Multi-Objective Mixed-Integer Evolutionary Algorithms for Building Spatial Design

Proefschrift

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van Rector Magnificus prof.mr. C.J.J.M. Stolker,
volgens besluit van het College voor Promoties
te verdedigen op woensdag 11 december 2019
klokke 10.00 uur

door Koen van der Blom

geboren te Berkel en Rodenrijs

in 1990

Promotiecomissie

Promotors:	Dr. M.T.M. Emmerich	
	Prof. Dr. T.H.W. Bäck	
Co-promotor:	Dr. H. Hofmeyer	(Technische Universiteit Eindhoven, NL)
Overige leden:	Dr. M. Baratchi	
	Prof. Dr. H.H. Hoos	(secretaris)
	Dr. C.J. Hopfe	(Loughborough University, UK)
	Prof. Dr. A. Plaat	(voorzitter)
	Dr. M. Preuß	
	Prof. Dr. B.D. de Vries	(Technische Universiteit Eindhoven, NL)
	Prof. Dr. H.A.G. Wijshoff	

This work is part of the TTW-Open Technology Programme with project number 13596, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO).

Backcover: Quote from The Rake as recorded by Ruby Throat.

Contents

Chapter 1

Introduction

1.1 Background

Darwin's description of the origin of species has not only revolutionised the field of biology, but also influenced many other disciplines. It even affected the most unexpected of fields, computer science. The relation between nature and computational machine may not be intuitive, but is certainly clear upon closer inspection. In nature, evolution is able to find the most extraordinary solutions to the problem of survival, in an unimaginably large set of possible solutions. Computers are tasked with quite similar challenges in optimisation problems. As such, researchers envisioned algorithms using the same process of evolution to find solutions to whatever problems are presented to our computers.

This mimicry of nature has proven to be a tremendously successful tool in exploring many alternative solutions, finding optimal solutions, or presenting the possible trade-offs between solutions. Yet, many challenges remain in the field of evolutionary optimisation. How can constraints be handled? How to optimise with multiple objectives in mind? How to handle mixed-variable search spaces? Although research is progressing in each of these areas, this work takes a step further by investigating the combination of the three: constraint handling, multi-objective optimisation, and mixed-variable optimisation.

As a starting point this combination is focused on the domain of the built environment. After all, what better place is there to challenge these techniques, and take them beyond their current capacities, than in the real world? The built environment is considered as a practical application area for the algorithms to optimise the build-

ings where people spend so many a waking – and sleeping – hour. Construction and exploitation of these buildings is estimated to amount to 40 % of our resource expenditure [45], and presents an opportunity for massive savings. Savings in costs, material and immaterial, ultimately reducing the negative impact on our environment.

Here it is investigated how multi-objective evolutionary computing can improve building designs, particularly in the earliest phases of the design process. During these early stages decisions are made that largely restrict the options later in the process. Changes to this fundamental design at later stages would result in having to redo a significant part of the work, making it essential to find optimal designs early on.

More specifically, this thesis focuses on spatial design, a field where all the interests in improving evolutionary computation are met. Spatial design considers the shape of a building, and its spatial organisation. In this design it is essential to consider multiple objectives. It should be structurally sound, but also energy efficient. A design also has to be feasible. This is enforced by structural constraints, that – for instance – prevent designs with rooms floating in the sky from being considered. Describing these designs to a computer requires multiple variable types, resulting in a mixed-integer search space. The width of a room can be expressed by a real number, whereas the number of rooms is necessarily integer. To handle all these challenges, a constraint handling multi-objective mixed-integer evolutionary algorithm is developed in this thesis.

1.2 Research Questions

Improving multi-objective evolutionary computing, and applying it to spatial design optimisation is no straightforward process. To achieve this vision, the following research questions are considered in this thesis.

RQ1 (Chapter 3) How can elements of the solution space be represented?

To automate building spatial design optimisation, a computer understandable representation is a must. However, defining such a representation is not straightforward. A representation easily understood and modified by a human is often difficult to handle for a computer, and vice versa.

RQ2 (Chapters 4 and 5) How can the discovery of feasible designs be ensured?

Constraints arise from physical limits (buildings cannot float in the sky), but also from practical matters such as the limitations of a simulator. Ensuring

no infeasible designs appear in a representation is challenging in itself, but can also conflict with ensuring all feasible designs can be represented. As such, the optimisation process must be able to navigate a search landscape limited by constraints, and be able to reach every feasible solution.

RQ3 (Chapter 6) How can local search contribute to the improvement of solutions found during global search in a multi-objective setting?

Identifying promising regions in the search space can be done efficiently through global search by evolutionary algorithms. Given time, they can also approach local optima. Assuming a relatively smooth surface however, traditional numerical methods may do so faster and more precisely. On the other hand, not many multi-objective numerical methods have been developed yet, and existing ones have only been evaluated on simple test problems. If these multi-objective numerical methods are applied successfully in local search, they may find better local optima.

RQ4 (Chapter 7) What can be learned about building spatial design from the optimisation process?

Optimisation processes produce large amounts of data about the problem they are tackling. This data provides an opportunity to learn about the optimisation problem, in this case building spatial design. By analysing the data, it may be possible to answer questions such as: What properties are intrinsic to an optimal building spatial design? How do optimal designs for different objectives compare? What heuristics can humans learn from the solutions found by algorithms?

RQ5 (Chapter 8) How can a generally applicable multi-objective mixed-integer algorithm be developed?

The case study of building spatial design is an illustrative example of how multi-objective mixed-integer optimisation can work. Unfortunately, algorithms specific to this problem will be inefficient, or may even not work at all, on other problems. Developing a general multi-objective mixed-integer algorithm is therefore an important next step.

RQ6 (Chapter 9) How applicable are the developed algorithms to real world problems?

Beyond the question of how the developed algorithms perform on the considered problem, it is important that they integrate well into the larger design process.

To this end, the interaction of the optimisation processes with existing design tools, and their human operators has to be investigated.

1.3 Terminology

Although this work primarily focuses on computer science aspects, there is a strong interaction with the field of the built environment. At the intersection of multiple disciplines it is important to clearly define the used terminology, in order to avoid confusion when using language common to both disciplines. For instance, *space* is used with different meanings in both fields. In computer science space commonly refers to the space of possibilities, such as the real numbers. Whereas in civil engineering space refers to a part of a building, such as a room, atrium, or hallway. To prevent this confusion, space is always preceded by an identifier when used in the computer science sense, e.g., real space, decision space, or objective space. While for civil engineering no identifier is used, e.g., the building has three spaces, or an elongated space is considered. Definitions for the large amount of technical terminology are provided in the glossary (page 183).

1.4 Notation

Per the international system of units (SI)[1] all unit names are written in lower case (e.g. kelvin, or kilogram), and unit symbols are also in lower case, except for the first letter when they are derived from a name (e.g. K, or kg).

1.5 Outline

Following on from this introduction, Chapter 2 provides the reader with background information on optimisation, multi-objective optimisation, evolutionary computation, and the basics of building spatial design. Chapter 3 then introduces a design space representation for the considered problem of building spatial design, as progressively developed over multiple published articles [16, 17, 21, 25], with the aim of answering RQ1. Furthermore, additional unpublished work is presented on alternatives and limitations of the described representation. In Chapter 4 the objectives for building spatial design are considered as individual optimisation objectives, as previously published in

[1] https://www.bipm.org/utils/common/pdf/si-brochure/SI-Brochure-9-EN.pdf

[17]. In addition, basic constraint handling with penalty functions is investigated, which is related to RQ2. Chapter 5 continues to look at constraint handling (still RQ2), but now for the multi-objective case of building spatial design, as published in [15, 16]. Since penalty based constraint handling proved to be of limited use, problem specific operators, that do not violate the constraints, are developed. Furthermore, the chapter employs landscape analysis to improve the understanding of the problem, as well as parameter tuning to maximise algorithm performance. In Chapter 6 the previously introduced problem specific multi-objective algorithm is combined with local search as published in [19] in order to answer RQ3. The optimisation data produced during the local search study is analysed in Chapter 7 (RQ4). Here, the aim is to extract design rules from the data in order to verify that the resulting spatial designs are sensible solutions, as published in [18]. Moving forward from specialised algorithms, Chapter 8 pertains to RQ5 and considers general multi-objective mixed-integer optimisation with evolution strategies, which has been published in [20]. Chapter 9 then looks at various applications of the introduced optimisation techniques for building structural and building spatial design [21, 22, 23, 24, 25], which answers RQ6. The thesis is then concluded in Chapter 10 with a discussion of the achieved results, answers to the posed research questions, and interesting directions for future studies relevant to this work.

1.6 Contributions of this Thesis

[15] Koen van der Blom, Sjonnie Boonstra, Hèrm Hofmeyer, Thomas Bäck, and Michael T. M. Emmerich. Configuring advanced evolutionary algorithms for multicriteria building spatial design optimisation. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1803–1810. IEEE, 2017.

[16] Koen van der Blom, Sjonnie Boonstra, Hèrm Hofmeyer, and Michael T. M. Emmerich. Multicriteria building spatial design with mixed integer evolutionary algorithms. In Julia Handl, Emma Hart, Peter R. Lewis, Manuel López-Ibáñez, Gabriela Ochoa, and Ben Paechter, editors, *Parallel Problem Solving from Nature – PPSN XIV*, volume 9921 of *Lecture Notes in Computer Science*, pages 453–462, Cham, 2016. Springer International Publishing.

[17] Koen van der Blom, Sjonnie Boonstra, Hèrm Hofmeyer, and Michael T. M. Emmerich. A super-structure based optimisation approach for building spatial designs. In M. Papadrakakis, V. Papadopoulos, G. Stefanou, and V. Plevris, editors, *VII European Congress on Computational Methods in Applied Sciences*

and Engineering – ECCOMAS VII, volume 2, pages 3409–3422, Athens, Greece, 2016. National Technical University of Athens.

[18] Koen van der Blom, Sjonnie Boonstra, Hèrm Hofmeyer, and Michael T. M. Emmerich. Analysing optimisation data for multicriteria building spatial design. In Kalyanmoy Deb, Erik Goodman, Carlos A. Coello Coello, Kathrin Klamroth, Kaisa Miettinen, Sanaz Mostaghim, and Patrick Reed, editors, *Evolutionary Multi-Criterion Optimization*, pages 671–682, Cham, 2019. Springer International Publishing.

[19] Koen van der Blom, Sjonnie Boonstra, Hao Wang, Hèrm Hofmeyer, and Michael T. M. Emmerich. Evaluating memetic building spatial design optimisation using hypervolume indicator gradient ascent. In Leonardo Trujillo, Oliver Schütze, Yazmin Maldonado, and Paul Valle, editors, *Numerical and Evolutionary Optimization – NEO 2017*, pages 62–86. Springer, Cham, 2018.

[20] Koen van der Blom, Kaifeng Yang, Thomas Bäck, and Michael T. M. Emmerich. Towards multi-objective mixed integer evolution strategies. In Michael T. M. Emmerich, André H. Deutz, Sander C. Hille, and Yaroslav D. Sergeyev, editors, *Proceedings LeGO – 14th International Global Optimization Workshop*, volume 2070, pages 020046–1–020046–4. AIP Publishing, 2019.

[21] Sjonnie Boonstra, Koen van der Blom, Hèrm Hofmeyer, Robert Amor, and Michael T. M. Emmerich. Super-structure and super-structure free design search space representations for a building spatial design in multi-disciplinary building optimisation. In *Electronic proceedings of the 23rd International Workshop of the European Group for Intelligent Computing in Engineering*, pages 1–10. Jagiellonian University ZPGK, 2016.

[22] Sjonnie Boonstra, Koen van der Blom, Hèrm Hofmeyer, Joost van den Buijs, and Michael T. M. Emmerich. Coupling between a building spatial design optimisation toolbox and bouwconnect BIM. In *35th CIB W78 2018 Conference: IT in Design, Construction, and Management*, pages 95–102. Springer, 2018.

[23] Sjonnie Boonstra, Koen van der Blom, Hèrm Hofmeyer, and Michael T. M. Emmerich. Combined super-structured and super-structure free optimisation of building spatial designs. In C. Koch, W. Tizani, and J. Ninić, editors, *24rd International Workshop of the European Group for Intelligent Computing in Engineering*, pages 23–34. University of Nottingham, 2017.

[24] Sjonnie Boonstra, Koen van der Blom, Hèrm Hofmeyer, and Michael T. M. Emmerich. Conceptual structural system layouts via design response grammars and evolutionary algorithms. *Automation in Construction*, pages 1–24, 2019. (submitted).

[25] Sjonnie Boonstra, Koen van der Blom, Hèrm Hofmeyer, Michael T. M. Emmerich, Jos van Schijndel, and Pieter de Wilde. Toolbox for super-structured and super-structure free multi-disciplinary building spatial design optimisation. *Advanced Engineering Informatics*, 36:86–100, 2018.

1.7 Other Work by the Author

[14] Koen van der Blom and Thomas Bäck. A new foraging-based algorithm for online scheduling. In *GECCO '18: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 53–60, New York, NY, USA, 2018. ACM.

[26] Sjonnie Boonstra, Koen van der Blom, Hérm Hofmeyer, and Michael T. M. Emmerich. Co-evolutionary design processes applied to building spatial design optimization. In *Advances in Structural and Multidisciplinary Optimization. Proceedings of the 13th World Congress of Structural and Multidisciplinary Optimization (WCSMO13)*, pages 1–6. Springer, 2020. (in print).

[105] Kaifeng Yang, Koen van der Blom, Thomas Bäck, and Michael Emmerich. Towards single- and multiobjective bayesian global optimization for mixed integer problems. In Michael T. M. Emmerich, André H. Deutz, Sander C. Hille, and Yaroslav D. Sergeyev, editors, *Proceedings LeGO – 14th International Global Optimization Workshop*, volume 2070, pages 020044–1–020044–4. AIP Publishing, 2019.

Chapter 2

Preliminaries

This chapter provides a gentle introduction to the core subjects of this thesis. It covers the basics of optimisation, multi-objective optimisation, evolutionary computation, and building spatial design.

2.1 Optimisation

In optimisation the aim is to find an optimal, or at least an improved solution, to a given problem. A problem may be that you want to design a house, but how do you decide on the properties (number of rooms, floorspace, type of isolation material) of the house? Let us assume – for the moment – that your only concern is the purchase price, which you want to be as low as possible. That is, your objective is the minimisation of the price of the house. Naturally, how much a house costs depends on the properties of the house. A larger number of rooms, for instance, may result in a higher price. Such properties may be captured in variables. Given an objective, and the variables that influence its value, an objective function can be described. In other words, a function describing how the objective value changes, given different input variables. Altogether the optimisation problem can now be described mathematically as follows. An objective function f, with a vector of input variables $\mathbf{x}$, is to be minimised. With $\mathbf{x}$ being an element of the search space $\mathcal{S}$. Here the search space is the set of all possible variable combinations. In short:

$$f(\mathbf{x}) \to \min, \quad \mathbf{x} \in \mathcal{S}. \tag{2.1}$$

2.1.1 Constraints

Although unconstrained objective functions exist, in practice most optimisation problems consider constraints. Two classes of constraints are distinguished here, namely equality and inequality constraints. For instance, for the design of a house, a specific number of rooms c_1 may be required, leading to an equality constraint of the form:

$$g(\mathbf{x}) = c_1. \tag{2.2}$$

Further, there may be a minimum requirement to the number of square metres c_2 in floorspace, resulting in an inequality constraint of the form:

$$h(\mathbf{x}) \geq c_2. \tag{2.3}$$

Finally, it is noted that a constraint might be applicable to a subset of the input variables. For instance variables $x_1, \ldots, x_5$ may be subject to constraint function g_1, while variables $x_3, \ldots, x_n$ are subject to constraint function g_2.

2.1.2 Mixed-Integer Problems

Various types of variables may be considered in optimisation problems. Objective values will generally have some ordering to them, and no major problems arise whether they are real or integer. However, decision variables may consist of different types. In this work we consider three classes of decision variables: real (also, continuous), integer (also, discrete), and nominal discrete (also, categorical). Examples for each are as follows, the height of a window could be a real variable, the number of rooms on the ground floor is an integer, and the type of material in a wall can be nominal discrete. Note that nominal discrete variables are usually encoded as integers, but by their nominal nature are not ordered. As a result, handling them differently from regular integer variables may be advantageous in an optimisation algorithm. Although both real and integer variables are ordered, they still have differing properties and therefore benefit from being treated separately as well.

2.2 Multi-Objective Optimisation

Equivalent to single-objective optimisation, multi-objective optimisation considers the minimisation (or, without loss of generality, maximisation) of objective functions. Instead of optimising one function at a time, multiple functions are optimised in concert.

That is, rather than considering $f(\mathbf{x}) \to \min$, we consider $f_1(\mathbf{x}) \to \min, \ldots, f_m(\mathbf{x}) \to \min$. Here, m indicates the number of objectives.

Beyond the basic concepts introduced in the following, the interested reader is referred to [44] for a general overview and introduction to the field of multi-objective optimisation, and state-of-the-art multi-objective evolutionary algorithms (MOEAs).

2.2.1 Pareto Optimality

In the single-objective case defining optimality is trivial: The smallest (or largest) possible value for a given objective function is the optimum. In the multi-objective case we could simply take the minimal value of each objective function, and say this is the set of optima. However, this poses a problem. Consider a bi-objective problem, with conflicting objectives. That is, optimal variables for one objective are not optimal for the other and vice versa. An optimal solution in one objective is in this case likely to be a low quality solution in the other objective. Clearly, it would be of interest to find solutions that are in between, good – but not necessarily optimal – in both individual objectives.

The Pareto (also: Edgeworth-Pareto) order makes it possible to measure optimality of such in between solutions. This order is a partial order on the solutions. That is, for some solutions it is clear which solution is preferable, but in other cases there is no strict preference for one solution over the other. Any two solutions can be compared using the notions of dominance, incomparability, and indifference. Here $\mathbf{y}$ denotes a vector of m objective values. Also note that minimisation of the objectives is considered in the following.

Definition 2.1 (Dominance)**.** Solution $\mathbf{y}^{(1)}$ is said to dominate solution $\mathbf{y}^{(2)}$ if and only if $\forall_{i\in\{1,\ldots,m\}} : y_i^{(1)} \leq y_i^{(2)}$ and $\exists_{i\in\{1,\ldots,m\}} : y_i^{(1)} < y_i^{(2)}$, in symbols $\mathbf{y}^{(1)} \prec \mathbf{y}^{(2)}$.

Definition 2.2 (Incomparability)**.** Solution $\mathbf{y}^{(1)}$ is said to be incomparable to solution $\mathbf{y}^{(2)}$ if and only if $\exists_{i\in\{1,\ldots,m\}} : y_i^{(1)} < y_i^{(2)}$ and $\exists_{i\in\{1,\ldots,m\}} : y_i^{(1)} > y_i^{(2)}$, in symbols $\mathbf{y}^{(1)} \| \mathbf{y}^{(2)}$.

Definition 2.3 (Indifference)**.** Solution $\mathbf{y}^{(1)}$ is said to be indifferent to solution $\mathbf{y}^{(2)}$ if and only if $\forall_{i\in\{1,\ldots,m\}} : y_i^{(1)} = y_i^{(2)}$, in symbols $\mathbf{y}^{(1)} \sim \mathbf{y}^{(2)}$.

Observe that indifference is equivalent to equality in the objective space. If the mapping $f : \mathcal{S} \to \mathbb{R}^m$ is considered, it is not equivalent to equality in the search space $\mathcal{S}$:

$$\mathbf{x}^{(1)} \sim \mathbf{x}^{(2)} \implies \mathbf{x}^{(1)} = \mathbf{x}^{(2)} \tag{2.4}$$

Note that these are all pairwise comparisons. In other words, they compare two solutions to each other, but do not say anything about how an individual solution compares to the set of all other solutions. For instance, identifying whether a solution $\mathbf{y}^{(1)}$ is Pareto optimal with respect to all other solutions $\mathbf{y}^{(i\neq 1)}$ requires (in the worst case) a pairwise comparison to each of them.

2.2.2 The Hypervolume Indicator

Although the Pareto order makes it possible to compare two solutions with each other, it provides no direct way to compare larger sets of (mutually incomparable) solutions. Indicators aggregate information about a set of solutions, or introduce additional preference information (on how to distribute the set across the front), but in doing so are necessarily imperfect measures. Two sets of solutions may, for instance, have an equivalent indicator value, even if they are not equivalent themselves. The hypervolume indicator [107] measures the region ($m = 2$, area; $m = 3$, volume; $m \geq 4$, hypervolume) covered by a set of points $\mathbf{Y}$, relative to a reference point $\boldsymbol{\rho}$. Given a single point this is easily done by measuring the region between this point and the reference point. With more points, things naturally get more complicated, since the different regions between the two points and the reference point may overlap. The hypervolume indicator measures the union of these two regions, rather than their sum. Further, it is defined as a general measure, for any number of dimensions. For a more detailed description of the hypervolume indicator, see [107].

Definition 2.4 (Hypervolume Indicator)**.**

$$H(\mathbf{Y}) = \Lambda_m(\cup_{\mathbf{y}\in\mathbf{Y}}[\mathbf{y}, \boldsymbol{\rho}]) \tag{2.5}$$

here Λ_m denotes the Lebesgue measure on $\mathbb{R}^m$, with m being the number of objective functions.

2.3 Evolutionary Computation

Optimisation – in the single as well as the multi-objective case – concerns the minimisation (or maximisation) of some objective function(s). Well-behaved functions can be optimised efficiently by exact methods, such as gradient-based search. However,

many functions are non-differentiable or have complex (e.g. multimodal) landscapes, and are not well suited to traditional optimisation techniques. Heuristic methods, and nature inspired methods in particular, have shown great success in optimising for such complex objective functions.

One class of nature inspired methods draws ideas from Darwinian evolution and genetics, and is termed evolutionary computation. Different branches of evolutionary computation were originally developed by independent communities. This resulted in evolution strategies [81, 88], evolutionary programming [48], and genetic algorithms [53]. Although differences exist between these branches, the core concepts are largely the same. As such, a generalised model of evolutionary computation is considered in the following.

