the second approach would generate a facet of the true near-optimal region. In general, it is unlikely that only the near-optimality constraint would not

be satisfied at every iteration, and so the second approach cannot be used to guarantee convergence.

Our implementation of ORACLE uses both approaches whenever the trial point is infeasible. The additional optimization for the second approach increases the computational cost per iteration; however, adding two cuts can reduce the total number of iterations (and hence total computational cost) for convergence[11]. In general, the computational trade-off between additional effort per iteration and the total number of iterations depends on the system under consideration and the desired tolerance.

Appendix A.4.1. A strictly separating hyperplane

A strictly separating hyperplane is given by the dual variables of equation (9b). To show that the dual variables have this property, consider the dual of problem (9):

$$\max_{\mu,\lambda_A,\eta_S,\eta_F,\lambda_v} \quad \mu^T z^*_{\mathcal{O}} + \inf_{x,z} \Big(\lambda_A^T (Ax - b) + \eta_S^T (Sx - z_f) + \eta_F^T (Fx - d) + \lambda_v (c^T x - v^*(1+\epsilon)) - \mu^T z_f \Big) \tag{A.8a}$$

$$\text{s.t.} \quad \lambda_A \geq 0, \quad \lambda_v \geq 0, \quad \|\mu\|_{p^*} \leq 1 \tag{A.8b}$$

where $\|\cdot\|_{p*}$ is the dual norm[12] of $\|\cdot\|_p$, and $\mu \in \mathbb{R}^{n_z}$ is the dual variable of (9b), and $\lambda_A, \eta_S, \eta_F, \lambda_v$ are the other dual variables.

Suppose that p is chosen so that problem (9) is convex, and that strong duality holds. Let $\mu^*, \lambda_A^*, \eta_S^*, \eta_F^*, \lambda_v^*$ correspond to a dual optimum solution, with corresponding primal solution z_f^*, x^*. If $z^*_{\mathcal{O}} \neq z_f^*$, the optimal objective of problem (9) is positive. By strong duality, the optimum value of equation (A.8) is equal to the primal optimum value and hence also positive. If the optimal dual objective is positive, then when using the optimum dual variables:

$$(\text{A.8a}) > 0 \tag{A.9a}$$

$$\implies \mu^{*T} z^*_{\mathcal{O}} - \inf_{z_f} \mu^{*T} z_f > 0 \tag{A.9b}$$

$$\implies \mu^{*T} z^*_{\mathcal{O}} - \mu^{*T} z_f > 0 \qquad \forall \text{ primal feasible } x, z_f \tag{A.9c}$$

[11] Ideally, this increase in computational cost can be partially reduced by warm-starting the second optimization.

[12] The dual of the l_1 norm is the l_∞ norm and vice versa. From Hölder's inequality the dual of the l_p is the l_q norm with $q = \frac{p}{p-1}$ for $p, q \in (1, \infty)$.

Thus, μ^* defines the normal of a hyperplane that strictly separates $z^*_{\mathcal{O}}$ and the near-optimal set.

Appendix A.4.2. Cutting the outer approximation using the total system cost

After a feasible point z^*_f has been found, one can generate a local approximation of the optimal total system cost and add this constraint to the outer approximation. The benefit of introducing this second cut is that the convergence of the outer approximation may be accelerated, and hence the total number of iterations may be reduced. Consider the parametric LP:

$$v^*_f(z) = \min_x \; c^T x \tag{A.10a}$$

$$\text{s.t. } Sx = z \tag{A.10b}$$

$$Ax \leq b \tag{A.10c}$$

$$Fx = d \tag{A.10d}$$

which finds the cost-optimal solution, subject to constraint (A.10b), which fixes the exploratory variables to some vector z, i.e., the remaining decision variables are cost-optimal.

Importantly, the value function v^*_f is a convex, piecewise linear function of the parameter z. The dual variable of constraint (A.10b), $\lambda \in \mathbb{R}^{n_z}$, can be used to construct a linear approximation of the value function, parametrized by z. Consider this linearization for the point $z = z^*_f$:

$$\hat{v}^*_f(z) \approx v^*_f(z^*_f) + \lambda^T(z^*_f)(z - z^*_f) \tag{A.11}$$

As $\lambda(z^*_f)$ defines the derivative of v^*_f with respect to z^*_f, convexity implies that $\hat{v}^*_f$ is a global under-estimator of v^*_f. Therefore the constraint:

$$v^*_f(z_f) + \lambda(z^*_f)^T(z - z^*_f) \leq v^*(1 + \epsilon) \tag{A.12}$$

is a supporting hyperplane of the near-optimal region. Additionally, if $z^*_{\mathcal{O}}$ is infeasible only due to the near-optimality constraint, the normal of this constraint is a normal of a facet of the near-optimal region.