Basically, a parent population of size μ is considered and evaluated on the target objective function. From this population, individuals are selected through mating selection. The selected parent individuals then produce λ offspring individuals through recombination. Offspring individuals are then mutated. In this manner variation is introduced into the offspring population, enabling the discovery of new solutions. These offspring individuals are evaluated, and finally survival selection determines a new parent population (again of size μ) to be used in the next generation, based on the used performance metric.

While the above represents a general outline for an evolutionary algorithm, much variation is possible. Imagine, for instance, an algorithm that only uses mutation. Furthermore, for each of the basic steps multiple variations have been introduced over time. In the case of survival selection the so called plus-strategy, and comma-strategy [89, 90] were considered. A plus-strategy, written $(\mu + \lambda)$, selects the new parent population from both the old parents and the offspring. On the other hand, the comma-strategy, written (μ, λ), selects only from the offspring (which necessitates $\lambda \geq \mu$).

2.4 Building Spatial Design

Applying optimisation techniques in building spatial design requires a basic understanding of the topic, which is introduced in the following.

Building spatial design concerns the design of the internal, and external shape of a building, subdivided into so-called spaces[1]. A space is similar to a room, but also

[1] In some fields spaces are referred to as zones

encapsulates concepts such as corridors and atria. In Figure 2.1 an example of a simple spatial design consisting of three spaces is shown.

The building spatial design is an important part of the full building design since it influences and restricts many other design decisions. In order to minimise the need for

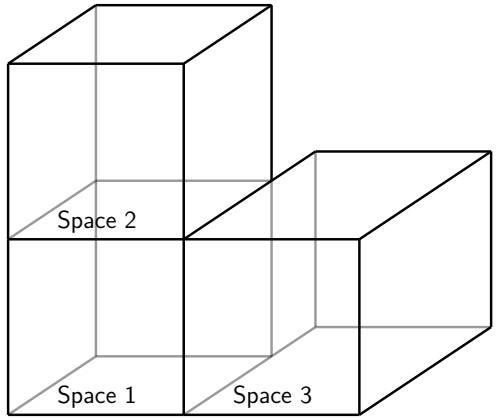


Figure 2.1: Example of a simple spatial design with three spaces.

repeated adjustments in other design components, the spatial design is more or less fixed early on. However, due to its large impact on the rest of the design process, it is of great value to find the best spatial design possible. Optimising spatial designs requires a performance metric. Here the structural performance, and the energy performance are considered.

Structural performance will be measured by the total amount of strain energy of the structural elements of the building. Strain energy is an indicator for how well loads are distributed via the different structural elements. The needed structural elements are, however, not directly available in the spatial design. As such, structural components with given properties are added to the spatial design in order to be able to measure the strain energy. Specific properties of the added components, and the procedures to arrange them, differ between experiments, and are therefore described with the experiments.

Measuring energy performance is carried out in two ways, (a) outside surface area, and (b) heating and cooling simulations. Outside surface area is cheap to compute, and serves as a proxy for energy transfer between the building and the environment in some experiments. Heating and cooling simulations make it possible to measure the required energy to maintain a comfortable temperature. These simulations give a more accurate picture since they are based on properties of the building elements, and they

take also into account internal energy transfer between spaces. As with strain energy, the measurement of heating and cooling energy also requires additional properties not available in the spatial design. Likewise, these are described along with the specific experiments. More details are available in [25].

Chapter 3

Design Space Representations

This chapter aims to answer RQ1, which asks for a problem representation for building spatial design. This is essential since, in order to optimise anything, a problem representation is needed. That is, an encoding of the problem in decision variables. Broadly, two classes of representations are considered here: Superstructure based representations, and superstructure free representations. This chapter first discusses the differences between these two types. As will become clear in this chapter, the superstructure representation is preferred here for evolutionary computation. As such, a superstructure is defined for spatial design representation, termed the supercube. Following that, limitations of the chosen representation are analysed and compared to an alternative representation. Finally, the chapter is summarised and directions for future work are discussed.

3.1 Superstructure or Superstructure Free

Algorithmic optimisation – like any optimisation process – requires a problem representation. This representation formalises the design space. In other words, the space of possible solutions. If the representation is poorly defined the optimal solution may be excluded from the design space. Or there may be so many infeasible solutions that it becomes difficult to find any acceptable solution at all. A good representation is therefore paramount to finding high quality solutions in an efficient manner. Even so, different goals may benefit from different representations. Here two classes are investigated: superstructure representations, and their counterpart, superstructure free representations.

3.1. Superstructure or Superstructure Free

A superstructure prescribes the possible solutions by encoding each of them in a vector of constant length. This results in a preselection of solutions that can be found by the optimiser, and which cannot, because they exist outside the superstructure. By limiting the design space in this manner, the optimiser can focus its search, instead of having to consider every possible alternative. On the other hand, if the global optimum is not contained in the superstructure, it will never be found. Given the fixed size of the superstructure, it may be possible to describe it as a mathematical program. If this is done the problem can be solved by standard solvers (such as described in e.g. [47]). The superstructure terminology originates from the process industry, where configurations of chemical engineering plants are optimised. For example, Jackson [56] described flow configurations in chemical reactors with a superstructure, although without explicitly mentioning the term. Various more recent works [8, 11, 92] also employ superstructures in other engineering fields.

To give an example of a superstructure, let us consider the optimisation of sunlight illumination in a building. Using binary variables, windows can be in- or excluded from the building design (where each binary variable is associated with a specific window). Assume also a fixed number of variables per window that indicate their location in the building. The maximal number of windows is fixed at the start of the optimisation process, resulting in a fixed number of binary and positioning variables. If the optimiser starts with four windows that can be turned on or off, it will find a solution with somewhere between zero and four windows. However, it will never find a solution with five windows, even if that would outperform all other solutions.

Since superstructures limit the optimisation to a predefined region of the design space, superstructure free optimisation has been suggested to be able to reach all possible solutions. Conversely to superstructures, the optimiser is now faced with the task of searching through all possibilities, but there is no risk of excluding the global optimum. Emmerich et al. [43] propose the use of replacement, insertion, and deletion rules to modify (mutate, recombine) chemical process configuration designs in evolutionary algorithms. However, the development of these modification operators requires domain knowledge. Voll et al. [96] suggest a more general framework that uses generic replacement rules in evolutionary algorithms. A similar strategy is followed in [36], where it is exemplified for the optimisation of decision diagrams. Other examples of superstructure free design spaces include the work found in [6, 62].

Many optimisation methods rely on the number of variables remaining fixed during optimisation, which is not the case for a superstructure free representation. Due to this, many optimisation methods can not handle superstructure free representa-

tions. However, algorithms such as simulated annealing, evolutionary algorithms, and heuristic local search can handle superstructure free representations. Simulated annealing has been used in the design of processes, e.g. in [35]. In the field of structural design, [59] describes a superstructure free approach in the optimisation of structural topologies. Moreover, in [52] simulations of a co-evolutionary design process (these simulations can also be interpreted as asymmetric subspace optimisation [75]) are used to find a building spatial design for which a structural design created by certain design rules shows minimal strain energy.

To provide an example of a superstructure free representation, consider again the optimisation of sunlight illumination in a building. With a superstructure free representation it is possible to only consider the fixed number of positioning variables per window. Now, the optimiser can freely add and remove sets of positioning variables to reduce or increase the number of windows. There are pitfalls, however. When a window is removed its positioning information is lost. Then, when the optimiser later adds a window, no information is retained on which positions have been tried before. With the superstructure representation some information is saved (although still only the last known position), even when a window is temporarily excluded from the design. Furthermore, an optimiser may attempt to evaluate designs with excessively many windows, whereas a human designer might realise that there is no chance that this particular design will perform well.

3.2 Supercube

Based on the definitions of superstructure and superstructure free representations it becomes clear that with a superstructure a more focused search is possible, whereas a superstructure free representation allows for a more exploratory search. A focused search makes it possible to cover a significant part of the focus area, and possibly guarantee local optimality (RQ3). Given extensive knowledge on this focus area, it may be possible to analyse the data to gain new design insights (RQ4). As such, this section proposes a superstructure representation for building spatial design, namely: the supercube.

The supercube is a superstructure that aims to encode the spaces making up a spatial design, as well as the shapes of these spaces. This is achieved by mapping each space to a 3D rectangular grid, as visualised in Figure 3.1. By using binary variables to turn different cells in this grid on or off, the shape of a space is carved out. Further, each row, column, and pillar of the 3D grid can be stretched or shrunk

with continuous variables, increasing the diversity of possible shapes. Using this 3D grid limits the spaces to be composed out of cuboid (3D rectangles) shapes. However, this also means the optimiser does not have to search through a massive number of irregular shapes until it finds something that works reasonably well.

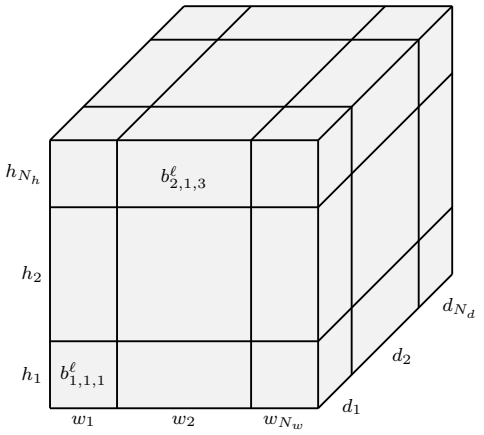


Figure 3.1: The grid used in the supercube representation, consisting of continuous width, depth and height divisions denoted by w_i, d_j, h_k, and for every space ℓ a collection of binary cells $b^{\ell}_{i,j,k}$.

More formally, the supercube is delimited by the parameters N_{spaces}, N_w, N_d, and N_h. These parameters indicate the number of spaces encoded in the supercube, and the number of segmentations in width, depth, and height respectively. Each of these is indexed as follows: $\ell \in \{1, \ldots, N_{spaces}\}$, $i \in \{1, \ldots, N_w\}$, $j \in \{1, \ldots, N_d\}$, $k \in \{1, \ldots, N_h\}$. Given these parameters, the decision variables $b^{\ell}_{i,j,k} \in \{0, 1\}$, $w_i \in \mathbb{R}^+$, $d_j \in \mathbb{R}^+$, and $h_k \in \mathbb{R}^+$ are to be optimised. A binary variable $b^{\ell}_{i,j,k}$ indicates whether the cell with indices i, j, k is active for the space with index ℓ. The continuous variables w_i, d_j, h_k denote the length of the grid segments.

3.2.1 Examples

To illustrate how the variables in the supercube can be used to encode building spatial designs a few visual examples are introduced. Note that although these examples are in 2D, everything discussed here extends to 3D.

First, Figure 3.2 shows how changes to the continuous variables of the supercube affect the building spatial design. On the left, a supercube representation is shown for which all dimensioning variables (w_i, d_j) have a value of 1. On the right, w_1 is

changed to 2, increasing the width of the whole column of cells. The dashed lines in the figure indicate that the cells in the depicted supercube are active ($b^{\ell}_{i,j,k} = 1$). Given that in this case all cells are active, and belong to a single space ($\ell = 1$, for every cell), the corresponding building spatial design has the same exterior shape as these visualisations of the supercube. However, the internal divisions are not present in the spatial design. Those merely serve to indicate the division of the supercube into cells.

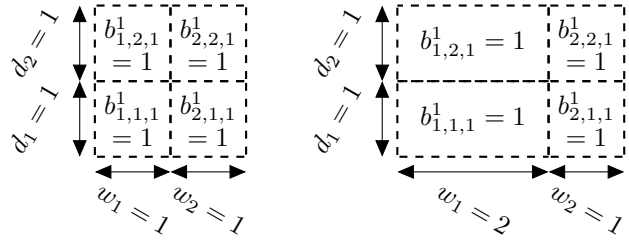


Figure 3.2: Example of continuous variation in the supercube representation.

In Figure 3.3 an example of variation resulting from the discrete variables is shown. On the left, the binary variable $b^1_{2,1,1}$ – associated with the top-right cell – is set to 1 (indicating an active cell), while on the right it is set to 0 (i.e. it is inactivate). Here, the dotted lines indicate an inactive cell ($b^{\ell}_{i,j,k} = 0$). As a result of this, the building spatial design is changed from a square-shaped form on the left, to an L-shaped form on the right. Like before, the outer contour of dashed lines correlates with the shape of the single space encoded for this particular building spatial design.

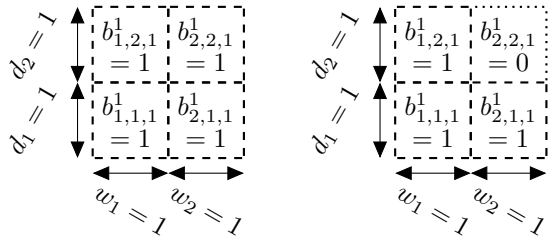


Figure 3.3: Example of discrete variation in the supercube representation.

The binary variables from the example in Figure 3.3 can be described equivalently by a 2×2 matrix (extending to 3D building spatial design naturally requires a 3D matrix). Recall that a separate bit-mask is used for each space. Here each bit-mask is denoted by $\mathbf{B}^{\ell}$, with $\ell \in 1, \ldots, N_{spaces}$. When three spaces are considered, this results

in three matrices, this is depicted in Figure 3.4. The building spatial design is drawn with solid lines, to differentiate it from the dashed and dotted lines previously used in the visualisations of the supercube. Note that, unlike in the previous figures, not all cell separators are shown. This is because those separators merely serve to visualise the cells of the supercube, whereas here a building spatial design is shown. In the building spatial design, separators only exist on their exterior. That is, only between spaces, and to the outside. Notably, there is no divisor between the left half of space 3 and its right half.

$$\mathbf{B}^1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \qquad \mathbf{B}^2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{B}^3 = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$

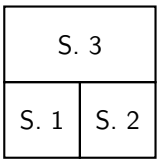


Figure 3.4: Example of multiple spaces (abbreviated by 'S.') encoded in the supercube representation.

3.2.2 Constraints on the Binary Variables

When it is not restricted by constraints, the supercube representation as described so far may lead to infeasible building spatial designs, for example floating spaces could occur. Therefore the following constraints are defined on the binary variables:

C1 There should be no overlap between spaces.

C2 All components of the building should be connected to the ground.

C3 Each space must exist, i.e. it must have at least one active cell.

C4 Each space forms a cuboid (3D rectangle) out of the cells assigned to it.

These four topology constraints are described here in mixed-integer nonlinear programming (MINLP) form, see also [17, 21, 25]. By describing the constraints in MINLP form, it is possible to use standard solvers (e.g. [47]) in combination with the supercube representation, conditional on all objective functions also being defined in MINLP form.

Constraint C1 (**No overlap**)**.** Overlap between building spaces is not allowed, because it is impractical, and may lead to erroneous results in subsequent design analysis. A constraint is needed for this because every space is represented by a separate bitmask (enumerated by ℓ) of all cells in the supercube. Thus, overlap is not automatically prevented in the representation. Equation 3.1 achieves this by taking the sum of each cell over all masks. As a result of the binary representation, only if such a sum is smaller or equal to one, no overlap exists at that position. If the bits corresponding to the same cell were active in different bit masks, the sum would be greater than one, and the overlap would be detected.

$$\forall_{i,j,k} \sum_{\ell=1}^{N_{spaces}} b_{i,j,k}^{\ell} \leq 1 \tag{3.1}$$

Constraint C2 (**Ground connected**)**.** Building spatial designs normally stand on the ground. Since this is difficult to check by a simple equation if vertical gaps are allowed in the spatial design, the constraint considered here also enforces that there are no vertical gaps. As a result of this constraint it is not possible to describe structures with cantilevers, overhangs or archways. The no vertical gaps restriction could be abandoned if one is willing to use more complex procedures to check ground connectedness constraints, but that would complicate the use of standard mathematical programming solvers. In order to enforce that every space is ground connected, and no vertical gaps exists, transitions from 0 to 1 for i, j beams are disallowed. This ensures that for every pillar (i.e. a vertical column in the supercube) all active cells have consecutive k indices, such that no vertical gaps exist. Moreover, if $b_{i,j,1}^{\ell} = 0$ no cells can be active in this beam without inducing a 0 to 1 transition, and thus violating the constraint. As such, ground connectedness is also ensured.

To check this based on the supercube variables, let $b_{i,j,k}$ be the outcome of a logical OR of all ℓ bits belonging to cell i, j, k. In equations: $\forall_{i,j,k} : b_{i,j,k} = sgn(\sum_{\ell=1}^{N_{spaces}} b_{i,j,k}^{\ell})$, where the $sgn()$ may be omitted if the previously defined no overlap constraint (Equation 3.1) is satisfied. Using $b_{i,j,k}$, if Equation 3.2 holds, the building spatial design has no vertical gaps and stands on the ground. In short, it checks that no change from 0 to 1 occurs when moving upwards in a vertical beam of cells.

$$\forall_{i,j} : \left(\sum_{k=1}^{N_h - 1} (1 - b_{i,j,k})\, b_{i,j,k+1} \right) = 0 \tag{3.2}$$

Constraint C3 (**Existence**)**.** The number of described spaces is kept constant. That is, all spaces should exist in the resulting spatial design. A building is usually designed

with a certain purpose in mind, resulting in a requirement for a specific number of spaces. With the supercube the desired number of spaces is easily specified, but that every space exists (has an active cell) requires a constraint. Checking the existence of each space is achieved by ensuring that every space is described by at least one cell. In Equation 3.3, this is achieved by simply taking the sum over all combinations of indices of a space, and checking whether this sum is at least one. I.e., at least one active cell exists for this space.

$$\forall_\ell : \sum_{i=1}^{N_w} \sum_{j=1}^{N_d} \sum_{k=1}^{N_h} b_{i,j,k}^{\ell} \geq 1 \tag{3.3}$$

Constraint C4 (Cuboid). Spaces are constrained to cuboid shapes for practicality. The restriction to cuboid spaces also helps to keep the size of the design space manageable. Consider, for instance, how the number of – especially infeasible – designs would increase when slanted structures are allowed. Searching through these to find reasonable designs would be prohibitively expensive.

To check this condition, pairwise comparisons can be made between rows, columns, and pillars of cells. In this comparison the number and position of transitions from 0 to 1, and from 1 to 0 have to be checked for equivalence. However, due to the nature of the bit-masks used in the supercube, these checks would have to be done in two directions. When comparing two rows with each other, for instance, transitions have to be checked from left to right, as well as from right to left.

In order to prevent having to check in two directions, first each bit-mask in the supercube will be extended with a single layer of cells all around it. These new cells are set to have no relation to any space ($b_{i,j,k}^{\ell} = 0$), this extension is described by Equation 3.4:

$$\begin{gathered} \forall_\ell : \forall_{i,j,k} \in \{0, \ldots, N_w + 1\} \times \{0, \ldots, N_d + 1\} \times \{0, \ldots, N_h + 1\} : \\ i = 0 \vee j = 0 \vee k = 0 \vee i = N_w + 1 \vee j = N_d + 1 \vee k = N_h + 1 \Rightarrow b_{i,j,k}^{\ell} = 0 \end{gathered} \tag{3.4}$$

Now it is possible to identify the position, and number of transitions from 0 to 1, and from 1 to 0 by checking a single direction, as shown in Figure 3.5. This process is formulated in Equation 3.5 for transitions from 0 to 1, and in Equation 3.6 for transitions from 1 to 0 (where transitions are checked from lowest to highest index value). Note that these equations serve as example for transition checks in the height, and similar equations are used for width and depth.

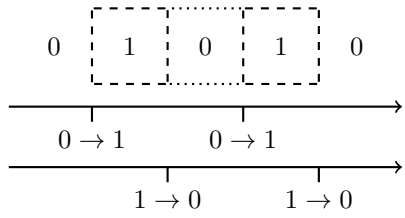


Figure 3.5: Checking transitions from 0 to 1, and from 1 to 0. Zeros without border indicate cells added by Equation 3.4.

$$\forall_\ell : \forall_{i_1,j_1,i_2,j_2} : \left(\left(\sum_{k=1}^{N_h} k\left(1 - b^\ell_{i_1,j_1,k-1}\right) b^\ell_{i_1,j_1,k}\right) - \left(\sum_{k=1}^{N_h} k\left(1 - b^\ell_{i_2,j_2,k-1}\right) b^\ell_{i_2,j_2,k}\right)\right)\left(\sum_{k=1}^{N_h} b^\ell_{i_1,j_1,k}\right)\left(\sum_{k=1}^{N_h} b^\ell_{i_2,j_2,k}\right) = 0 \quad (3.5)$$

$$\forall_\ell : \forall_{i_1,j_1,i_2,j_2} : \left(\left(\sum_{k=1}^{N_h} k\left(1 - b^\ell_{i_1,j_1,k+1}\right) b^\ell_{i_1,j_1,k}\right) - \left(\sum_{k=1}^{N_h} k\left(1 - b^\ell_{i_2,j_2,k+1}\right) b^\ell_{i_2,j_2,k}\right)\right)\left(\sum_{k=1}^{N_h} b^\ell_{i_1,j_1,k}\right)\left(\sum_{k=1}^{N_h} b^\ell_{i_2,j_2,k}\right) = 0 \quad (3.6)$$

Unfortunately, this still allows for some designs that ought to be infeasible. For example, Figure 3.6a shows how a row of zeros is allowed. This, however, makes it possible for a space to consist of multiple disconnected parts, which is not desirable. At the same time, rows of zeros have to be allowed, to ensure parts of the bit-mask can remain empty, leaving room for other spaces. In Figure 3.6b it is shown how taking this further can lead to many disconnected components of a space.

(a) A row of zeros is allowed. **(b)** A cross of zeros is allowed.

Figure 3.6: Designs that satisfy Equations 3.5 and 3.6, but should be infeasible.

To prevent these voids, and to ensure spaces are truly cuboid, Equation 3.7 is introduced. This equation enforces spaces to have a connected, ortho-convex shape. As before, this also relies on the layer of zero cells as added by Equation 3.4. With Equation 3.7 the number of changes from zero to one are counted for each space, and

for each axis. Any space where there are multiple changes from zero to one is not fully connected and therefore infeasible. In conjunction with the previous equations (3.5 and 3.6) this ensures that every space has a fully occupied cuboid shape.

$$\begin{aligned} &\forall_\ell : \\ &\forall_{i,j} : \sum_{k=0}^{N_h} \left(1 - b^\ell_{i,j,k}\right) b^\ell_{i,j,k+1} \leq 1 \\ &\forall_{i,k} : \sum_{j=0}^{N_d} \left(1 - b^\ell_{i,j,k}\right) b^\ell_{i,j+1,k} \leq 1 \\ &\forall_{j,k} : \sum_{i=0}^{N_w} \left(1 - b^\ell_{i,j,k}\right) b^\ell_{i+1,j,k} \leq 1 \end{aligned} \tag{3.7}$$

3.2.3 Constraints on the Continuous Variables

In order to compare between different building spatial designs the total volume V_0 is kept constant during optimisation. This can be achieved by a constraint on the continuous variables. Since only the active cells (i.e. cells that are encoding part of a space) are relevant to the volume computation, inactive cells have to be excluded. To do this, here (as in Constraint C2) $b_{i,j,k}$ is again taken to be the result of a logical OR over all ℓ bits of a cell i, j, k. By multiplying with $b_{i,j,k}$, the volume contribution of any cell that is inactive for all spaces ($b_{i,j,k} = 0$) will be zero. For active cells, the volume contribution is simply the product of the relevant width, depth, and height variables. Note that $b_{i,j,k}$ is used here instead of summing over all spaces with $b^\ell_{i,j,k}$ to ensure that the volume is still correct even if a cell is active for multiple spaces (and thus violates Constraint C1). Together, this yields the equality constraint in Equation 3.8 below:

$$\sum_{i=1}^{N_w} \sum_{j=1}^{N_d} \sum_{k=1}^{N_h} b_{i,j,k} w_i d_j h_k = V_0 \tag{3.8}$$

In addition, all continuous variables should be positive or taken from a range of positive values:

$$\forall_i : w_i > 0, \qquad \forall_j : d_j > 0, \qquad \forall_k : h_k > 0 \tag{3.9}$$

3.2.4 Analysis of the Supercube Search Space

To make optimal use of the supercube representation, understanding of its workings and limitations is essential. To this end, here the representation is analysed with a focus on duplicates and feasibility. Duplication is important, because having duplicates in a representation may hamper the search process. For one, evaluating the same solution multiple times is undesirable, but there may also be a problem with bias if some some solutions are encoded more frequently than others. How many designs are (in)feasible in the representation is indicative of how much time a search algorithm might spend on actually useful (feasible) solutions, versus infeasible solutions.

Duplicate Designs

Within the previously described supercube representation, duplicate building spatial designs can occur. Here a number of duplication types are discussed.

Stretch duplicates indicate duplicate designs that have different binary representations for a space, but still result in the same building spatial design due to variation of the continuous variables. An example of this is shown in Figure 3.7. By reducing the width of the two active cells in Figure 3.7a, and activating the other two cells, the representation in Figure 3.7b is reached, with an equivalent spatial design associated to it.

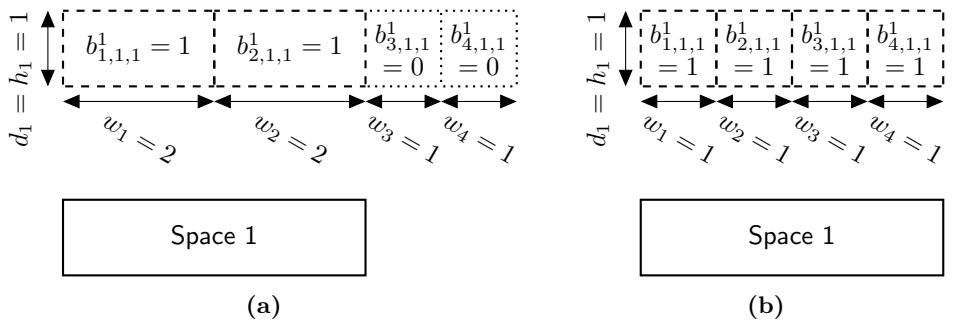


Figure 3.7: Example of stretch duplicates. Top: Supercube representations; Bottom: Spatial designs.

Swap duplicates result in equivalent building spatial designs by swapping the binary assignments of one space with those of another. In Figure 3.8 this is exemplified by swapping space 1 and 2 with each other. Although the spaces now have different identifiers, these identifiers carry no meaning, they merely serve to differentiate between spaces. As such, since the shapes of the spaces and the building stay the same,

the design is equivalent. Note that duplicates of this type are not affected by the continuous variables at all.

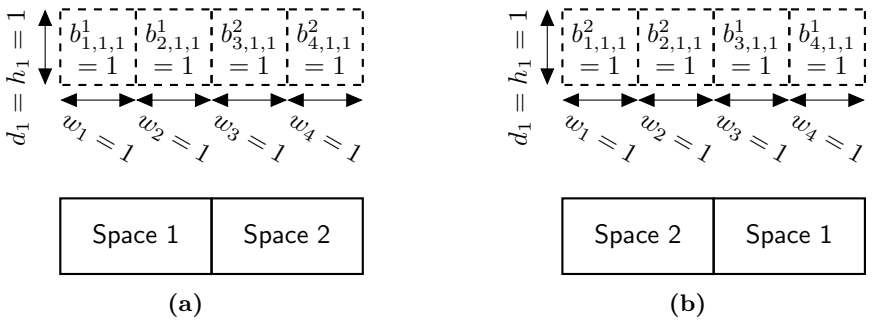


Figure 3.8: Example of swap duplicates. Top: Supercube representations; Bottom: Spatial designs.

Shift duplicates are equivalently shaped building spatial designs, encoded in a different selection of cells in the supercube. The example in Figure 3.9 shows how this can occur. Although in this case the continuous variables influence whether the design is equivalent, both binary representations can be stretched in the same way by the continuous variables (assuming all continuous variables of a dimension have the same bounds).

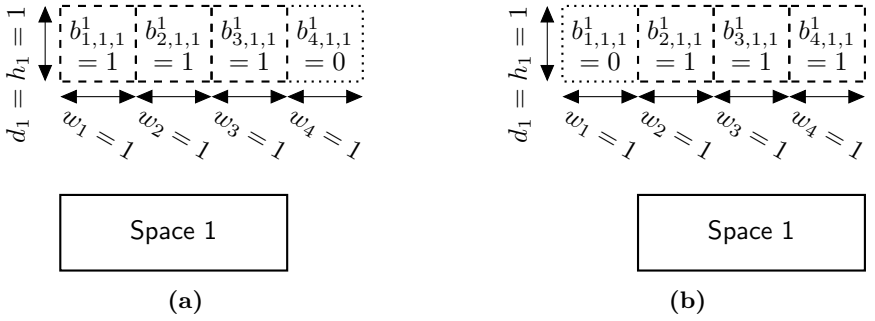


Figure 3.9: Example of shift duplicates. Top: Supercube representations; Bottom: Spatial designs.

Other types of duplication, besides those mentioned up till now, may be considered. So far all considered duplication types produce the same building spatial design. However, rotated and mirrored designs may be considered duplicates as well under certain conditions. These are duplicates in the sense that they are structurally the same, while their building spatial design differs. It is important to note that what

is or is not a duplicate is largely determined by the considered objective function(s). Rotations of a design, for example, may be equivalent in a structural sense, but differ in terms of sunlight illumination. This is also the case for wind, which may have different intensities for each direction. Since rotations and mirror images of designs will not always be considered duplicates, these duplication types are not investigated further here.

Duplicates and Feasibility

The binary representation considered for the supercube has a combinatorial search space that encodes $2^{N_{spaces} \times N_{cells}}$ designs. Here $N_{cells} := N_w \times N_d \times N_h$, i.e. the number of cells in a bit-mask of the supercube. Feasible designs (that is, designs satisfying all constraints) are duplicated exactly $N_{spaces}!$ times,[1] in the swap duplication sense. This is because for a design to be feasible, every space must consist of at least one cell (Constraint C3). As a result, cells assigned to space A may be swapped with those of space B (without losing any structure , such that space A gets the cells of space B and vice versa. Since every space has at least one cell assigned (Constraint C3), $N_{spaces}!$ such configurations of the same feasible building spatial design can be reached by swaps.

For infeasible designs the number of swap duplicates is *at most* $N_{spaces}!$. The exact number of times a specific design is duplicated depends on how many spaces have equivalent bit-masks. When the bit-masks of all spaces are distinct, $N_{spaces}!$ duplicates exist. If there are equivalent bit-masks, there are fewer duplicates. This is because swaps between equivalent bit-masks result in equivalent representations, and thus do not lead to duplicates. For example, when two spaces both have no active cells (their bit-masks are all zeros), swapping between them results in the same representation.

Due to their dependence on the continuous variables, stretch and shift duplicates are much less likely to occur, and as such not analysed in detail here.

Alternative Representation

It is clear now that the proposed supercube representation has a number of limitations. Large numbers of infeasible, as well as duplicate designs are represented. Here an alternative is investigated that replaces the binary variables with integers. This is visualised in Figure 3.10. Instead of using a binary matrix $\mathbf{B}$ for each space, a single

[1] Assuming $N_{cells} \geq N_{spaces}$. If this does not hold, no feasible designs exist.

integer matrix $\mathbf{I}$ is used. In the integer matrix every element holds the index of the space to which the corresponding cell belongs, while 0 still indicates an inactive cell.

This representation has a combinatorial search space with $(N_{spaces}+1)^{N_{cells}}$ designs, resulting in a smaller search space than the binary supercube. Notably, only designs that are infeasible because they violate Constraint C1 (no overlap) are not represented anymore.

$$\mathbf{B}^1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \qquad \mathbf{B}^2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{B}^3 = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{I} = \begin{bmatrix} 3 & 3 \\ 1 & 2 \end{bmatrix}$$

Figure 3.10: Example of integer instead of binary encoding for the supercube representation.

Although this alternative representation reduces the size of the search space, it cannot make use of the previously defined MINLP constraints. Having to find new MINLP formulations, or discarding the idea of mathematical programming altogether, makes this an alternative of limited interest. Furthermore, procedures to deal with the remaining infeasible regions are needed, regardless of using the binary or integer encoding. Moreover, all the discussed duplication types remain present in the considered alternative. A downside of the binary encoding, however, is that the space complexity grows from $O(\log_2 N_{spaces} N_w N_d N_h)$ to $O(N_{spaces} N_w N_d N_h)$, which limits the size of the buildings that can be represented.

Given these facts, the remainder of this work makes use of the binary representation. Nevertheless, studying alternative representations may make it possible to optimise more efficiently, and should be considered in the future.

3.3 Conclusions

3.3.1 Summary

In this chapter the differences between superstructure and superstructure free representations have been investigated. Superstructures allow for an extensive search for the optimum with a great variety of algorithms, within a predefined region. On the other hand, superstructure free representations make it possible to explore all options and

potentially discover surprising solutions. As such, a superstructure is more suitable for pure optimisation, while free representations allow more extensive exploration.

Based on this analysis, and the goal considered in RQ1, a superstructure representation has been defined. This representation, termed the supercube representation, allows for the description of building spatial designs with cuboid spaces. In addition, constraints on the supercube variables were defined to ensure feasibility of the designs. Here feasibility was considered both in a practical sense (what can the simulator handle), and in a realistic sense (what can actually be built).

The supercube representation was analysed to verify its suitability to be used in practice, and to understand its limitations. Based on the analysis, limiting factors to the supercube representation are the large number of infeasible designs that are represented, and the number of duplicate designs, both in the feasible and infeasible space. A considered alternative somewhat reduced the number of infeasible designs represented. However, this advantage was found to be very limited, and the alternative would also require new constraint formulations. In addition, regardless of using the supercube representation or the alternative, sophisticated constraint handling techniques would have to be developed. Due to the limited advantage and the required time investment for new constraint definitions, the supercube representation was regarded to be of greater practical use.

3.3.2 Future Work

A number of interesting research directions to follow up on the work presented here remain. For instance, there is the question whether better representations exist than those proposed here. It would be of particular interest to reduce the number of designs that need to be searched through. Defining a representation that excludes the infeasible designs from the search space would naturally greatly simplify the optimisation process. At the same time, care needs to be taken when defining such a representation to ensure none of the feasible design alternatives are lost.

Another improvement to the existing representation would be one that does not contain duplicate designs. However, beyond excluding exact duplicates, this requires careful consideration of what constitutes an equivalent design for the considered problem. Rotated designs, for instance, may be equivalent in a structural sense, but not in terms of solar illumination.

In the real world, spatial designs are not restricted to spaces with cuboid shapes as considered here. Another follow up question is thus how a representation can be

defined that removes this restriction. Moreover, can such a representation provide a toggle mechanism to choose whether it is restricted to cuboid shapes or not? This mechanism would make it possible to investigate only cuboid designs when a broader analysis is not needed, while still providing the freedom to search through all designs for use cases that require it.

An ideal representation would combine all of the aforementioned aspects. Furthermore, it could be a generic spatial design representation. One aspect in that would be the option to include or exclude certain designs, based on what is considered a duplicate for the optimisation problem under consideration. Naturally, a representation that solves all of these issues is more challenging, but each individual challenge can serve as a milestone towards it.

Given any of these suggested representations it would be valuable to still be able to define mathematical programming constraints. Although this is yet another challenge, it is a desirable property of a design space representation. Even so, depending on the representation it may simply not be possible, in which case it would be good to prove that this goal is unattainable.

Chapter 4

Basic Constraint Handling

With the definition of the supercube representation in the previous chapter, everything is now available to start optimising building spatial designs. However, it was also identified that numerous infeasible designs exist in this representation. To navigate the infeasible space, this chapter aims to evaluate constraint handling techniques to use during optimisation. This also ties in to answering RQ2, which asks for methods to effectively handle constraints, specifically to ensure feasible building spatial designs are discovered.

In order to improve the understanding of the supercube representation, the search space, and the objectives, this chapter approaches building spatial design as a single-objective problem. Based on the results it will be easier to extend to multi-objective building spatial design in the next chapter (Chapter 5). To this end an evolution strategy [82, 90] is used, and extended here with constraint handling mechanisms suitable to the considered problem.

This chapter continues in Section 4.1 with a discussion of the problem of building design, as well as a brief overview of work related to building design optimisation. Section 4.2 introduces evolution strategies, which will be used in the optimisation procedure. Then, in Section 4.3 the considered objective functions are discussed in more detail. In Section 4.4 the integration of evolution strategies with constraint handling techniques are discussed. The setup of experiments used to evaluate the approach are then described in Section 4.5. An in-depth discussion of results is then presented in Section 4.6. Finally, Section 4.7 provides a summary of the main results, and indicates directions for future work.

4.1 Building Design Optimisation

Building design is traditionally performed by architects and engineers who create solutions for discipline specific design problems. Nowadays these solutions are usually assessed by, and modified in accordance with, design analysis tools. Such tools include finite element methods (FEM) to simulate for instance structural performance or heat transfers, and computational fluid dynamics (CFD) to simulate e.g. heat, ventilation, and lighting problems. The division between the different disciplines within the field of building design also calls for tools that allow engineers from different disciplines to cooperate. An example of such a tool is computer aided design (CAD), which is used to create and share designs. However, currently building information modelling (BIM) [37] is on the rise. BIM is a method that uses data management in order to dynamically share information with other disciplines. This allows engineers to – among other things – take other disciplines into account in the early (also, conceptual) design phase. The early design phase is important for optimal building designs, because decisions in the conceptual design stage often affect performances across all disciplines. A design based on a single discipline may therefore lead to a suboptimal multi-disciplinary design.

Optimisation in the built environment is mostly performed by parametrising building components, e.g. installation type, construction type, material type, dimensions, or shapes. In [79] an overview of software tools for building optimisation is presented, followed by the introduction of a new tool. The new tool allows design variables of a building design to be selected for optimisation. Following that, an optimisation strategy can be selected. Although such tools can change and greatly improve a design, they cannot discover new designs (e.g. a new window cannot appear). Very recently, advances in early design optimisation have been made. For example, in [52] an optimisation approach inspired by the human design process is used to optimise a building spatial design for the structural performance of its related structural design. In the building physics discipline, the software tools discussed in [3, 102] are able to provide performance information for building designs. Statistical sensitivity analysis to predict the impact of design variables on the optimality of a building design is presented in [54]. This analysis is interesting for early design optimisation as the impact of each design variable in distinct design stages can be investigated.

In this chapter building optimisation for early stage building spatial design is performed using the previously defined supercube representation (Section 3.2). Although the supercube does not allow for completely free exploration of designs, it does permit

significant changes in the shape of the building. Completely free exploration will be investigated in Chapter 9, where the supercube is used in conjunction with a corresponding superstructure free representation. Here, the supercube considers a layout of building spaces that can be rearranged and resized for optimal performance. Optimisation methods are investigated for two different objective functions: (a) Structural performance, for which the compliance is to be minimised, and (b) building physics, for which the outside surface area is to be minimised. These disciplines are selected because they are known to be dependent on the building spatial design. Later in this thesis (Chapter 6) a resistance/capacitance (RC) network will be employed to analyse heating and cooling energy, it will serve as a more accurate measure of building physics performance. For the development of the optimisation method presented here minimal surface area is used as objective function because it is cheaper to compute. Details on the objective functions follow in Section 4.3.

4.2 Evolution Strategies

As outlined in the introduction evolutionary algorithms (EAs, Section 2.3) subsume different algorithms that mimic natural evolution, in order to find improved or optimised technological designs [4]. Population-based evolutionary algorithms generally work according to a basic loop structure, the so-called generational loop. It starts after an initialisation phase where an initial parent population consisting of μ individuals (solution candidates) is generated and evaluated. Then the loop begins by establishing a ranking among the individuals according to their fitness (their performance according to some objective function). Next, parents are selected to generate an offspring population (also referred to as reproduction). In this step the ranking of the population might be taken into account, although in Evolution Strategies – an important EA variant – parent individuals are chosen randomly. From the selected parent individuals, λ offspring individuals are created. Recombination is applied to allow parts of the genomes (the decision variables, possibly encoded) from multiple parents to be combined into a new genome. In order to introduce new – possibly not previously considered – information into the genome, random perturbations are applied through mutation of some of the variables in the newly produced genome. When applicable, this is followed by constraint evaluation, where invalid individuals may either be repaired, penalised, or discarded. Finally, the offspring population is evaluated on the objective function, a new parent population is produced (note that

here performance may be taken into account, unlike in the reproduction step), and the loop starts anew.

The specific type of evolutionary algorithm used in this chapter is the $(\mu + \lambda)$-Evolution Strategy. Evolution Strategies (ESs) were developed by Ingo Rechenberg and Hans-Paul Schwefel at the Technische Universität Berlin in the 1960s and are especially well suited for solving engineering design problems [82, 90]. They are interesting for this work because they are able to handle discrete as well as continuous decision variables, as outlined in [69].

In Algorithm 1 the main loop of a $(\mu + \lambda)$-ES is summarised. In short, the parent population (multiset) X_t, indexed by the number of iterations and consisting of μ individuals, is used as a template to generate the offspring population X'_t of size λ. Then, from the combination of parents and offspring the best individuals are selected as the parents of the next generation X_{t+1}. The initialisation, mutation, and recombination operators are chosen in a domain specific way, as will be discussed later in this section. For a more detailed discussion on evolution strategies and their properties the reader is referred to [4] and [12].

Algorithm 1 $(\mu + \lambda)$ Evolution Strategy [90]

```
1: t ← 0
2: X_t ← init()                                   ▷ X_t ∈ S^μ : Set of individuals
3: while t < t_max do      ▷ Generate λ solutions by (stochastic) variation operators
4:     X'_t ← generate(X_t)
5:     evaluate(Q_t)
6:     X_{t+1} ← select(X'_t ∪ X_t)                          ▷ Rank and select μ best
7:     t ← t + 1
8: end while
```

4.3 Objective Functions

Structural performance is optimised here by minimising the compliance. To compute the compliance corresponding to a building spatial design, a building structural design needs to be provided. This is carried out by applying a so-called structural grammar on each space of the building spatial design. The grammar that is used here adds four walls (slabs), with a roof (also a slab) on top. All of these are made of concrete with a thickness $t = 150\,\mathrm{mm}$ (millimetre), Young's modulus $E = 30\,000\,\mathrm{N\,mm^{-2}}$ (newton per square millimetre), and a Poisson's ratio $\nu = 0.3$. The building spatial design is then loaded with a live load of $1.8\,\mathrm{kN\,m^{-2}}$ (kilo newton per square metre)

on each floor/roof surface. Further, each outside surface is subjected to wind loads of $1.0\,\mathrm{kN\,m^{-2}}$ for pressure, $0.5\,\mathrm{kN\,m^{-2}}$ for suction, and $0.4\,\mathrm{kN\,m^{-2}}$ for shear from eight general directions (North, Northwest, West, etc.). After the transfer of the loads to the building structural design and meshing of the structure, finite element analyses are carried out to find the total compliance for all loads together. More detailed information about this procedure can be found in [52]. In summary, the first optimisation task is to minimise the total compliance (measured in newton metres), subject to the given constraints.

Next, building physics performance is optimised by minimising the total outside surface area. This objective can be computed for the supercube representation (Section 3.2) as follows. Firstly, it is required that the building spatial design contains no cantilevers, overhangs, or archways. This is achieved by Constraint C2, which ensures that no vertical gaps exist with Equation 3.2. Additionally, the computation requires a layer of cells with their binary variables equal to zero around the supercube (Equation 3.4).