If the system is modelled such that any z (within its upper and lower bounds) is feasible if the near-optimality constraint is neglected, z^*_f will be the cost-optimal solution, as otherwise more of the near-optimal budget could be "spent" to move z^*_f closer to $z^*_{\mathcal{O}}$, i.e. the decisions x will be cost-optimal. Thus, the primal solution of problem (9) could be used to construct the primal and dual solution of problem (A.10) without requiring the optimization

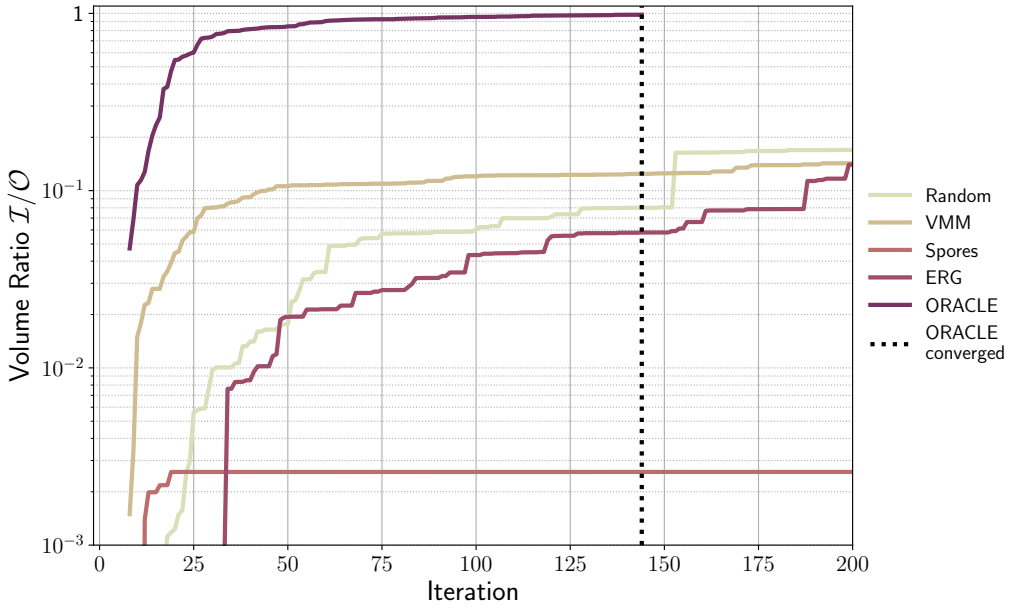


Figure A.5: Ratio of the volume of the inner and outer approximations of state-of-the-art MGA methods and ORACLE. ORACLE terminates after 144 iterations, having converged to within 0.1 GW, which corresponds to a volume ratio of 0.98.

problem to be solved. In general, the system will not necessarily be modelled in this way, and problem (A.10) will have to be solved. However, the optimizer can be warm-started by using the solution of problem (9).

Appendix A.5. Comparing convergence by volume ratio

Schricker et al. [32] propose monitoring the ratio of volumes, as illustrated in Figure A.5. This approach yields the same ranking as our proposed distance-based metric; however, the volume ratio is generally less interpretable. The same ranking is achieved as both Schricker et al. [32] and our metric use inner and outer approximations, which enables the tracking of convergence. While volume-based metrics are limited to low-dimensional cases ($n_z \lesssim 8$) [34], our proposed metric is applicable to higher-dimensional systems.

Appendix A.6. Comparing convergence to the near-optimal region

Figures 2, 3, and A.5 plot the convergence of the compared method's inner and outer approximations at iteration k, using the data generated by

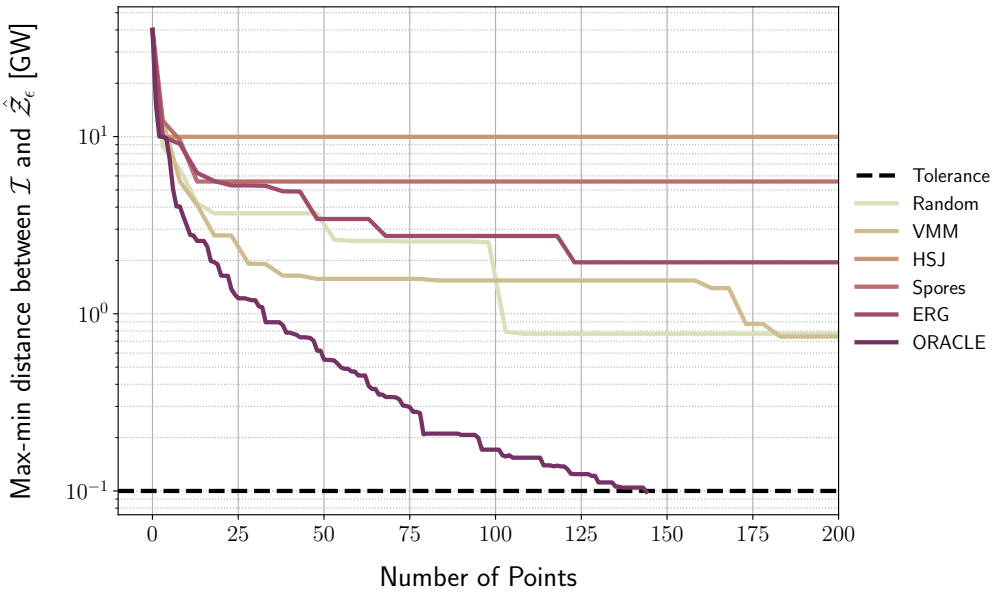


Figure A.6: Approximate distance between the true near-optimal region and the inner approximations of state-of-the-art MGA methods and the ORACLE algorithm.