To compute the outside surface area the surfaces of all outer walls, and the roof are considered. These may be found by considering all rows, columns, and pillars of the supercube. In width and depth the number of changes from zero to one are counted, and then multiplied by the area corresponding to the considered dimensioning indices. Finally, to consider both the entry and exit points, the outcome is multiplied by two. In case of the height direction the multiplication by two is omitted, because the connection with the ground layer is not counted as outside surface area. Since there are no vertical gaps in feasible designs, the height direction is essentially the sum of areas of all pillars with an active cell. The sum $S_A = S_w + S_d + S_h$ of Equations 4.1, 4.2, and 4.3 below is then the total outside surface area. Here S_w, S_d, and S_h are the total outside surface area of the width vectors (rows), depth vectors (columns), and height vectors (pillars), respectively. Note that once more, $b_{i,j,k}$ is taken as the result of a logical OR over all ℓ bits of a cell i, j, k. In summary, the second objective function is to minimise the outside surface area S_A (measured in square metres), subject to the given constraints.

$$S_w = \sum_{j=1}^{N_d} \sum_{k=1}^{N_h} \left(2 \left(\sum_{i=1}^{N_w+1} \left(1 - b_{i-1,j,k}\right) b_{i,j,k} \right) d_j h_k \right) \tag{4.1}$$

$$S_d = \sum_{i=1}^{N_w} \sum_{k=1}^{N_h} \left(2 \left(\sum_{j=1}^{N_d+1} \left(1 - b_{i,j-1,k}\right) b_{i,j,k} \right) w_i h_k \right) \tag{4.2}$$

$$S_h = \sum_{i=1}^{N_w} \sum_{j=1}^{N_d} \left(\left(\sum_{k=1}^{N_h+1} \left(1 - b_{i,j,k-1}\right) b_{i,j,k} \right) w_i d_j \right) \tag{4.3}$$

4.4 Methods

This section provides a description of how the earlier introduced $(\mu + \lambda)$-ES is customised for building spatial design optimisation with the supercube representation, and how the different constraints are handled. To this end, first a general overview is given of the procedure that will be the subject of the later presented experiments. The outline is visualised in Figure 4.1.

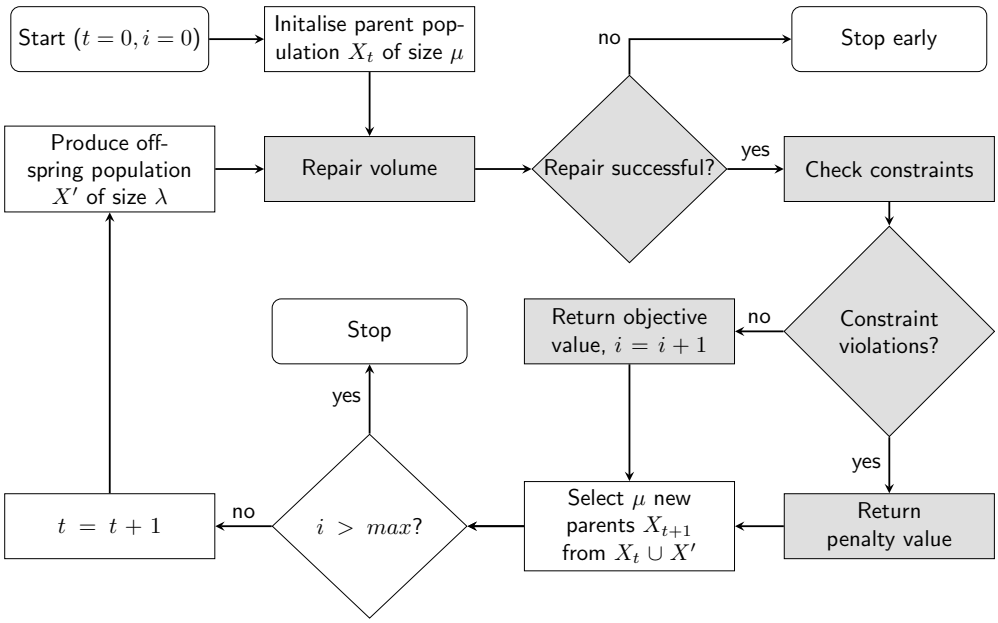


Figure 4.1: Optimisation outline. Nodes shaded in grey are performed for each individual.

The process starts by initialising the parent population X of size μ. Following this, the volume of all new individuals is repaired to be within a small margin of the desired volume V_0. If volume repair fails it is likely that incompatible settings were provided, and the process is stopped early. Next, the constraints are checked, and in case of constraint violations a penalty value is returned. If no constraints are violated, then the objective value is computed and returned instead. Moreover, the evaluation counter i is incremented. Based on these returned objective and penalty values, the

μ best individuals are selected from the parent and offspring populations. In the first iteration this will naturally always be the initial population. If the maximum number of evaluations is reached the process stops here, otherwise the iteration counter t is incremented. Finally, a new generation starts by producing an offspring population X' of size λ. The loop is then repeated by starting again from the volume repair step. This process continues until the desired number of evaluations is reached. Note that here evaluations are counted based on the number of valid (non-constraint violating) solutions.

In the following subsections a number of these processes are discussed in more detail. Specifically, domain specific ES operators (initialisation, selection, mutation, crossover), penalty functions, and repair functions will be introduced.

4.4.1 Domain Specific Operators

The ES starts by generating an initial parent population of $\mu = 20$ individuals. For continuous variables initial values are drawn uniformly at random from $[lb, \ldots, ub]$, with $lb = 3.0$, and $ub = 19.8$ (both in metres) being the lower, and respectively upper bounds for the continuous variables. Step sizes of the continuous variables are simply initialised to $\sigma = 0.1$. While binary variables are initialised to one with a probability $1/N_{cells}$, or zero otherwise. Recall that $N_{cells} = N_w \times N_d \times N_h$.

Parental selection is done by choosing two parents (possibly the same twice) uniformly at random, for each of the $\lambda = 100$ offspring individuals that are generated.

Next, intermediate crossover is applied to the continuous variables as well as their corresponding step sizes. That is, for each variable the arithmetic mean of the two parents is taken. For binary variables dominant crossover is applied by copying the value of the bit from one of the parents, chosen uniformly at random for each bit.

Mutation works as described in Algorithm 2. For convenience two definitions are introduced, the number of continuous variables $N_{cont} = N_w + N_d + N_h$, and the dimensionality of the search space $N_{dims} = N_{cont} + N_{cells} \times N_{spaces}$. Gaussian mutation with individual step sizes is applied to the continuous variables. To this end the number g_3 is drawn from a Gaussian distribution $G(\cdot, \cdot)$. Step sizes σ of the continuous variables are mutated using the Gaussian numbers g_1, g_2 (also drawn from $G(\cdot, \cdot)$), the local learning rate $\tau_1 = 1/\sqrt{2\sqrt{N_{dims}}}$, and the global learning rate $\tau_2 = 1/\sqrt{2 \times N_{dims}}$ as in [69]. Finally, binary variables are mutated by flipping each bit with a probability of $1/(N_{cells} \times N_{spaces})$, for which a number is drawn from the uniform distribution $U(\cdot, \cdot)$.

Algorithm 2 Mutate

$\tau_1 \leftarrow 1/\sqrt{2\sqrt{N_{dims}}}$ ▷ Local learning rate
$\tau_2 \leftarrow 1/\sqrt{2 \times N_{dims}}$ ▷ Global learning rate
$g_1 \leftarrow G(0,1)$
for all $i \in \{1, \ldots, N_{dims}\}$ **do**
  $g_2 \leftarrow G(0,1)$
  $g_3 \leftarrow G(0,1)$
  if $i \leq N_{cont}$ **then**
    $\sigma_i' \leftarrow \sigma_i \times \exp(g_1 \times \tau_2 + g_2 \times \tau_1)$ ▷ Mutate step size
    $x_i' \leftarrow x_i + g_3 \times \sigma_i'$ ▷ Mutate continuous variable
  else
    if $U(0,1) < 1/(N_{cells} \times N_{spaces})$ **then**
      $x_i' \leftarrow (x_i + 1) \ \% \ 2$ ▷ Mutate binary variable
    end if
  end if
end for

Following mutation, some variables may exceed their bounds. To repair them, modified interval bounds treatment is applied as in [69]. Decision variables consider the lower bounds lb and the upper bounds ub, while step size variables use the bounds $lb_s = 0.01$ and $ub_s = ub \times 0.1$.

Finally, survival selection works as is standard for evolution strategies. Namely, the μ best (lowest objective value) individuals from the $(\mu \cup \lambda)$ (parents and offspring) individuals are selected to be the parent population of the next generation, this is also referred to as elitist selection (the best/elite wins).

4.4.2 Penalties

The supercube representation considers a number of constraint functions that must hold to ensure feasible designs are produced. However, when many infeasible (constraint violating) designs exist (like in the supercube representation, see Section 3.2.4) an optimiser needs some technique to navigate towards feasible space. If no such technique is employed, in the worst case the optimiser might not find any feasible design at all.

Here two approaches are considered, which will be compared empirically in Section 4.5. First, a single fixed penalty value of $pen = 999\,999\,999$ is used whenever any constraint is violated. This value is chosen to be well beyond any realistically expected objective value to prevent any infeasible solution from being favoured over a feasible solution, but is otherwise arbitrary. The penalty provides the optimiser

with the information that the considered design is of very low quality. However, if no feasible design is found, this does not help to steer the search closer to feasible space. Even so, this still provides a baseline when comparing with other methods. Second, a graded penalty approach is used where the penalty value depends on the number of constraint violations. The idea behind this approach is that the penalty value should decrease when fewer constraints are violated. By favouring solutions that violate fewer constraints, the search will be biased towards areas closer to feasible space. Given this property, the graded penalty approach should be more suited to navigate the constraint landscape than the single penalty approach. Specifically, an infeasible solution will receive a penalty value equal to $pen + CV - 1$. Here $CV \in \{1, \ldots, 5\}$ represents the number of constraint violations, and *pen* is the same as before. Although the absolute differences between these penalty values are small, together with elitist selection this means the individual with the least constraint violations is always favoured. Note that five constraints are considered here. This concerns the four constraints described in Section 3.2.2, where Constraint C4 is split into two parts: (a) *cuboid shape* (Equations 3.5 and 3.6), and (b) *connected cuboid* (Equation 3.7).

4.4.3 Repair Functions

Given the constraint on the volume introduced by Equation 3.8, a mechanism is needed to maintain a constant volume during optimisation. Since this concerns an equality constraint on continuous variables, the use of penalty values would be less effective than for the previously discussed constraints. That is, finding solutions with exactly the right volume by the stochastic processes of an evolutionary algorithm is so unlikely that another technique is needed to handle this constraint. Here, a repair function is used to change any design that does not satisfy this constraint into one that does.

Repairing the volume works by scaling the continuous variables to satisfy a predefined desired total volume V_0 (in the following experiments $V_0 = 4^3 \times N_{cells}$. How these variables have to be scaled depends on the current total volume V_c, which is simply the outcome of the left-hand side of Equation 3.8. Given the current and desired volume, it is possible to compute the factor $\alpha = V_0/V_c$. The desired volume is then reached by multiplying the continuous variables by the cubic root of α, as shown in Equation 4.4. Note that this is only necessary for vectors (rows, columns, or pillars in the supercube) that contain at least one active cell (cells that belong to a space). Inactive cells do not influence the volume, and therefore can remain unchanged.

$$\forall_i : w'_i = \sqrt[3]{\alpha w_i} \qquad \forall_j : d'_j = \sqrt[3]{\alpha d_j} \qquad \forall_k : h'_k = \sqrt[3]{\alpha h_k} \tag{4.4}$$

Both the creation of new individuals (recombination and mutation), and the volume repair procedure may result in continuous variables that exceed their boundaries. Such boundaries result from the requirement for continuous variables to have positive values (Equation 3.9), but also from limits set for a specific design process. For instance, a lower bound lb may be set because a space that is less than half a metre wide is not practically useful. Likewise, an upper bound ub is used to avoid excessively wide/deep/high spaces. To correct for this, variables exceeding the lower bound are set to the lower bound, while variables exceeding the upper bound are multiplied by 0.95 until they are within the bound. Since these corrections affect the volume, volume repair is done iteratively together with bound corrections until both requirements are satisfied. Note that the volume requirement is considered satisfied as long as the volume remains within 1 % from the desired volume. This prevents an excessive amount of time from being spent solely on satisfying this constraint, while staying reasonably close to the requirement. The iterative process is repeated at most 26 times. If this number of repetitions is insufficient for any of the individuals the optimisation process is stopped and considered unsuccessful. The likelihood that this occurs is dependent on the chosen bounds and the desired volume, but this did not occur during any of the experiments presented in this chapter.

4.5 Experiments

Based on the described optimisation outline a number of experiments have been devised. Single objective optimisation is considered, concerning the surface area and the compliance objectives. By focussing on single-objective optimisation, and – for now – leaving out the additional complications involved in multi-objective optimisation, it is possible to purely focus on developing good constraint handling techniques. This is first done with single penalty values to investigate how well the proposed supercube representation functions in practice, and to set a baseline to compare against. Next, the same objectives are considered, but now with the graded penalty, based on the number of constraint violations. Through the comparison of these two approaches, some first insights are gained into constraint handling for this heavily constrained problem.

These experiments are all conducted with a budget of 1000 evaluations, and repeated five times. As mentioned before, only feasible candidates are evaluated, and thus only feasible candidates reduce the remaining evaluation budget. To prevent the optimisation process from going on forever (in case only infeasible solutions are found), the process is stopped after generating one million candidate solutions, even if the budget is not exhausted yet. Moreover, six different configurations of the supercube are considered to give insight into algorithm behaviour and problem characteristics for different numbers of cells and spaces. The considered configurations are: 2221, 2223, 2225, 3331, 3333, 3335. This notation has to be interpreted as follows. The first three numbers indicate the number of width, depth, and respectively height divisions in the supercube. Whereas the last number indicates how many spaces are considered. The 2221 configuration, for example, indicates a supercube that is two cells wide, deep, and high, and encodes a single space.

Settings as used in the experiments for the parameters introduced in the previous section are summarised in Table 4.1.

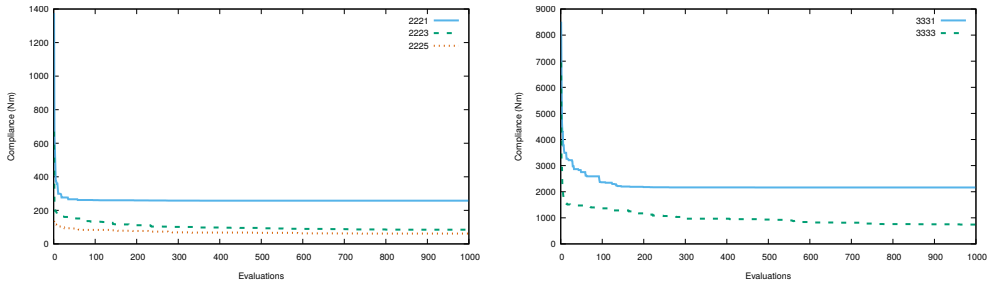
μ	λ	V_0	lb	ub	lb_s	ub_s
20	100	$4^3 \times N_{cells}$	3.0	19.8	0.01	$ub \times 0.1$

Table 4.1: Parameter settings used for the constraint handling evolution strategy.

4.6 Results

Figure 4.2 shows mean convergence plots for the compliance objective when using the single penalty approach. Note that this mean is only computed over the successful runs, i.e. runs that found 1000 feasible solutions. After a rapid decrease in the objective value during the first few hundred evaluations the optimisation process tends to stagnate. In Figure 4.2a the results for the 222x configurations are shown. Only three of the five runs for configuration 2225 were successful, so the mean of this configuration is based on only three runs. In fact, for these unsuccessful runs no feasible designs were found at all. This means that, for this problem size constraint handling is already a major problem with a single penalty value. With the larger supercube considered for the 333x configurations in Figure 4.2b this problem becomes even more apparent. In case of the 3333 configuration two out of five runs also failed to find any feasible solution, while for the 3335 configuration none of the runs found

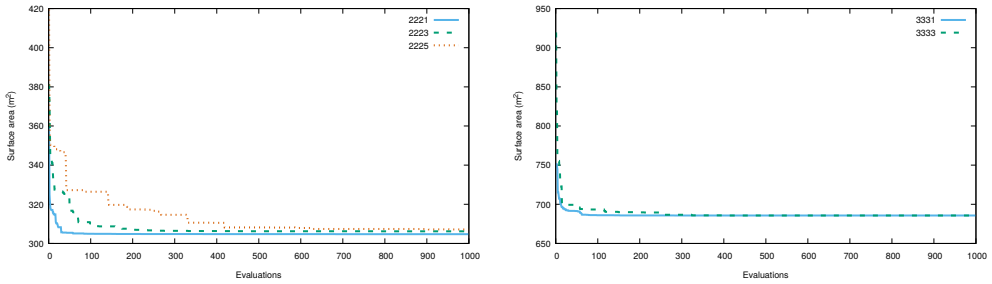
a feasible design. Evidently, a more sophisticated constraint handling technique is needed.



(a) 222x configurations. **(b)** 333x configurations.

Figure 4.2: Mean convergence of the compliance for all successful runs (maximum of five) using a single penalty value. Configurations 2225 and 3333 had 3 successful runs, while 3335 had none.

For the surface area objective similar results can be observed in Figure 4.3. Here too, a number of runs were not successful for various configurations. In Figure 4.3a configuration 2225 completed two of the five runs successfully. Further, the 3333 and 3335 configurations in Figures 4.3b respectively had three and zero successful runs. As was the case for the compliance objective, all unsuccessful runs here did not find any feasible solution at all. When comparing between the two objectives, it appears that the convergence speed is fairly similar.



(a) 222x configurations. **(b)** 333x configurations.

Figure 4.3: Mean convergence of the surface area for all successful runs (maximum of five) using a single penalty. Configurations 2225, 3333, and 3335 had 2, 3, and no successful runs respectively.

For the ease of discussion the considered constraints are briefly summarised next.

- *No overlap* (Constraint C1, Equation 3.1), each cell belongs to no more than one space.
- *Ground connected* (Constraint C2, Equation 3.2), all cells are either on the ground level, or have an active cell on the level below them.
- *Existence* (Constraint C3, Equation 3.3), all spaces exist, i.e. they are described by at least one cell.
- *Cuboid shape* (part one of Constraint C4, Equations 3.5 and 3.6), the cells describing a space together form a cuboid shape, possibly with gaps.
- *Connected cuboid* (part two of Constraint C4, Equation 3.7), all cells in the same row, column, or pillar are connected. Given that the cuboid shape constraint also holds, this means that all cells forming a space are connected and form a cuboid without voids.

Note that for some problem configurations certain constraints are always satisfied, these are indicated as not applicable (N/A). For the 2221 and 3331 problem instances, a single space has nothing to overlap with and can therefore never violate Constraint C1. The 222x instances cannot have cuboids with gaps in them, because there are not enough cells for this to occur.

Given these constraints, Table 4.2[1] shows the ratios of constraint violations for every configuration and constraint type for the minimal compliance objective when using a single penalty value. These values are computed by taking the ratio of constraint violations for a single run, and then taking the mean over the five runs. For the small supercube sizes considered here, the existence constraint does not appear to be a major problem. Although for the 3335 configuration it is already violated frequently. The likelihood of violating the no overlap constraint naturally depends on the ratio between the number of spaces and the number of cells. I.e., more spaces per cell increases the likelihood of overlap. This is supported by the results for the 2223 and 2225 configurations. Meanwhile, the probability of constraint violation for the 3335 configuration is extremely high (0.999327413), making it difficult to search

[1] The values shown in the table here differ from those originally reported in [17], because the results there mistakenly showed numbers for a single run, rather than the average over five runs. However, the primary conclusions still hold. The same goes for Tables 4.3, 4.4, and 4.5.

at all. The remaining three constraints show a similar pattern to the no overlap constraint, increasing violation probability with more spaces for both the 222x and 333x configurations, and excessively many constraint violations for the 3335 configuration.

Config.	No overlap	Ground connected	Existence	Cuboid shape	Connected cuboid
2221	N/A	0.365825317	0.058657830	0.444417496	N/A
2223	0.360701373	0.370035639	0.140828056	0.515696088	N/A
2225	0.806357513	0.735303689	0.130767034	0.784227148	N/A
3331	N/A	0.502740860	0.017993546	0.587753423	0.115546192
3333	0.479018371	0.701397235	0.054896992	0.755925962	0.481095119
3335	0.999327413	0.999974401	0.816441071	0.999885802	0.998834023

Table 4.2: Mean constraint violation probability over five runs for minimal compliance optimisation with a single penalty value for various problem configurations.

The constraint violations of the surface area in Table 4.3 show largely similar behaviour to those of the compliance objective, with a large portion of the constraints being violated with high probabilities. There are some minor variations between which constraint is violated more or less often between the two objectives. However, this can likely be attributed to chance, considering the small number of runs. Moreover, constraint violations in both cases frequently occur more than 50 % of the time. This further supports the findings from the convergence analysis, that the single penalty approach is unable to handle this heavily constrained problem.

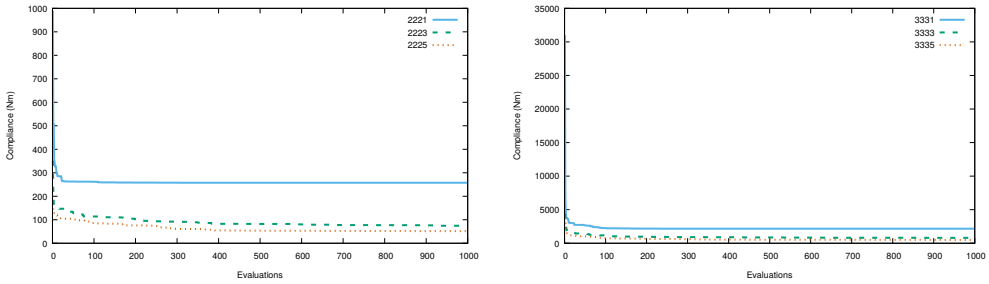
Config.	No overlap	Ground connected	Existence	Cuboid shape	Connected cuboid
2221	N/A	0.283941972	0.058233004	0.460658365	N/A
2223	0.363531813	0.363550249	0.135668651	0.510627974	N/A
2225	0.865383477	0.817330126	0.089653027	0.850689870	N/A
3331	N/A	0.486303466	0.046760850	0.524354276	0.137718976
3333	0.532956461	0.744087185	0.049183401	0.795422495	0.500961661
3335	0.999399612	0.999975601	0.816310074	0.999910402	0.998905822

Table 4.3: Mean constraint violation probability over five runs for surface area optimisation with a single penalty value for various problem configurations.

Next, it is investigated whether a graded penalty approach is able to remedy this problem. The used approach penalises based on the number of constraints that are violated. This should allow evolutionary search to gradually correct violations, and move towards feasible solutions faster.

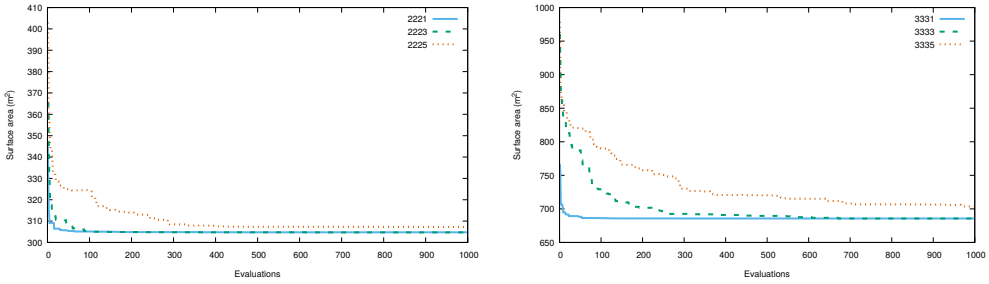
Figure 4.4a shows the mean convergence of the compliance on the 222x configurations when using the graded penalty method. Notably, all runs were completed

successfully, which shows the advantage of this method over the single penalty approach. The convergence behaviour is largely similar to the single penalty method, with quick improvements early on and stabilisation after a few hundred evaluations. For the 333x configurations shown in Figure 4.4b the results are also largely the same. Despite the fact that these configurations are more challenging than their 222x counterparts, here too all runs were successful with the graded penalty method. Evidently, the graded penalty method shows the expected and desired result of being better equipped to deal with the constraint landscape than the single penalty approach.



(a) 222x configurations **(b)** 333x configurations

Figure 4.4: Mean convergence of the compliance over five runs using graded penalty values based on the number of constraint violations.



(a) 222x configurations **(b)** 333x configurations

Figure 4.5: Mean convergence of the surface area over five runs using graded penalty values based on the number of constraint violations.

Mean convergence results of the graded penalty method for the surface area objective show similar improvements to those of the compliance objective. This holds for both the 222x configurations in Figure 4.5a, and the 333x configurations in Figure 4.5b. Furthermore, with all runs now being completed successfully, a more accurate analysis

of the convergence behaviour is possible. Particularly for the 333x configuration it is now possible to clearly observe the difference in convergence speed as the number of spaces is increased from 1, to 3, and to 5, whereas this (expected) behaviour was not clear at all with the single penalty approach (Figure 4.3b).

In terms of constraint violation probability, the results of the graded penalty method are also an improvement over those of the single penalty method. This can be observed for both the compliance objective in Table 4.4, and the surface area objective in Table 4.5. Most notably, the probabilities around 0.99 observed in many cases with the single penalty method (e.g. for compliance in Table 4.2) have disappeared. Even so, many of the constraint violation probabilities are still around or above 0.5. Especially the number of violations of the existence constraint for the 3335 configuration are cause for concern, since this suggests that this will remain a problem with larger sized designs, which often have to be dealt with in real world applications.

Config.	No overlap	Ground connected	Existence	Cuboid shape	Connected cuboid
2221	N/A	0.361342915	0.058746357	0.453979888	N/A
2223	0.330288406	0.321899619	0.153398193	0.474461674	N/A
2225	0.475043604	0.345127277	0.305617513	0.443144107	N/A
3331	N/A	0.512244896	0.019280040	0.592070113	0.107721204
3333	0.114589803	0.476313711	0.131886344	0.564315490	0.100258658
3335	0.159904807	0.407218686	0.833920127	0.474255397	0.075510223

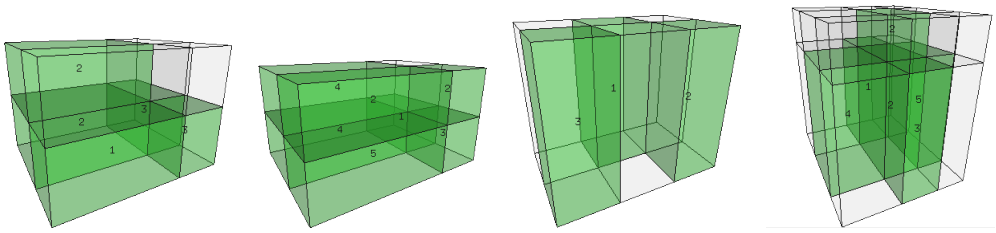
Table 4.4: Mean constraint violation probability over five runs for minimal compliance optimisation with a graded penalty value for various problem configurations.

Config.	No overlap	Ground connected	Existence	Cuboid shape	Connected cuboid
2221	N/A	0.294695023	0.065057210	0.449316221	N/A
2223	0.352604824	0.150746218	0.153169050	0.475316636	N/A
2225	0.527390468	0.389167427	0.285392221	0.455517944	N/A
3331	N/A	0.482476598	0.048915412	0.523626333	0.121148832
3333	0.130755993	0.477704137	0.144945542	0.573464583	0.106156523
3335	0.173648794	0.400757221	0.839106603	0.477907732	0.086331151

Table 4.5: Mean constraint violation probability over five runs for surface area optimisation with a graded penalty value for various problem configurations.

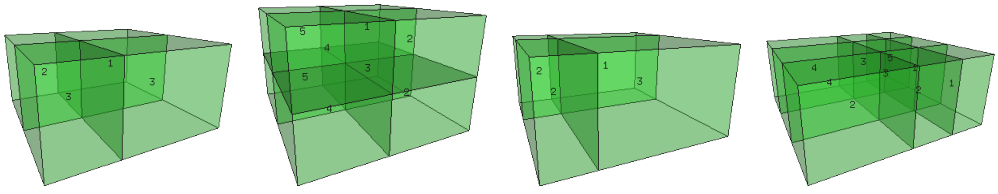
Visualisations of example results for some building spatial designs are analysed next. A first observation is that different configurations result in different spatial designs. This substantiates the need for optimisation since results from one configuration do not generalise for another configuration. There is a clear distinction between the

results from minimal compliance optimisation (Figure 4.6) and those of surface area optimisation (Figure 4.7). Surface area optimisation leads to compact cuboid, or near cuboid, shapes, as might be expected. Minimal compliance optimisation on the other hand produces a variety of shapes. Little use is made of the extra space in the 333x configurations. For the 3333 configurations this may be explained by the availability of only three spaces. There is a limited number of valid building spatial designs that can make use of the larger number of cells with only three spaces. Note that while it is possible to produce spaces consisting of a large number of cells, reaching such a situation becomes increasingly difficult with more cells while also satisfying the constraints. For example the largest space in the 3335 configuration for surface area optimisation (Figure 4.7d) consists of just two cells. This is likely to play a role in the limited use of space for both the 3333 and 3335 configurations. These frequent issues with constraints complicate exploring all feasible solutions in the search space. In particular, transitions between different feasible parts of the search space are challenging when many moves end up in infeasible parts of the search space. Evidently, this is something that needs to be addressed in future work.



(a) 2223 **(b)** 2225 **(c)** 3333 **(d)** 3335

Figure 4.6: Examples of spatial designs optimised for minimal compliance.



(a) 2223 **(b)** 2225 **(c)** 3333 **(d)** 3335

Figure 4.7: Examples of spatial designs optimised for minimal surface area.

4.7 Conclusions

4.7.1 Summary

With this chapter, the previously introduced supercube representation (Chapter 3) is subjected to first practical tests and assessed in combination with an optimisation algorithm. To this end, a brief overview of optimisation in building design, and the canonical Evolution Strategy (ES) have been presented. Furthermore, objective functions related to the two disciplines (structural, and building physics performance) considered in this thesis are also introduced.

Based on the canonical ES, and the knowledge about the number of infeasible (constraint violating) designs in the supercube representation (Chapter 3, Section 3.2.4), an ES variant has been developed to accommodate this situation. This algorithm considers variation operators (mutation, recombination) suitable to the mixed-integer nature of the supercube representation. Furthermore, penalty functions are used to deal with the constraints on the binary variables (Chapter 3, Section 3.2.2), while repair functions have been developed to satisfy the constraints on the continuous variables (Chapter 3, Section 3.2.3). With that, first steps have been taken towards answering RQ2, which calls for effective constraint handling.

This constraint handling ES is subjected to experiments comparing a single penalty method with a graded penalty method. In the single penalty method a fixed penalty value is returned regardless of the number of constraints that are violated, while the graded penalty method applies penalties that grow with the number of violated constraints. The experiments showed that, as expected, a graded penalty allows for a more effective search than a single penalty value. However, despite being more effective, constraint violations remain frequent. Due to this, it seems unlikely that larger supercube configurations than those considered here can be optimised effectively, even when using the graded penalty method. With this in mind, different constraint handling techniques have to be developed.

4.7.2 Future Work

As mentioned, although the improvements when using a graded penalty are promising, constraint violations still occur frequently, and likely prohibit the optimisation of larger building spatial designs. One direction in which the graded penalty approach could be improved is by increasing the granularity in which it penalises infeasible designs. In the version evaluated in this chapter a design that violates (for instance) the existence

constraint once, would get the same penalty value as a design that violates the same constraint multiple times (for different spaces). By penalising designs based on a more fine-grained measure of constraint violation it should be possible to indicate a smoother path towards feasible space for the search process. This removes plateaus in the penalised part of the search space, but it might also introduce local optima.

Despite the possibilities to improve on the graded penalty approach, any penalty based approach has the inherent downside of spending time on infeasible solutions (except for the special – and unlikely – case were only feasible solutions are found). An alternative that does not suffer from this drawback is the use of specialised search operators that only navigate the feasible space. Naturally, developing such operators requires careful consideration of how to navigate the feasible search space. After all, every feasible solution should still be reachable, even if there are disconnected feasible regions. The development of specialised operators for the building spatial design problem will be investigated in the next chapter.

Another issue that was identified is that many optimised solutions seem to use a limited selection of the cells in the supercube. Although this may just be the end result of the optimisation process, it could also indicate that the search has difficulty moving from one part of the feasible space to another. Considering the number of constraint violations, and the number of changes that are often needed in the binary part of the search space to transition from one feasible solution to another, this is a serious concern. Fortunately, it should be possible to address the reachability of all feasible solutions with specialised operators, as will be addressed in Chapter 5. Additionally, in future work it may be worth investigating how global optimisation strategies such as niching [93] can improve the variety of the discovered solutions.

Chapter 5

Problem Specific Constraint Handling and Multi-Objective Optimisation

In the previous chapter first steps have been made towards constraint handling techniques for building spatial design, as is required to answer RQ2. The main conclusion was that although the proposed penalty based methods were reasonably effective for the small scale designs considered in the experiments there, they are unlikely to be sufficient for the larger scale designs that are frequently needed in the real world. To resolve this issue, in this chapter problem specific operators are developed which ensure that only feasible designs are found. Furthermore, the two considered objectives (structural, and energy performance) have so far only been considered separately. Since there is an interaction between these disciplines, and they both affect the spatial design, in this chapter they will be considered in a multi-objective context.

The use of multi-objective optimisation in building design is not new, and has been shown to be effective in [54] when two objectives from the same discipline are considered. However, traditionally, energy efficiency and structural design objectives are dealt with in different engineering disciplines. Multidisciplinary optimisation aims to combine different disciplines in order to find building designs that perform well with respect to criteria from each of them. While in the building design domain the use of multidisciplinary optimisation is still limited, it has already been used with great success in areas such as automotive and aerospace engineering [70].

5.0.

This chapter contributes towards the use of multidisciplinary optimisation in building spatial design. Here the previously introduced (Section 4.3) objectives from structural design (compliance) and energy efficiency (total surface area) are considered again. By proposing a multi-objective optimisation approach, the problem of conflicting objectives is also taken into account. In this case a Pareto front of building spatial designs is computed that can be used in preparation of decision making, to understand design principles that lead to high performance in one discipline or the other discipline, and to find valid compromise solutions.

To achieve these goals (evolutionary) multi-objective optimisation algorithms, such as SMS-EMOA [39] and NSGA-II [32], are employed. Within the algorithm structure, the problem specific constraint handling operators are developed. In this manner these traditional algorithms are adapted to handle the heavily constrained, and mixed-integer search space of the supercube representation. First it is shown that the use of problem specific operators (here initialisation and mutation are considered) is indeed a promising direction by developing initial versions of them. Next, based on the positive results, these methods are developed further to navigate the search space without bias.

The unbiased initialisation operator that will be introduced should be well suited for use in landscape analysis. By using landscape analysis based on the initialisation operator, insight is gained into the objective landscape corresponding to the objective functions. Various methods for landscape analysis have been proposed in the literature, see e.g. [80, 76]. A basic approach for landscape analysis is to look at the distribution of function values over random samples, for example by density of states analysis [83]. Additionally, landscape analysis can be used to identify how variation operators behave in the objective landscape [57]. In this chapter it is applied to evaluate the improved mutation operator, and to investigate its behaviour for different step sizes on multiple problem instances.

Naturally, obtaining the optimal performance from the developed algorithms is desirable. To achieve this, parameter tuning can be used, which aims at finding the optimal settings for an algorithm to solve a problem. For instance, in [68] the tuning and configuration of an image processing pipeline in coronary vessel image analysis is considered as a mixed-integer optimisation problem. Specific algorithmic and statistical solutions for tuning also exist, e.g. SMAC [55], irace [71], and SPOT [9]. Another way to improve algorithm performance is by integrating problem knowledge into operators, which can significantly improve performance as shown for chemical process design in [84]. This is also done in this chapter, and the settings of these operators are tuned to further improve performance.

Many real world optimisation problems, such as building spatial design, are characterised by expensive evaluation functions. As a result, tuning optimisation algorithms for this type of problem involves an additional challenge. The irace package [71], based on statistical significance of configurations, and the Mixed-Integer Evolution Strategy (MIES) [69], a self-adaptive evolutionary algorithm with specialised operators for mixed-integer problems, are compared here for the tuning of the parameters of the standard SMS-EMOA [39] and a tailored version of SMS-EMOA that uses the newly introduced problem specific operators. This comparison serves three purposes. Firstly, it allows fair insight into how the tailored and standard versions of SMS-EMOA differ. Secondly, the qualities of two tuning methods, with different philosophies behind them, can be compared. Finally, while state-of-the-art tuning methods have been extensively evaluated on academic case studies, the comparison here considers a real world problem.

Given these preliminaries, this chapter contributes as follows. Initialisation and mutation operators suitable to the problem specific constraints, and the mixed-integer nature of the problem are developed. Next, first results on multi-objective optimisation of building spatial designs are provided and discussed. Following that, the problem specific mutation and initialisation operators are improved to avoid biases. Using the unbiased operators, landscape analysis of the building spatial design problem is performed to identify problem features and to investigate the behaviour of the mutation operator. Finally, parameter tuning is conducted for the considered algorithms and discussed for optimisation problems with time consuming evaluation functions. Moreover, two parameter tuning algorithms are compared and their different merits are discussed.

This chapter continues with Section 5.1, where algorithm details and the problem specific operators are introduced. Thereafter, in Section 5.2 experiments are described, followed by a discussion of the numerical results. In Section 5.3 the improved unbiased initialisation and mutation operators are described. Section 5.4 then investigates the objective landscapes for structural design and energy efficiency, as well as how the mutation operator behaves in this landscape. To compare the standard SMS-EMOA algorithm and a tailored version with the unbiased operators described in this chapter, Section 5.5 discusses parameter tuning and algorithm performance for both of these approaches. Finally, Section 5.6 summarises the chapter and discusses directions for future work.

5.1 Algorithms

This section first describes how standard multi-objective evolutionary algorithms are adapted for building spatial design by using the same techniques as applied for single objective optimisation in Chapter 4. After that, problem specific initialisation and mutation operators are introduced to replace the standard versions in an algorithm tailored to the building spatial design problem.

5.1.1 Standard Algorithms Applied to Building Spatial Design

NSGA-II [32] and SMS-EMOA [39] are both standard algorithms in evolutionary multi-objective optimisation (EMO), and will be used as a baseline in the experiments here. Both algorithms largely work according to the same principles and are described together in the following, and their differences are indicated as needed.

Initialisation works by setting binary variables to one with probability $1/N_{cells}$, or to zero otherwise. Here N_{cells} is again defined as $N_w \times N_d \times N_h$. The continuous variables are initialised to a uniformly random value in $[lb, ub]$, where $lb = 3$ and $ub = 19.8$ (both in metres). As a final step of the initialisation procedure, the volume is repaired to be within 1 % of the desired volume $V_0 = 4^3 \times N_{cells}$ (in cubic metres). Here volume repair works as previously described in Section 4.4.3.

To generate offspring an individual is selected uniformly at random from the parent population. Then recombination is applied with probability $RP = 0.5$, in which case a second parent is selected uniformly at random. Otherwise the first parent is copied to the offspring individual. Binary variables are swapped between the parents with a probability of 0.25, while simulated binary crossover [30] is applied to the continuous variables with probability 0.5. Variables that exceed their bound are set to lb or ub, depending on which one is exceeded. Finally, either of the children is selected with probability 0.5.

Mutation is applied with a probability of $MT = 1/N_{dims}$, where $N_{dims} = N_{cells} \times N_{spaces} + N_w + N_d + N_h$ is once more the total number of variables. Binary variables are mutated by bit flips, while polynomial mutation [32] is applied to continuous variables above the lower bound. Continuous variables that are exactly at the lower boundary are reinitialised (according to the previously described initialisation procedure). Following mutation, variables exceeding their bounds are set to their appropriate boundary values again, and the volume of the produced offspring is repaired as described in Section 4.4.3.

Both algorithms use a $(\mu + \lambda)$ selection strategy, where for NSGA-II $(20 + 20)$

is considered, and for SMS-EMOA $(50 + 1)$. NSGA-II does so by applying non-dominated/crowding distance sortingto the population, and then selecting the first μ individuals for the next parent population. On the other hand, SMS-EMOA selects the population of size μ that retains the largest hypervolume, using $(1.1e9, 1.1e9)$ as reference point during optimisation.

Equivalent to the process in the previous chapter, in case of constraint violations a penalty value $pen = 999\,999\,998 + CV$ is returned according to the graded penalty method (Section 4.4.2). The objective functions are only evaluated when no constraints are violated. As such, infeasible solutions do not influence the remaining evaluation budget.

5.1.2 Tailored Algorithm for Building Spatial Design

The general mutation and recombination operators used in NSGA-II and SMS-EMOA have difficulties navigating heavily constrained objective landscapes, such as considered for the building spatial design problem. To resolve this, a problem specific mutation operator is proposed which only produces mutants that do not violate the constraints. Since in preliminary experiments the algorithms have fairly similar performance, only SMS-EMOA is considered with the problem specific mutation operator.

(a) Initialisation. **(b)** Mutation.

Figure 5.1: Problem specific operators that prevent constraint violations.

Additionally, initialisation is adapted to ensure all initial solutions are feasible as well. Initialisation of the continuous variables causes no problems and is kept as described in Section 5.1.1. Binary variables are initialised by selecting a non-fully occupied pillar from the supercube for every space uniformly at random. The first cell from the bottom that does not belong to any previously initialised space is then set to active ($b^{\ell}_{i,j,k} = 1$) for this space. This process is visualised in Figure 5.1a, where spaces A, B, and C can be imagined as having been dropped – one after the other – into the supercube from above. Since the initial population resulting from this process consists solely of single-cell spaces, it is not very diverse. To resolve this, twenty mutations (as

described next) are applied to each individual in the initial population.

Having defined the initialisation procedure, a variation operator is needed to guide the search. To this end, a problem specific mutation operator is defined next. With a probability of 0.25 a binary mutation is applied, and otherwise a continuous mutation is applied. In case of a continuous mutation, polynomial mutation [32] is always applied to a single continuous variable, selected uniformly at random from those variables that are relevant to at least one active cell. In case of a binary mutation, a random space is selected and contracted or expanded by a surface of cells in a random direction as shown in Figure 5.1b (where the arrows indicate possible moves for space B). This is achieved by selecting one of the following faces of the space to make either an outward or an inward move: left, right, top, bottom, front, or back. All moves are applied to all cells along the selected face of a space, such that the space remains cuboid when adding and removing cells. These moves are of size one, meaning that the width, depth or height (depending on the selected face) of a space grows or shrinks by a single cell. For any move Constraint C3 (all spaces must have at least one cell assigned to them) and the supercube borders must be respected. In other words, any move violating these constraints cannot be chosen. From all possible mutation moves that do not result in a constraint violation one is chosen uniformly at random. To avoid issues with the no overlap constraint (C1), whenever an outward move adds a cell to a space B that is already part of another space A the cell is set to inactive ($b^{\ell}_{i,j,k} = 0$) for space A.

A new offspring individual is then created as follows. A parent is selected uniformly at random, and mutation is applied. No recombination is used, but this may be considered in future work. Although recombination may be able to aid the search, designing a problem specific recombination operator that does not violate any constraint is complicated. Moreover, at the same time as not violating constraints, it should also result in sensible new designs. Penalty values are also no longer used since all offspring is now guaranteed to be feasible. The remaining procedures (such as selection based on hypervolume contribution) are the same as before in Section 5.1.1.

5.2 Tailored Algorithm Evaluation

5.2.1 Experiments

Having adapted NSGA-II and SMS-EMOA to operate on the building spatial design problem, it is now possible to compare them empirically. They will also form a baseline

to compare with the tailored SMS-EMOA variant. For these experiments a similar setup is used as in the previous chapter (Section 4.5). The 2221, 2223, 2225, 3331, 3333, and 3335 configurations are considered again. Here the first three digits again indicate the supercube size in width, depth, and height, while the last digit indicates the number of spaces. The most important change is that now the two objectives (minimal compliance, and minimal surface area) are optimised at the same time in a multi-objective setting. Besides this, here an evaluation budget of 2500 is used for five runs of each configuration and algorithm pair. The resulting Pareto front approximations of the five runs are then averaged and analysed visually with median attainment curves [49].