that method up to iteration k. As the final outer-approximation of ORACLE has converged to within 0.1 GW of the true near-optimal region, we can use it as an approximation of the true region, $\hat{\mathcal{Z}}_\epsilon$. Thus, in a post-processing step, we can measure the distance of any method's inner approximation to the true region (Figure A.6).

Despite the inner-approximations of ORACLE, ERG, VMM and Random appearing to have converged to nearly the same value in Figure 3, Figure A.6 shows that the inner-approximations of these methods have not converged to within the specified tolerance. By 200 iterations, VMM and Random have converged to within around 0.8 GW, and ERG to 2 GW. These distances are smaller than the distances that the methods can report using only the data they generate (Figure 2). This is consistent with the finding that the outer approximations of Random, VMM, and ERG are not tight (see section 4.3).

Appendix A.7. Computational time of ORACLE before speed up

Figure A.7 shows the contribution of each step of ORACLE to the cumulative solve time. In total, ORACLE takes under 16 000 seconds (4 hours, 26 minutes) to converge. After approximately 100 iterations, Step 2, solving

the MILP, becomes the most computationally expensive contribution. However, the computational time of Steps 3 and 4 depends on the energy system model. Step 4 is computationally cheaper than Step 3 as the optimization is warm-started. The computational cost of these steps is relatively constant, as these problems do not scale with an increasing number of iterations. Additionally, note that we specified a relatively tight tolerance of 0.1 GW for the convergence of ORACLE. If a tolerance of 0.5 GW had been specified, then ORACLE would have converged after around 55 iterations, and Step 3 would have consistently been the most computationally expensive step.

Step 2 becomes the most computationally expensive step relatively early as we are using the EnergyScope model, which was specifically designed for rapid evaluation (under 60 seconds per run) [8]. Indeed, this fast run-time is specifically why the model was chosen, as we wished to compare various methods to each other. In contrast, most energy system models require significantly longer evaluation times due to more detailed representations; for example, PyPSA-Eur [7] takes around 4 hours (14 400 seconds) per optimization. The computational cost of Step 2 is primarily governed by the number of iterations (which is determined by the convergence tolerance) and the number of exploratory variables. The complexity of the model does not determine these factors. Therefore, in a study with a model similar to PyPSA-Eur, it appears reasonable to expect that the total computational burden of Step 2 will be negligible.

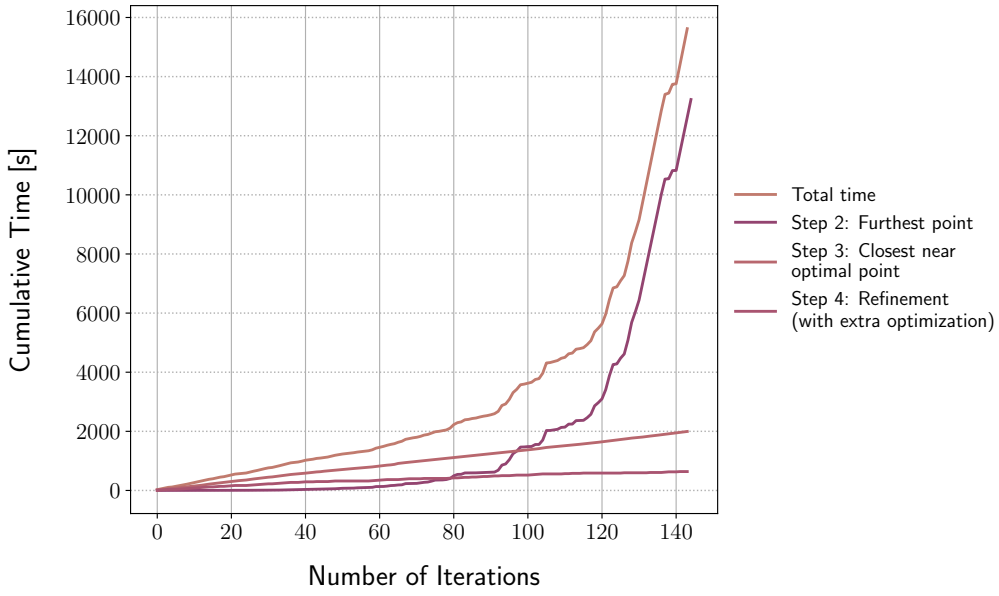


Figure A.7: Cumulative computational time of the steps of ORACLE. The time of Steps 3 and 4 depends on the model used, while the time for Step 2 is primarily determined by the desired convergence tolerance and number of exploratory variables.