Further settings, as previously mentioned in the algorithm descriptions, are summarised in Table 5.1.

$(\mu+\lambda)$ NSGA-II	$(\mu+\lambda)$ SMS-EMOA	V_0	lb	ub	RP	MT
(20+20)	(50+1)	$4^3 \times N_{cells}$	3.0	19.8	0.5	$1/N_{dims}$

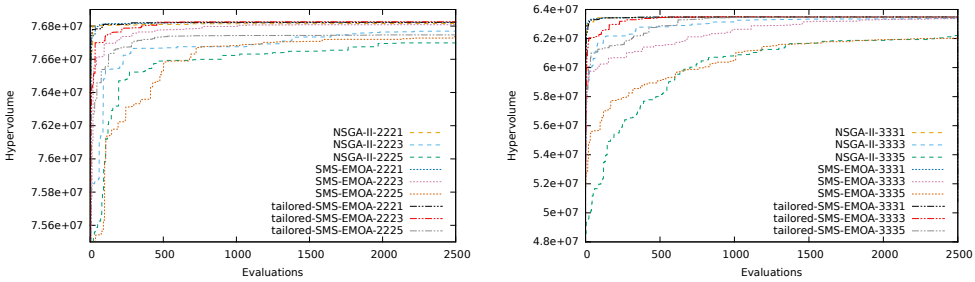
Table 5.1: Parameter settings for experiments with standard and tailored algorithms.

5.2.2 Results

Figure 5.2 shows the average convergence rate of the various problem configurations over five runs. Here, the hypervolume is computed over $\log(1 + \text{compliance})$, with a reference point $(35000, 2500)$ (compliance, surface area). Moreover, extreme outliers in either objective (points beyond the reference point), are left out of the analysis.

Most configurations and algorithms show a quick convergence to a relatively stable hypervolume value. In Figure 5.2 it can be observed that the tailored SMS-EMOA variant produces similar results to the other two approaches for problems with a single space. However, when three spaces are considered the tailored method improves over the other two by a decent margin, and for five spaces it is clearly better, both in terms of convergence speed and with regard to the final solution. In addition, for the most complex configuration (3335) in particular it is not clear whether NSGA-II and SMS-EMOA have converged,whereas the tailored SMS-EMOA does converge.

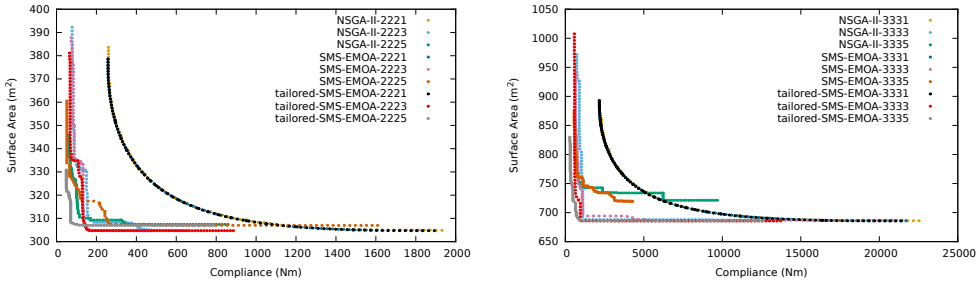
Next, the median attainment curves [49] produced by the algorithms will be analysed, and are shown in Figure 5.3. The attainment curve serves as a measure of the probability to attain (dominate) certain nondominated points by the approximation set. Both NSGA-II and SMS-EMOA produce similar curves as depicted in Figure 5.3.



(a) 222x configurations. **(b)** 333x configurations.

Figure 5.2: Average hypervolume growth over five runs, for one, three, and five spaces.

This indicates the considered optimisation process works, and is able to discover a Pareto front approximation. The same behaviour as seen for the hypervolume convergence can be observed from the attainment curves. Namely, the differences in performance become more pronounced with larger problem sizes. Clearly, problem specific operators are able to produce a better Pareto front approximation.



(a) 222x configurations. **(b)** 333x configurations.

Figure 5.3: Median attainment curves from five runs for one, three, and five spaces.

The standard deviations of the hypervolume at the final generation are relatively small for most problem configurations and do not change the numerical result. Only for the 3335 configuration (Table 5.2) large deviations occur for the generic methods, but even their highest hypervolume solutions do not outperform the smallest hypervolume found by the tailored method.

Since the 3335 configuration is the largest, it is also the best indicator for performance on even larger building spatial designs that may be considered in real world applications. As such, a Wilcoxon signed-rank test[1] [104] is conducted for this configu-

[1] A non-parametric test to compare the distributions of different samples

algorithm	min	max	mean	median	std dev
NSGA-II	60665475	63379795	6.22089e+07	62247270	1.05263e+06
SMS-EMOA	59642594	63387974	6.20227e+07	62313768	1.25616e+06
tailored SMS-EMOA	63492022	63495486	6.34941e+07	63494483	1145.48

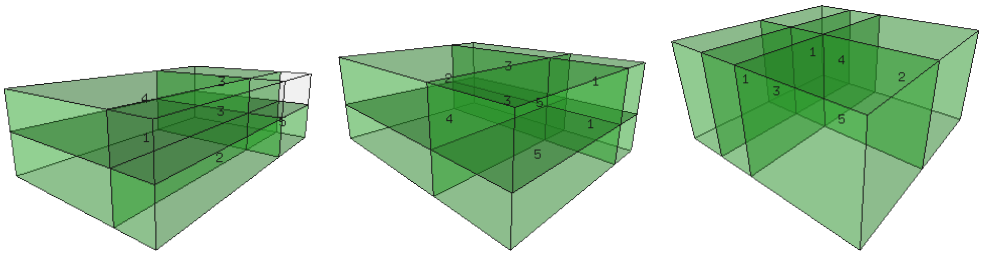
Table 5.2: Hypervolume statistics per algorithm for the 3335 problem instance.

ration to identify whether performance differences are statistically significant. Results of this test are summarised in Table 5.3. First NSGA-II and SMS-EMOA are compared to see whether one is preferable over the other for this application. This comparison results in $W = -1$, indicating there is no significant difference. Next, the same test is applied between NSGA-II, and the tailored SMS-EMOA. This results in $W = 15$, indicating the tailored method is better than NSGA-II with a statistical significance of 0.05. The exact same outcome is found when the same comparison is made between the standard SMS-EMOA, and the tailored SMS-EMOA. Evidently, the problem specific constraint handling operators make a valuable contribution to the performance, and are worth developing further.

run	NSGA-II	SMS-EMOA	tailored	sgn × rank NS	sgn × rank NT	sgn × rank ST
1	60665475	59642594	63492022	-5	5	5
2	61445894	62300678	63493947	4	4	4
3	62247270	62313768	63494483	2	3	3
4	63305880	62468595	63494488	-3	2	2
5	63379795	63387974	63495486	1	1	1
W				-1	15	15

Table 5.3: Wilcoxon signed-rank test results for the hypervolume attained for the 3335 problem instance.

Figure 5.4 shows the best found spatial designs in terms of each objective, as well as a compromise solution at the knee point of the median attainment curve. This is accompanied by some additional information on these spatial designs in Table 5.4. As can be expected, the optimal spatial design in terms of minimal surface area has a cuboid shape. The knee point solution is largely similar, but has a slightly lower structure, and as a result it is stretched in both width and depth to maintain the volume. Finally, the minimal compliance solution has an elongated structure. Moreover, it has a lower structure, which can be explained since it results in less strain on the structural elements, which in turn reduces the compliance.



(a) Minimal compliance. **(b)** Knee point. **(c)** Minimal surface area.

Figure 5.4: Best spatial designs found with the tailored SMS-EMOA for the 3335 configuration.

	Compliance (N m)	Surface area (m)	Soil surface (m)	Height (m)	Longest edge (m)	Shortest edge (m)
Minimal compliance	214.658	741.181	303.228	6.361	22.182	13.670
Knee point	451.351	690.077	252.545	6.845	17.658	14.302
Minimal surface area	9483.320	685.562	227.177	7.603	15.128	15.017

Table 5.4: Details about the best spatial designs found with the tailored SMS-EMOA for the 3335 configuration.

Table 5.5 shows the CPU time used by SMS-EMOA with problem specific operators. The other methods performed similarly because the compliance computations used by far the most CPU time. Each experiment used a single core of an i7-3770 CPU @ 3.40GHz processor, and had 16GiB DIMM DDR3 Synchronous 1600 MHz memory available.

problem configuration	2221	2223	2225	3331	3333	3335
CPU time (minutes)	42	342	888	42	620	1008

Table 5.5: Average runtime of SMS-EMOA with problem specific operators over five runs, rounded to the closest whole minute.

5.3 Unbiased Operators

Based on the success of the problem specific operators introduced in the previous section, they are developed further. First, bias free versions of the initialisation and mutation operators are introduced. Then, a new version of the tailored SMS-EMOA and its tunable parameters is summarised.

5.3.1 Initialisation

The initialisation approach presented in Section 5.1.1 contains some clear biases. Starting from single cell spaces results in a much lower probability of any space being stacked on top of another. While the application of a number of mutations results in a more varied initial population, the starting state of each initialisation is similar. The outcome of the mutations depends on this initial state, and as such does not lead to a truly random initialisation.

Here, an improved initialisation operator is presented that aims to produce an unbiased initial population. This provides a better distribution of the initial population over the search space. In addition, an unbiased random initialisation strategy is useful for a number of landscape analysis approaches.

This newly proposed initialisation method works from the principle that spaces of any shape should have equal probability to be included, as long as there is a large enough area to place them. As a starting point, for a supercube of a given size with a given number of spaces, all possible shapes and all possible positions of those shapes are considered. For a supercube of a size given by its width, depth and height (N_w, N_d, N_h) there are exactly $N_w \times N_d \times N_h$ possible shapes that do not violate any of the constraints, this is depicted in Figure 5.5. The number of possible shapes is limited by the number of spaces N_{spaces} that need to fit in the supercube. Therefore, the largest shape should leave at least $N_{spaces} - 1$ cells empty, such that the remaining spaces can use at least one cell each. It follows from these observations that the maximum size of a shape for a supercube described by the four parameters $N_w, N_d, N_h, N_{spaces}$ is limited to $N_w \times N_d \times N_h - (N_{spaces} - 1)$.

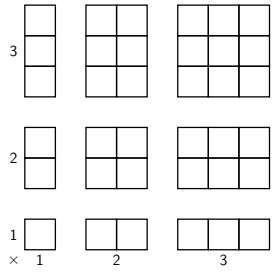


Figure 5.5: Possible feasible shapes for a 3×3 2D grid, this pattern extends to 3D and therefore to the supercube representation.

A space is then placed by uniformly at random selecting a pair of a possible shape s and a feasible position for that shape. The shape's possible positions are limited by

the number of cells belonging to the shape. A heightmap of size $N_w \times N_d$ is considered, and a shape has a width s_w, depth s_d, and height s_h. The heightmap $M_{i,j}$ is initialised as a matrix of zeros, where $i \in \{0, \ldots, N_w - 1\}$ and $j \in \{0, \ldots, N_d - 1\}$. The possible positions of a shape are then every pair i, j where $i + s_w \leq N_w$ and $j + s_d \leq N_d$ hold. A position is feasible if and only if $M_{i,j} + s_h \leq N_h$ and $\forall_{m,n} : M_{m,n} = M_{i,j}$, where $m \in \{i, \ldots, i + s_w\}$ and $n \in \{j, \ldots, j + s_d\}$. Once a shape and position combination is selected the corresponding bits $b^{\ell}_{m+1,n+1,k+1}$ are activated, where m and n are the same as before, $k \in \{M_{m,n}, \ldots, M_{m,n} + s_h\}$ and ℓ is the space under consideration. For every space that is placed, the heightmap is updated by $M_{m,n} + s_h$, again for the same values of m and n. Finally, once every space is placed, the continuous parameters are initialised uniformly at random within their bounds as in Section 5.1.2.

5.3.2 Mutation

Mutation in Section 5.1.2 used a probability to either apply a binary mutation or a continuous mutation. In case of a binary mutation a random space was selected and contracted or expanded in a random direction (Figure 5.1b). For this move the existence constraint (C3: all spaces must consist of at least one cell) and the supercube borders are respected. In other words, any move violating these constraints cannot be chosen. For a continuous mutation, polynomial mutation [32] was always applied to a single continuous variable, randomly selected from those variables that are relevant to at least one active cell. In the following this operator is improved and extended.

Firstly, when continuous mutation is selected, polynomial mutation is applied with some probability to each continuous variable. This has the following implications. There is now a chance that nothing is mutated, but continuous mutations are also no longer limited to a single variable. This means that both small and large changes in the continuous space are possible.

The binary mutation introduced in Section 5.1.2 contains a bias, making some moves more likely to occur than others. However, in the design of evolutionary algorithms, biases in the mutation operator should be avoided [103]. In this case, the bias is an artefact of the used procedure, where first a space is selected, and then a feasible (resulting in no constraint violations) move for that specific space is selected. As a result, when space A has one possible feasible move and space B has three possible feasible moves, the probability to select one of the moves for space B is lower than the probability of selecting the move for space A. This is resolved here by selecting a

combination of a space and a move instead, resulting in equal probabilities for every move.

Finally, the binary mutation operator is extended to allow mutations consisting of multiple steps. With multiple steps a mutation may result in larger changes to candidate solutions, which may also help to escape local optima. Additionally, multi-step mutations may be able to traverse infeasible regions in case of multiple disconnected feasible regions. In Section 5.1.2 inward moves from the bottom of a space were disallowed because they always result in an infeasible (constraint violating) space by introducing a vertical gap (Constraint C2) in the topological design. Here, such moves are allowed, along with most other constraint violating moves (only Constraint C3 and the supercube boundaries may never be violated). As a result, intermediate moves may lead to infeasible states.

Allowing infeasible intermediate states has a few implications. Figure 5.6 shows how it is possible to move from an initial feasible state to both feasible and infeasible intermediate states. Moreover, the same figure shows how an infeasible intermediate state could lead to both a feasible and an infeasible final state. Moves to infeasible final states are disallowed because only feasible final states are considered for evaluation.

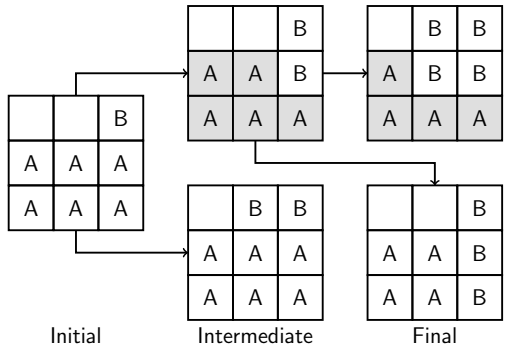


Figure 5.6: An initial state can result in both feasible and infeasible intermediate states, in turn an infeasible intermediate state may or may not result in a feasible final state. Infeasible spaces are shaded in grey, in both cases space A is not cuboid and thus violates Constraint C4.

Next, Figure 5.7 shows how spaces with an infeasible shape are mutated. Like with a feasible shape, only the outermost line of spaces in one direction is contracted or extended. Note that moving up from the initial state in the figure would be infeasible since the space would expand beyond the supercube boundary, it would not result in a feasible space of six cells as might be the intuition.

Mutations are then applied as follows: (1) apply a move to a uniformly at random

selected combination of a space and movement direction as long as all spaces keep at least one cell (Constraint C3) and supercube boundaries are respected, (2) if this is the final move, try space and direction combinations, in a uniformly at random selected order, until a move is found that satisfies all constraints. If no feasible state is found after exhausting all possible moves the offspring is returned without any mutation (all previous moves are discarded).

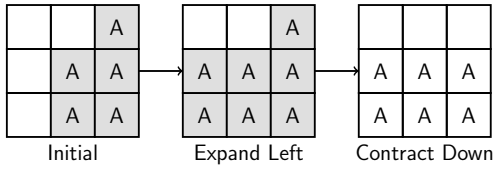


Figure 5.7: Expansions and contractions are applied exclusively to the outermost line of cells, even for a space in an infeasible state. The initial state here is infeasible (Constraint C4) and only shown for illustrative purposes. Infeasible spaces are shaded in grey.

5.3.3 Tailored SMS-EMOA

The improved operators are integrated into a tailored version of SMS-EMOA, outlined in Algorithm 3. A number of options are available that will be used to tune the algorithm later in this chapter (Section 5.5). Firstly, the population size is as usual controlled by μ. One of two initialisation techniques is selected with IT: (1) initialisation as in Section 5.1.2; or (2) unbiased initialisation as proposed in Section 5.3.1. Next, IM controls how many mutations are applied with initialisation technique (1) to increase initial diversity. Parameter MT controls the type of mutation that is applied. It represents the probability to mutate the binary variables with the unbiased mutation operator from Section 5.3.2, and the inverse probability to apply polynomial mutation to the continuous variables. With ST a technique to control the step size in the binary mutation operator is selected: (1) a fixed number of moves FS; or (2) pooling, where uniformly at random either a local move of one step or an explorative move of three steps is chosen. In the future, more options, such as step size adaptation, may be explored. When a fixed step size is used FS controls the number of steps, otherwise it has no effect on the algorithm. MC denotes the probability to apply polynomial mutation (if chosen instead of binary mutation) for each continuous variable. The tailored SMS-EMOA (using the improved problem specific operators) is compared to the standard SMS-EMOA (described in Section 5.1.1) after tuning the parameters of both versions in Section 5.5.

Algorithm 3 Outline of the tailored SMS-EMOA and tunable parameters

Require: IT, μ, IM, MT, ST, FS, MC
if $IT = 1$ **then**
    Generate population X of μ parents as in Section 5.1.2
    for $i \in \{1, \ldots, IM\}$ **do**
        Mutate all individuals in X as in Section 5.1.2
    end for
else ▷ $IT = 2$
    Generate population X of μ parents as in Section 5.3.1
end if
while Stop condition not met **do**
    $X' \leftarrow$ A uniform random individual from X
    if $U(0,1) \leq MT$ **then**
        if $ST = 1$ **then** ▷ (Fixed step size)
            n_steps $\leftarrow FS$
        else ▷ $ST = 2$ (Pooling)
            if $U(0,1) \leq 0.5$ **then**
                n_steps $\leftarrow 1$ ▷ Local move
            else
                n_steps $\leftarrow 3$ ▷ Explorative move
            end if
        end if
        Mutate binary variables in X' with n_steps as in Section 5.3.2
    else
        Apply polynomial mutation to each continuous variable in X' with probability MC
    end if
    $X \leftarrow$ Select μ individuals from $X \cup X'$
end while

5.4 Landscape Analysis

As mentioned in the introduction to this chapter, it is possible to employ landscape analysis both to learn about the objective landscape, and to evaluate variation operators (e.g. mutation, recombination). With both an unbiased initialisation operator, and an unbiased mutation operator having been defined in the previous section, everything is now in place to do landscape analysis. First, Section 5.4.1 describes the considered setup, and then Section 5.4.2 discusses the results.

5.4.1 Setup

The landscape of the multi-objective problem is analysed by randomly sampling solutions generated by the newly introduced unbiased initialisation operator (Section 5.3.1). This provides insight in the distribution of solutions over the objective landscape. In addition to random sampling, mutations are applied to investigate the landscape around the randomly sampled points. Through these mutations, the appearance of the local landscape can be investigated. Moreover, it gives insight into the behaviour of the unbiased mutation operator introduced in Section 5.3.2.

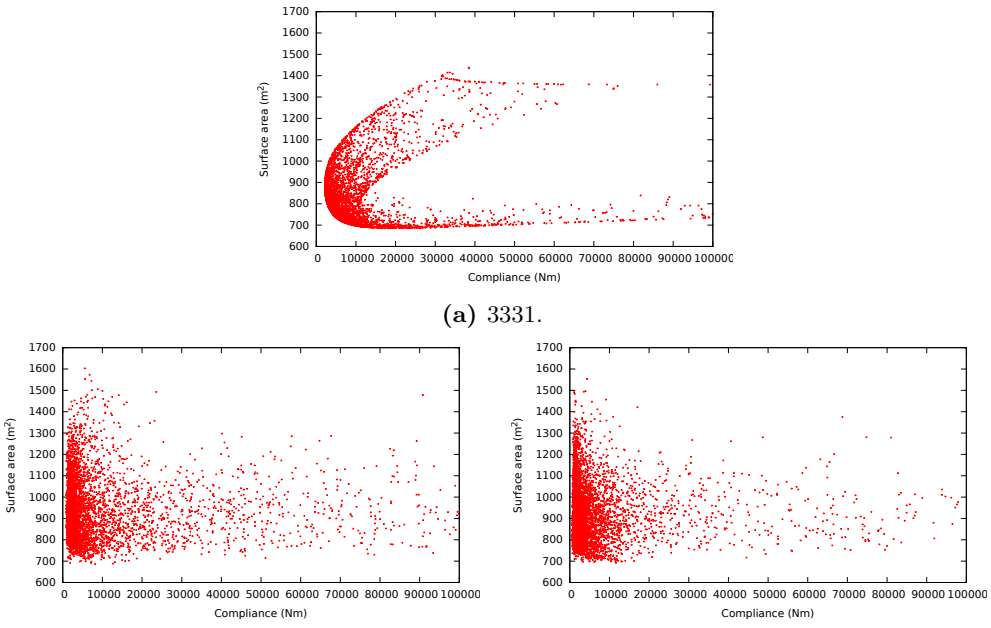
As before, problem instances are defined by four numbers corresponding to the supercube parameters N_w, N_d, N_h and N_{spaces}. Experiments are conducted for three different instances: 3331, 3333 and 3335. For each instance fixed step sizes from 1 to 5 are considered. In the experiment a single parent individual is generated with the unbiased initialisation operator, and for this parent a single offspring individual is generated using the unbiased mutation operator. This process is repeated 1000 times for each instance and step size combination, totalling 5000 executions per instance.

In these experiments only binary mutations are considered. The rationale behind this is that each possible distribution of spaces in the supercube corresponds to a different – possibly overlapping – subspace in the objective landscape. When only the continuous variables are changed the parent and offspring would be in the same subspace. Since the number of possible space distributions in the supercube is large, binary mutations will lead to better coverage of the objective landscape. Future work may explore the subspaces more extensively, and for instance investigate the degree of their overlap.

Continuous variables are initialised in $]3, 19.8]$ and a volume $V_0 = 4^3 \times N_{cells}$ is maintained where $N_{cells} = N_w \times N_d \times N_h$. Recall that every individual is scaled to this volume whenever it deviates (e.g. after mutation).

5.4.2 Results

In Figure 5.8 all parent individuals are plotted for the different problem instances (5000 each). For the 3331 instance (Figure 5.8a) a clear Pareto front approximation is visible, which is very similar to the one found in Section 5.2.2. Additionally, many other parts of the landscape also show smooth curves. This suggests that these areas are well covered by random sampling. The largest concentration of points is seen reasonably close to the Pareto front. This makes it probable that random sampling results in a decent solution for this instance.



(a) 3331.

(b) 3333.

(c) 3335.

Figure 5.8: Landscapes per problem instance, each with 5000 randomly initialised samples, limited to points with compliance below 100 000 N m.

Both the 3333 and 3335 problem instances (Figures 5.8b and 5.8c) show a different distribution over the objective landscape, indicating that these are, as expected, more complicated. On the other hand, both of these also show a large concentration of points in an area of decent solutions. It appears that finding fairly good solutions is feasible with random sampling, but finding exact optima is still challenging.

Next, the absolute change in the compliance objective values is shown for each instance and step size combination in Table 5.6. First of all, it may be observed that the variance in the change to objective values is large for all considered cases. For instances 3331 and 3333 the change in compliance from small to large step sizes shows a very slight trend towards greater change with larger step sizes. For the 3335 instance the absolute change for different step sizes shows a more parabolic behaviour, where both small and large step sizes have less impact on the compliance than medium step sizes. This is likely caused by larger step sizes reaching infeasible intermediate states in the mutation more frequently, and then being unable to find a feasible final move and reverting back to the original.

steps	min	max	mean	median	std dev	zeros
			instance 3331			
1	3.10	333667	9359.36	2833.29	24848.95	0
2	0.00	401687	9228.16	2048.79	30041.31	209
3	0.00	3621254	16002.67	3758.13	118798.72	10
4	0.00	36910000	50354.11	3638.98	1166743.33	79
5	0.00	403216	11749.73	3516.93	32740.83	18
			instance 3333			
1	0.01	3.20e+16	3.22e+13	2672.43	1.01e+15	0
2	0.00	1.29e+15	1.31e+12	2283.77	4.07e+13	44
3	0.00	9.63e+13	1.61e+11	4527.37	3.58e+12	8
4	0.00	7.89e+18	8.13e+15	4319.85	2.49e+17	36
5	0.00	4.28e+18	5.00e+15	4823.08	1.37e+17	35
			instance 3335			
1	1.02	3.07e+17	4.75e+14	1235.32	1.11e+16	0
2	0.00	5.45e+33	5.45e+30	1331.23	1.72e+32	19
3	0.00	8.68e+22	8.68e+19	1999.31	2.74e+21	12
4	0.00	7.78e+12	1.74e+10	1709.64	3.26e+11	16
5	0.00	3.96e+17	3.96e+14	1153.29	1.25e+16	28

Table 5.6: Statistics for absolute change in compliance per instance.

Absolute change to the surface area in Table 5.7 shows a somewhat clearer trend across the board. For instance, the trend of larger changes for larger step sizes seems to be more pronounced when, for instance, looking at the mean and median.

Figure 5.9 visualises how objective values are distributed in both objectives for the 3333 instance. The other instances produced largely similar visuals. Although the differences in absolute change do not appear to be very large between the various step sizes, there does appear to be a trend towards larger changes for larger step sizes.

For both objectives zero changes also occur (see the rightmost column in Tables 5.6 and 5.7). There are multiple explanations for this. Firstly, when multiple steps are taken a second move may be the inverse of the first and as such return to the original solution. A second explanation is found in the design of the mutation operator, where the algorithm reverts to the original individual if no feasible final moves are possible. Finally, there are moves where only the interior of the building design changes, in other words, where only walls are moved. In those cases the compliance changes, but the surface area does not.

steps	min	max	mean	median	std dev	zeros
			instance 3331			
1	0.22	632.13	77.51	51.28	82.98	0
2	0.00	534.44	74.70	36.49	96.36	209
3	0.00	657.30	95.10	61.32	99.20	10
4	0.00	667.93	100.61	64.83	115.66	79
5	0.00	662.83	104.26	67.63	107.07	18
			instance 3333			
1	0.00	450.32	64.02	40.59	70.70	20
2	0.00	641.03	70.54	42.61	84.05	50
3	0.00	478.77	84.81	57.22	86.48	16
4	0.00	540.57	93.02	62.35	96.33	38
5	0.00	642.32	96.46	63.63	104.26	36
			instance 3335			
1	0.00	482.75	50.78	28.10	67.51	15
2	0.00	573.65	57.96	28.76	77.71	27
3	0.00	532.80	68.83	40.19	84.21	17
4	0.00	520.17	71.95	40.28	92.19	15
5	0.00	470.09	63.18	27.68	83.76	28

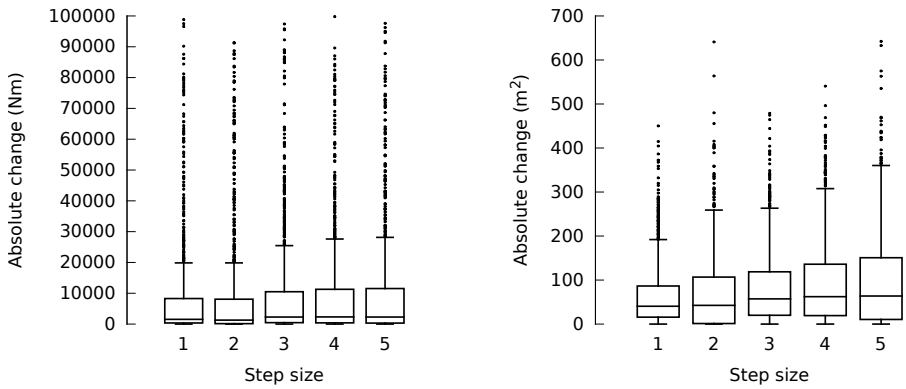
Table 5.7: Statistics for absolute change in surface area per instance.

5.5 Optimisation and Parameter Tuning

The landscape analysis in the previous section showed that the unbiased initialisation operator is able to generate a diverse set of solutions, and the unbiased mutation operator produces larger changes with larger step sizes. Having confirmed these desirable properties of the operators, they will now be evaluated in an optimisation setting. This section first briefly reviews the experimental setup from Section 5.2.1, which will be used here as well. Next, the considered setup for algorithm tuning is introduced. Finally, the results of these experiments are discussed.

5.5.1 Optimisation Setup

SMS-EMOA as used here is the standard SMS-EMOA from [39], with the only changes being the volume rescaling and the evaluation as described in the following paragraphs (see also Section 5.1.1). A tailored version of SMS-EMOA with problem specific operators for building spatial design is used, this will be referred to as tailored SMS-EMOA. Changes are as follows: (1) problem specific initialisation is used, where tuning will select between the biased version from Section 5.1.2, and the unbiased one from Sec-



(a) Compliance. **(b)** Surface area.

Figure 5.9: Absolute change to the objective values with 1000 samples for a single mutation of 1 through 5 steps for the 3333 problem instance.

tion 5.3.1, and (2) offspring is generated with the unbiased problem specific mutation operator as described in Section 5.3.2. All other settings, except for the evaluation budget and tunable parameters detailed later, are exactly as in Section 5.2.1. Note that no recombination operator is used with the tailored SMS-EMOA. The reason behind this is that the standard recombination operators result in many constraint violations because of the many infeasible solutions in the supercube as reported in Section 3.2.4. Due to the complexity of developing such an operator, and the promising results when only using mutation as shown in Section 5.2.2, development of a problem specific recombination operator is deferred to future work.

For both the standard and tailored versions of SMS-EMOA the considered optimisation problem is as follows. Objective functions are (1) the compliance in N m (newton metre), and (2) the surface area in m^2 (square metre), both to be minimised. Binary and continuous variables are considered as described in Section 3.2. The binary variables are subject to four topological constraints as described mathematically in Section 3.2.2. To comply to the volume constraint (Section 3.2.3), the continuous variables are rescaled for any new individuals that are produced. Both algorithms consider the reference point $(1.1e9, 1.1e9)$.

Function evaluations for both of these algorithms proceed as follows. First constraints are checked. For constraint violations a penalty of $999\,999\,998+CV$ is returned as the objective value for both objectives, for a number of constraint violations CV. As in previous experiments, the fourth constraint is considered in two parts, result-

ing in $CV \in \{1, \ldots, 5\}$. If the objective value is 999 999 999 or higher (i.e. at least one constraint is violated) no evaluation is counted, but the individual may still be selected, provided it passes the selection criteria. In other words, any individual violating a constraint or showing extremely poor performance is considered infeasible. Note that although constraint violations do not occur with the tailored SMS-EMOA, there may be individuals with extremely poor performance. In all other cases the remaining evaluation budget is decreased by one.

5.5.2 Parameter Tuning Setup

Acknowledgement

The author gratefully acknowledges the helpful advice of Leslie Pérez Cáceres and Manuel López-Ibáñez on setting up and using the irace package.

To tune the parameters of the standard and tailored SMS-EMOA for building spatial design, two approaches are compared: The irace package [71] and the Mixed-Integer Evolution Strategy (MIES) [69]. The objective is the maximisation of the hypervolume indicator (HVI, sometimes also referred to as just hypervolume), with reference point $(100000, 1500)$. Note that here the hypervolume is taken over all evaluated individuals, not just over the individuals in the final population.

Objective function values are normalised to a $[0, 1]$ range for both objectives. For the compliance the used range is $[0, \ldots, 100000]$ and for the surface area a range of $[0, \ldots, 1500]$ is considered. Data points exceeding either of these ranges are excluded from the analysis. These ranges are based on investigation of the result data from Section 5.2.2. This ensures that only extreme outliers are discarded and the data can still be analysed visually.

For the standard SMS-EMOA three tunable parameters are considered as shown in Table 5.8. The range for the mutation probability is restricted for the standard SMS-EMOA because large mutations are likely to result in infeasible individuals. Too small population sizes are also avoided since an initial population without feasible individuals has great difficulty navigating to a set of feasible individuals even with the graded penalties for constraint violations. As a result, the algorithm would use extreme amounts of time without being able to evaluate feasible individuals.

For the tailored SMS-EMOA the parameters from Algorithm 3 (Section 5.3.3) are tuned within ranges as shown in Table 5.9 (parameters without type are categorical).

From the results of the landscape analysis in Section 5.4.2 it can be concluded

parameter	symbol	range	type
Population size	μ	$\{10, \ldots, 100\}$	$\mathbb{Z}$
Recombination probability	RP	$[0.0, 1.0]$	$\mathbb{R}$
Mutation probability	MP	$[0.005, 0.100]$	$\mathbb{R}$

Table 5.8: Configurable parameters for the standard SMS-EMOA.

parameter	symbol	range	type
Population size	μ	$\{10, \ldots, 100\}$	$\mathbb{Z}$
Mutation type probability	MT	$[0.0, 1.0]$	$\mathbb{R}$
Step size technique	ST	$\{1, 2\}$	
Fixed number of steps	FS	$\{1, \ldots, 5\}$	$\mathbb{Z}$
Continuous mutation probability	MC	$[0.0, 1.0]$	$\mathbb{R}$
Initialisation technique	IT	$\{1, 2\}$	
Initialisation mutations	IM	$\{1, \ldots, 100\}$	$\mathbb{Z}$

Table 5.9: Configurable parameters for the tailored SMS-EMOA.

that the 3331 (supercube of size $3 \times 3 \times 3$ with a single space) problem instance is rather simple. A more challenging tuning problem can be found in the 3333 instance. Moreover, the evaluation time for this problem is less prohibitive than for the 3335 instance, as reported in Section 5.2.2. As such, the 3333 instance is used in the experiments. The tuning budget consists of two parts, namely (1) the number of algorithm executions, and (2) the number of function evaluations per execution. Firstly, the number of algorithm executions is set to the minimum requirement enforced by the irace package, which was 180 algorithm executions for the given settings and tunable parameters. Secondly, the results in Section 5.2.2 show convergence to a near stable state in about 300 function evaluations. With this in mind an evaluation budget of 300 is considered for tuning.

Both irace [71] (version 2.1.1662) and MIES [69] are used primarily with default settings as described in their respective papers, any deviations from those settings will be explicitly stated. As a result performance of both approaches is likely not optimal, but the configuration of parameter tuners could be the subject of a whole other study, and is considered out of scope here.

MIES is used with multiple step size mode for both real valued and integer parameters, and single step size mode for categorical parameters. In [69] it is suggested to keep step size bounds for categorical variables in $[\frac{1}{n_d}, \frac{1}{2}]$ when using single step size mode for a number of categorical variables n_d. However, for $n_d = 2$ this means the step size is constant at $\frac{1}{2}$. Since two categorical variables are considered for this

experiment, and a fixed step size is unable to adapt anything, categorical step sizes are instead bounded in $[\frac{1}{3}, \frac{1}{2}]$.

In [42] a (μ, κ, λ) strategy is shown to converge faster than a (μ, λ) strategy. Here κ is the maximum number of generations that an individual can stay in the population. Due to the small number of available evaluations, faster convergence is desirable here. Instead of the $(3, 5, 10)$ strategy from [42], here a $(3, 3, 10)$ strategy is used. The intuition is that due to the small number of generations, keeping individuals in the population for a too large fraction of the generations would otherwise not differ from a plus-strategy (see Section 2.3 for an explanation of a plus-strategy).

5.5.3 Tuned Configurations

Table 5.10 shows the configurations for standard SMS-EMOA. Here an untuned configuration (US) is included for comparison purposes. The parameters of this configuration are almost the same as those used in Section 5.2.1. Only *MP* is now 0.0111, instead of exactly $\frac{1}{N_{dims}}$.

ID	μ	*RP*	*MP*	mean HVI
untuned configuration (US)				
US	50	0.5000	0.0111	0.5348
irace configurations (IxS)				
I1S	31	0.8984	0.0385	0.5380
I2S	41	0.9650	0.0520	0.5385
I3S	73	0.8677	0.0427	0.5381
Mean	48	0.8942	0.0514	N/A
Std	19	0.1133	0.0105	N/A
MIES configurations (MxS)				
M1S	15	0.9679	0.0323	0.5386
M2S	40	0.5567	0.0891	0.5365
M3S	5	0.9709	0.0351	0.5364
Mean	35	0.8088	0.0545	N/A
Std	35	0.1834	0.0262	N/A

Table 5.10: Parameter configurations for the standard SMS-EMOA: untuned and from three repetitions of irace and MIES.

Both irace and MIES used around two weeks of computation time on a single CPU core per repetition to tune the parameters. The number of output configurations for irace can vary, but in this case there were nine in total after three repetitions. Since irace clearly indicates which configuration it considers to be the best, this configuration

is shown in Table 5.10 for each repetition. For MIES there are always three output individuals, because the population size $\mu = 3$. As a result, MIES also has a total of nine output configurations after three repetitions. For each repetition of MIES, the configuration with the largest hypervolume coverage is considered to be the best and shown in Table 5.10. The table also shows the mean and standard deviation for each parameter over all nine output configurations for both irace and MIES.

Compared to the untuned configuration (US), both tuning approaches found higher recombination (RP) and mutation (MP) probabilities for the standard SMS-EMOA. A possible explanation is that by producing larger and more frequent variations, the algorithm is able to explore more effectively, and in turn finds better solutions. While this is also likely to lead to more constraint violations, such solutions are not counted for the evaluation budget here, and thus in fact result in more solutions being considered. In terms of population size not much can be concluded, except that many different population sizes seem to allow for similar performance. Part of the reason may be that all considered solutions are counted towards the final Pareto front approximation, and therefore the population size has less influence. However, the population size also plays a role in the diversity of possible offspring individuals that can be reached, which is not explained by this argument. Combined with the larger mutation and recombination rates however, it may simply not play a very large role in reaching diverse solutions.

Parameter configurations for the tailored SMS-EMOA are reported in Table 5.11. Here too, the settings for the untuned configuration (UT) are mostly as in Section 5.2.1. Only MC is adjusted because continuous mutation changed in the unbiased mutation operator (Section 5.3.2). For this application, irace found eleven solutions in total, of which again only the best reported per repetition is shown in the table. For MIES once more nine configurations were found, and for each repetition the one with the largest hypervolume coverage is shown. As before, the mean and standard deviation for each parameter are taken over the full set of output configurations, rather than only the best per repetition.

With the tailored version using a smaller population size (μ), more frequent use of the binary mutation operator (MT) and an increased probability to apply continuous mutations (MC) seems to improve performance over the untuned configuration. While for the standard SMS-EMOA increased variation probabilities could be explained by being advantageous in the sense that constraint violations are not counted as evaluation anyway, this cannot be the reason here. For the tailored SMS-EMOA it appears that the increase in exploration caused by these settings is more advantageous than

ID	μ	MT	ST	FS	MC	IT	IM	mean HVI
Untuned configuration (UT)								
UT	50	0.2500	1	1	0.3333	1	20	0.5384
irace configurations (IxT)								
I1T	32	0.6890	1	4	0.6686	1	3	0.5390
I2T	21	0.2794	2	N/A	0.6894	2	N/A	0.5388
I3T	26	0.3960	2	N/A	0.3231	1	64	0.5393
Mean	28	0.5618	1.5	3.8	0.4752	1.4	34	N/A
Std	16	0.1463	0.5	0.4	0.2338	0.5	30	N/A
MIES configurations (MxT)								
M1T	12	0.6212	1	2	0.7970	1	69	0.5374
M2T	6	0.4993	2	N/A	0.4381	1	60	0.5374
M3T	5	0.1176	2	N/A	0.5118	1	43	0.5365
Mean	14	0.4413	1.3	2.5	0.6780	1.4	53	N/A
Std	9	0.1791	0.5	0.5	0.1921	0.5	11	N/A

Table 5.11: Parameter configurations for the tailored SMS-EMOA: untuned and from three repetitions of irace and MIES.

staying close to known solutions. It is also notable that there is a slight preference for the biased initialisation technique ($IT = 1$). A possible explanation is that the use of a large number of initialisation mutations (IM) mitigates the bias. What may also play a role is that although the unbiased initialisation technique ($IT = 2$) is more random, such solutions are not necessarily better.

In order to compare the optimised configurations and their corresponding algorithms each of them is evaluated on the considered optimisation problem (Section 5.5.1). Each configuration is compared at two stages in the optimisation process. First with a budget of 300 evaluations using fifteen repetitions, and then with a budget of 1000 evaluations using thirteen repetitions. In both cases all other settings are the same as before. The mean hypervolume (HVI) values after 1000 evaluations are also included in Tables 5.10 and 5.11.

Median attainment curves [49] of the resulting data are used to compare the different configurations, as well as the two tuning methods. In addition to those configurations, the Pareto front approximation found by taking 5000 random samples (RS) for the landscape analysis from Section 5.4.2 is also included. This makes it possible to compare random sampling with both SMS-EMOA variants. Figure 5.10 shows the situation after 300 evaluations.

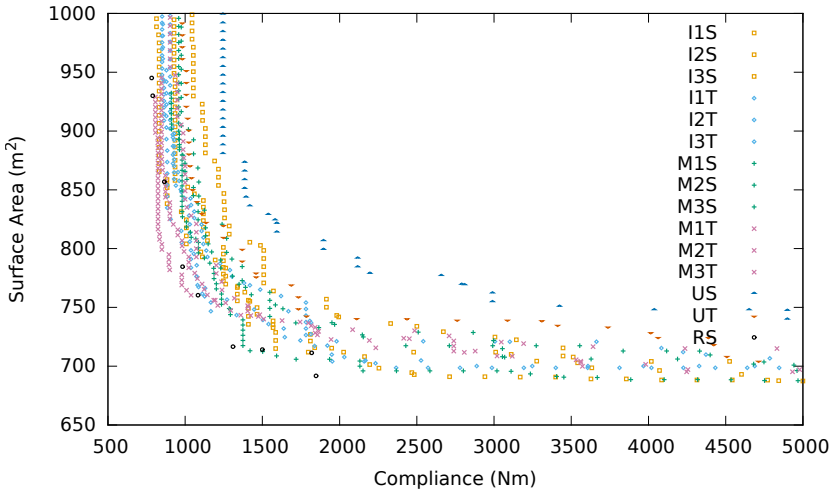


Figure 5.10: Median attainment curves after 300 evaluations, over fifteen repetitions for configurations tuned by irace and MIES for standard and tailored versions of SMS-EMOA, untuned configurations, and a Pareto front approximation from 5000 random samples.

A first observation is that even with only 300 evaluations, a number of configurations are already competitive with the results of 5000 random samples (RS). Evidently, optimisation is effective here. Furthermore, the untuned configuration for the standard SMS-EMOA (US) is clearly not competitive with anything else, while the untuned tailored SMS-EMOA (UT) performs better, it is among the worst of the other configurations. This shows that tuning has been worthwhile in improving performance. Notably, all configurations tuned for the standard SMS-EMOA (xxS) outperform the untuned configuration in every part of the Pareto front approximation (PFA). For the tailored SMS-EMOA (xxT), this is true for almost all parts of the PFA. In terms of performance differences between tuned configurations there does not seem to be a clear winner as far as optimal performance goes. However, there does seem to be less variance between configurations found for the tailored SMS-EMOA, indicating that it may be a more robust algorithm. With regard to the tuning approaches, it appears that MIES is prone to finding high quality solutions in one objective, but then slightly less so in the other objective. On the other hand irace seems to usually find a more balanced PFA, but as a result does not find the highest quality solutions for every part of the Pareto front.

Next, Figure 5.11 shows the situation for the same configurations after 1000 evaluations. It is obvious that everything progressed to find better solutions, and the use

of more evaluations was a worthwhile investment. For the surface area part of the PFA, almost all configurations reach a very similar approximation. More variation is seen in the knee point region and the compliance objective, where sometimes MIES tuned tailored SMS-EMOA configurations are better, and other times the irace tuned

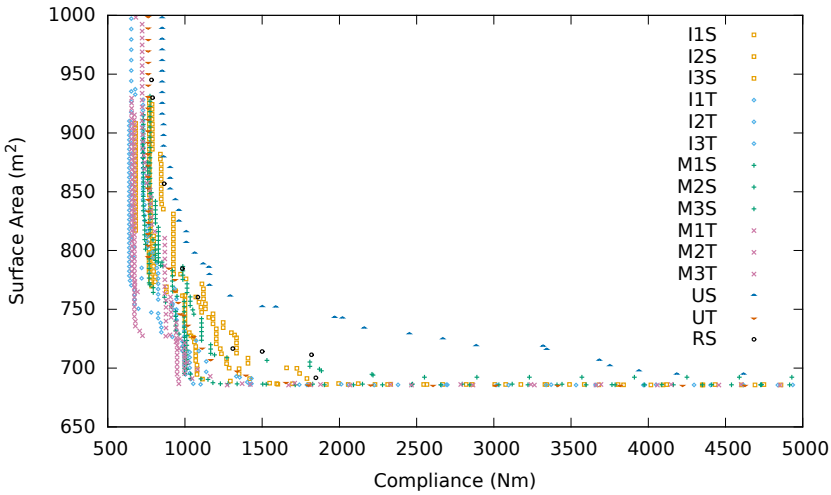


Figure 5.11: Median attainment curves after 1000 evaluations, over thirteen repetitions for configurations tuned by irace and MIES for standard and tailored versions of SMS-EMOA, untuned configurations, and a Pareto front approximation from 5000 random samples.

versions. Except for the occasional outlier, the standard SMS-EMOA configurations are not competitive in these areas of the PFA. Besides the untuned standard SMS-EMOA, all configurations now outperform, or are competitive with the 5000 random samples.

In summary it is evident that even with a relatively limited tuning budget both tuning approaches (irace and MIES) are able to improve over untuned configurations, but there is no clear preference for one or the other. Although the standard SMS-EMOA is not much worse in its final results, it appears to be less robust (even with tuned configurations) in its performance than the tailored version. Moreover, based on the results in Figure 5.11 it also appears to be the case that the standard SMS-EMOA is slower to converge to the true Pareto front, and might even get stuck before reaching it. Even so, these are promising results since based on Section 5.2.2 it can be expected that differences between the standard and tailored methods will only be larger for more difficult and realistic problem instances. Clearly, the development of the tailored SMS-EMOA has been beneficial.

5.6 Conclusion

5.6.1 Summary

Due to the limited success of penalty methods in navigating the constrained landscape of the building spatial design problem, this chapter continued the investigation of how to navigate these landscapes effectively (RQ2). To this end, problem specific operators have been developed. Specifically, initialisation and mutation operators. In addition, with a basic understanding of the objectives (thermal and structural performance) having been established, this chapter addressed them in a multi-objective fashion.

Development of the problem specific operators led to basic initialisation and mutation techniques that can only reach feasible solutions (non constraint violating). In order to assess the performance of an algorithm using these operators a comparison has been made to an algorithm using the penalty techniques previously developed in Chapter 4. The results showed that while both approaches work for the most basic problem instances, the penalty based algorithms quickly fall behind when larger problems are considered. Clearly, the problem specific operators are beneficial in navigating the search space.

Although these basic problem specific operators already showed promise, they contained clear biases. Moreover, while it is not clear whether any disconnected feasible regions exist, the basic mutation operator was unable to reach such regions due to fixed size moves. To resolve this, as well as the bias, an unbiased problem specific mutation operator has been developed. Likewise, an unbiased initialisation operator has also been introduced.

As a first test for these new unbiased operators, they have been subjected to landscape analysis. The initialisation operator showed differences in search space complexity between a very basic problem instance, and two more complex variants. In addition, the mutation operator showed a correlation between its step size and the degree of change to the produced solutions. This is a desirable property since it means that this parameter can control the balance between exploration and exploitation of this operator.

Following the landscape analysis, parameter tuning has been applied in order to assess whether the unbiased operators are indeed preferable over the basic ones. In addition, this served to compare the parameter tuning package irace [71] to the mixed-integer evolution strategy (MIES) [69]. The results showed that the unbiased mutation operator indeed reaches better performance than the basic one. For the initialisation operators there was no great preference for either, but since the unbiased operator

initialises a diverse population with less computational effort it is recommended over the basic version. In terms of tuning no clear advantage has been shown for either the irace package or MIES. Evidently, although less commonly used, MIES is an adequate tool to tune the parameters of an algorithm.

5.6.2 Future Work

The results with the problem specific operators showed that a mutation only evolutionary algorithm is able to navigate the constrained landscape. However, in order to scale up to larger problem sizes (buildings with more spaces) further research in this direction is needed. One aspect that could help the search process is the inclusion of a recombination operator. What makes this a challenging research direction is that it is not obvious how to recombine the discrete part of the supercube representation in such a way that designs remain both feasible, and result in a reasonable combination of two parent designs.

An aspect of the problem specific mutation operator that could be improved is the inclusion of step size adaptation. This is a standard component of Evolution Strategies [81, 88], that allows the mutation strength to be adapted during the execution of the algorithm. Through this process, the algorithm can start with a more explorative approach, and become more exploitative as it converges. The main challenge here is figuring out what an effective adaption mechanism is for the newly introduced operator.

Landscape analysis showed how the considered problem quickly becomes more difficult when problem instances are considered beyond the most basic versions. What is not yet clear is how the interplay between the continuous and discrete components of the supercube representation influences the complexity. To elucidate this, it should be investigated to which degree different discrete subspaces overlap in objective space, as a result from continuous variations within these distinct discrete subspaces.

Since the considered problem of building spatial design requires costly evaluations, algorithms that require fewer evaluations would be valuable. One way to reduce the number of evaluations that are needed is the use of so-called surrogate models (also called metamodels). Such models aim to provide a cheaper alternative for the actual objective function, at the cost of accuracy. How to integrate this into the building spatial design problem, and the used mixed-integer representation is an open question.

Like in the optimisation process, surrogate models can also be applied in parameter tuning. As seen in this chapter, tuning algorithms for relatively small building spa-

tial designs is already expensive. Future work can investigate how parameter tuning techniques with surrogate models (e.g. SPOT [9] and SMAC [55]) can be used to cope with tuning algorithms that optimise larger building spatial designs.

Chapter 6

Local Search with Set Gradients

In the previous chapter problem specific constraint handling operators were introduced to efficiently navigate the search space, and as such RQ2 was answered. Furthermore, parameter tuning was employed to maximise the performance of the algorithm. However, finding exact optima remains a challenge. To this end this chapter follows up on RQ3, and investigates whether local search can improve the solutions found during the global search performed by evolutionary optimisation.

The ultimate goal in optimisation is finding the global optimum. Convergence to the global optimum is quick for convex and continuous optimisation problems when exact methods like gradient search are used. In complex functions, however, such exact methods may get stuck in local optima. Exploring multiple local optima requires different methods, such as evolutionary algorithms for example. However, heuristic methods like evolutionary algorithms may be slower to lock in on an exact (local) optimum. As such, a combination of these methods could provide advantages over either of the individual methods. Hybrids of such heuristic and exact methods are called memetic algorithms [77]. Here, a combination of evolutionary search and gradient search is proposed.

Gradient search, like other exact optimisation methods, has traditionally been used for the single-objective case. In fact, until recently gradients were only defined for single points. In the generalisation of gradient methods for multi-objective optimisation the challenge arises that typically a set of points is considered together, the so-called

Pareto front. One way to measure the quality of a Pareto front approximation (PFA) is the hypervolume indicator (HVI). In [40] the HVI gradient is described, which allows a set of points to be moved towards the Pareto front, and to be distributed well across the Pareto front. The HVI gradient has been tested on benchmark functions [41], and improvements to its computation [40], as well as to the navigation of dominated points [97, 98] have been proposed, but it was never tested on real world problems.

Other gradient approaches for multi-objective optimisation exist as well. The key difference is the use of set gradients (in case of the HVI gradient), as opposed to single point gradients [46, 85], and gradients that are used for the computation of bounds on subspaces [34]. Other ideas to use gradients in memetic search have been proposed in the literature, such as continuation methods that start from a pre-computed Pareto optimal point and locally extend Pareto fronts by steps along tangent planes [74, 86]. Moreover, directed search has been proposed, which steers points in a desired direction, either across or towards the Pareto front. Such methods also use gradients in order to construct these directions in the decision space [87]. Further, recent developments on derivative free exact local methods are summarised in [29]. The HVI gradient is favoured here since it updates the Pareto front approximation as a whole, and local optimality has been verified [41].

In this work the HVI gradient is applied to the real world problem of building spatial design for the first time. Furthermore, this chapter employs simulations to measure energy efficiency, rather than the previous substitute of outside surface area. The combination of an evolutionary multi-objective optimisation algorithm (EMOA) and the HVI gradient results in a memetic multi-objective optimisation (MEMO) algorithm. Specifically, here the $\mathcal{S}$-metric (also known as HVI) selection EMOA (SMS-EMOA [39]), with specialised operators (Section 5.3) for the building spatial design optimisation problem is used. For the HVI gradient component the hypervolume indicator gradient ascent multi-objective optimisation (HIGA-MO) approach from [97] is employed.

To summarise, key points of this chapter are as follows. The HVI gradient is used in a memetic setting for the first time. Since local optimality of the HVI gradient has been verified [41] it is an excellent candidate to explore the potential of local search for the considered problem. In addition, it may provide guarantees with regard to local convergence in the continuous subspace. Furthermore, the HVI gradient method is also subjected to constraints and a mixed-integer search space for the first time. These new challenges should provide insight into the effectiveness of the HVI gradient in more complex search spaces. Finally, more accurate measures of energy and structural

performance are considered in this chapter.

The remainder of this chapter is structured as follows. Section 6.1 reviews the basics of multi-objective optimisation, and briefly describes the principles of the hypervolume indicator. In Section 6.2 the hypervolume indicator (HVI) gradient, and its use in algorithms are discussed. Section 6.3 starts by describing a local search algorithm (based on the HVI gradient) and a global search algorithm (based on SMS-EMOA) for the building spatial design problem, and then considers how they can be combined into a memetic algorithm. Following this, Section 6.4 describes the experimental setup for the evaluation of the different algorithms. This is naturally succeeded by Section 6.5, where the results are analysed. Finally, Section 6.6 summarises the chapter as a whole, and discusses future work resulting from the study.

6.1 Multi-Objective Optimisation and the Hypervolume Indicator

Given their importance to the introduction of the HVI gradient, key concepts of multi-objective optimisation and the HVI are briefly summarised here. A more gentle introduction is available in Section 2.2.

In continuous multi-objective optimisation problems (MOPs) the goal is to search for the candidate decision vector $\mathbf{x} = [x_1, \ldots, x_d]$ that optimises a tuple of objective functions $\mathbf{y} = \mathbf{f(x)} \coloneqq [f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})]$, simultaneously. Without loss of generality, it is assumed that each objective function $f_i : \mathbb{R}^d \to \mathbb{R}$ has to be *minimised*.

Given that the different functions will rarely have common optimal values for $\mathbf{x}$, the outcome of a MOP is usually a Pareto front of solutions, with differing values for $\mathbf{y}$. In continuous MOPs, the efficient set [38] is typically approximated by a finite set (of size μ): $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_\mu\} \subset \mathbb{R}^d$. The corresponding Pareto front approximation $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_\mu\} \subset \mathbb{R}^m$ is the image of X under $\mathbf{f}$, namely $\mathbf{y}_i = \mathbf{f}(\mathbf{x}_i),\ i = 1, 2, \ldots, \mu$.

The quality of a Pareto front, or an approximation thereof, can be measured with the hypervolume indicator (HVI), in the early literature also known as the $\mathcal{S}$-metric [106, 107]. The HVI measures the volume (or area in bi-objective cases) that is dominated by a set of points in the objective space, with respect to a reference point $\boldsymbol{\rho} \in \mathbb{R}^m$. For notational brevity the hypervolume indicator is denoted as H for mathematical use, while otherwise the more expressive abbreviation HVI is used. As such, the hypervolume indicator for Y is denoted with $H(Y)$. Given this quality measure, it is also possible to compare different Pareto front approximations (PFAs) to each

other. However, it should be noted that this measure is entirely dependent on how the reference point is chosen. In other words, the ranking of PFAs is determined in part by the value used for the reference point.

6.2 HVI Gradient Ascent Multi-Objective Optimisation

This section first introduces the hypervolume indicator (HVI) gradient in its general form. Following this, subsections for normalisation of the HVI gradient, step size adaptation for the HVI gradient, and finally update rules for the considered points are included.

6.2.1 Hypervolume Indicator Gradient

To give a derivation of the hypervolume indicator (HVI) gradient over approximation sets, it is proposed to use the so-called *set-oriented* approach: By concatenating the vectors in X the μd-vector $\mathbf{X} = [\mathbf{x}_1^\top, \ldots, \mathbf{x}_\mu^\top]^\top \in \mathbb{R}^{\mu d}$ is defined. Note that the restriction to $\mathbb{R}$ here is intentional, the gradients will be taken exclusively for the real subspace of the considered problem, while the discrete subspace remains constant. Likewise, a μm-vector $\mathbf{Y} = [\mathbf{y}_1^\top, \ldots, \mathbf{y}_\mu^\top]^\top \in \mathbb{R}^{\mu m}$ can be defined for the objective values. Furthermore, the following mapping can be introduced: $\mathbf{F} : \mathbb{R}^{\mu d} \to \mathbb{R}^{\mu m}, \mathbf{X} \mapsto \mathbf{Y}$. Using the mapping $\mathbf{F}$, the HVI can be related to the decision space: $\mathcal{H}_\mathbf{F}(\mathbf{X}) \coloneqq H(\mathbf{F}(\mathbf{X})) = H(\mathbf{Y})$. Note that this is simply the definition of a more concise symbol for the same concept.

The full HVI gradient can then be expressed as in Equation 6.1, and represents the direction of steepest improvement of the HVI for the entire Pareto Front Approximation (PFA).

$$\nabla \mathcal{H}_\mathbf{F}(\mathbf{X}) = \left[\frac{\partial \mathcal{H}_\mathbf{F}(\mathbf{X})}{\partial \mathbf{X}^{(1)}}^\top, \ldots, \frac{\partial \mathcal{H}_\mathbf{F}(\mathbf{X})}{\partial \mathbf{X}^{(\mu)}}^\top \right]^\top . \tag{6.1}$$

Subsequently, Equation 6.2 defines subgradients for each point of the PFA. Note that although subgradients are computed for individual points, their combination is not merely the direction of maximal improvement for each point, but for the whole set.

$$\frac{\partial \mathcal{H}_\mathbf{F}(\mathbf{X})}{\partial \mathbf{X}^{(i)}} = \left[\frac{\partial \mathcal{H}_\mathbf{F}}{\partial x_1^{(i)}}, \ldots, \frac{\partial \mathcal{H}_\mathbf{F}}{\partial x_d^{(i)}} \right]^\top . \tag{6.2}$$

Each subgradient can then be computed as:

$$\frac{\partial \mathcal{H}_{\mathbf{F}}}{\partial x_j^{(i)}}(\mathbf{X}) = \sum_{k=1}^{m} \frac{\partial H}{\partial y_k^{(i)}}(\mathbf{Y}) \times \frac{\partial f_k(\mathbf{x}^{(i)})}{\partial x_j^{(i)}} \,. \tag{6.3}$$

For the $m = 2$ case, when the indices are given by the ascending order of the first objective f_1, the first term of the summation can be expressed as follows:

$$\frac{\partial H}{\partial y_1^{(i)}} = y_2^{(i)} - y_2^{(i-1)} \,, \qquad \frac{\partial H}{\partial y_2^{(i)}} = y_1^{(i)} - y_1^{(i+1)} \,.$$

Note that strictly greater (smaller) values are subtracted, since any points that are equivalent in some objective, should also move the same in that objective.

Applying the HVI gradient is only possible if the gradient can be computed. Since for many problems, like the one considered in this work, the analytical expression of derivatives is not available, numerical computation of the gradient is considered as alternative. As such, the second term of the summation in Equation 6.3 may be computed numerically according to the finite difference method. For a small number h, the approximation reads,

$$\frac{\partial f_k(\mathbf{x}^{(i)})}{\partial x_j^{(i)}} = \frac{f_k(\mathbf{x}^{(i)} + \mathbf{e}_j h) - f_k(\mathbf{x}^{(i)})}{h} \,.$$

Note that $\mathbf{e}_j$ is the j-th standard basis in $\mathbb{R}^d$. Here $h = 0.01$ is chosen, which equates to a change of 10 mm (millimetre) in the building spatial design. This value was chosen such that it both represents a meaningful change to the design, and it is small enough such that it provides a sufficient accuracy to approximate the gradient. Moreover, it was ensured that the employed simulator for the evaluation of a solution's quality was sufficiently sensitive. In other words, that it gave different objective values for changes of this size.

6.2.2 Normalisation

A limitation of the HVI gradient method is the so-called *creepiness* behaviour, as analysed in [51]. Creepiness refers to how the points move towards the Pareto front in a suboptimal way. When the differences between subgradients are large, the steps taken by the points are largely unbalanced, leading to a non-uniform convergence to the Pareto front. To avoid creepiness, the subgradients are normalised according to Equation 6.4, before using them to update the original points.

$$G_{norm} = \left[\frac{\frac{\partial \mathcal{H}_{\mathbf{F}}}{\partial \mathbf{x}^{(1)}}}{\left\| \frac{\partial \mathcal{H}_{\mathbf{F}}}{\partial \mathbf{x}^{(1)}} \right\|}^{\top}, \ldots, \frac{\frac{\partial \mathcal{H}_{\mathbf{F}}}{\partial \mathbf{x}^{(\mu)}}}{\left\| \frac{\partial \mathcal{H}_{\mathbf{F}}}{\partial \mathbf{x}^{(\mu)}} \right\|}^{\top} \right]^{\top}, \quad \text{where} \quad \left\| \frac{\partial \mathcal{H}_{\mathbf{F}}}{\partial \mathbf{x}^{(i)}} \right\| = \sqrt{\sum_{j=1}^{d} \left(\frac{\partial \mathcal{H}_{\mathbf{F}}}{\partial x_j^{(i)}} \right)^2} . \quad (6.4)$$

6.2.3 Step Size Adaptation

In [98], the authors mentioned that the normalised subgradients may lead to oscillatory (even divergent) behaviour. To mitigate this effect, the step size adaptation mechanism that has been proposed in [97] is adopted as follows. In Equation 6.5 $\langle \cdot, \cdot \rangle$ stands for the dot product in $\mathbb{R}^d$. For each search point, I is calculated by the inner product of the normalised HVI subgradients in two consecutive iterations. This is used to find whether the step size should be increased, for positive values, or decreased otherwise. The subscript t on the subgradient is used to indicate the iteration.

$$I_t^{(i)} = \left\langle \left(\frac{\partial \mathcal{H}_{\mathbf{F}}(\mathbf{X})}{\partial \mathbf{x}^{(i)}} \right)_{t-1}, \left(\frac{\partial \mathcal{H}_{\mathbf{F}}(\mathbf{X})}{\partial \mathbf{x}^{(i)}} \right)_{t} \right\rangle , \qquad i = 1, \ldots, \mu , \qquad t = 1, 2, \ldots . \quad (6.5)$$

Since the inner product may vary largely between generations, the stabilisation is achieved by taking the cumulative p of this value, over t generations with exponential decay (Equation 6.6). The accumulation coefficient $0 < c < 1$ controls how much new information will be incorporated.

$$p_t^{(i)} \leftarrow (1 - c) \times p_{t-1}^{(i)} + c \times I_t^{(i)} , \qquad i = 1, \ldots, \mu , \qquad t = 1, 2, \ldots . \quad (6.6)$$

Given the cumulative inner product the value of the step size $\sigma_{t+1}^{(i)}$ for the next time step can be found according to Equation 6.7. The parameter α controls the rate of change for updates to the step size.

$$\sigma_{t+1}^{(i)} = \begin{cases} \sigma_t^{(i)} \times \alpha & \text{if } p_t^{(i)} < 0, \\ \sigma_t^{(i)} & \text{if } p_t^{(i)} = 0, \\ \sigma_t^{(i)} / \alpha & \text{if } p_t^{(i)} > 0. \end{cases} \qquad 0 < \alpha < 1 . \quad (6.7)$$

The parameters $c = 0.7$ and $\alpha = 0.8$ are used here as they were used in [97], but should ideally be tuned for the specific problem. Both $\left(\frac{\partial \mathcal{H}_{\mathbf{F}}(\mathbf{X})}{\partial \mathbf{x}^{(i)}} \right)_{t-1}$ and $p_{t-1}^{(i)}$ are initialised to zeros, such that the starting position is neutral.

6.2.4 Update

Which points are updated, and when, has a large influence on how the Pareto front approximation (PFA) changes. The obvious choice is to move the points on the current PFA. However, what to do with dominated points is not immediately obvious. In [66] the authors defined a new search direction according to which they suggested to move dominated points.

In the bi-objective case, this direction is defined as the sum of normalised gradients from two objective functions. It guarantees that dominated decision points move into the dominance cone [98]. However, such a method only considers the movement of single points, instead of a set of search points, and does not generalise naturally to higher dimensions.

Alternatively, in [97], the authors suggested to move all points, including the dominated points, according to the HVI gradient. In order to do this, the whole population is partitioned by the so-called nondominated sorting procedure [32], resulting in multiple subsets (fronts) of nondominated solutions. Subsequently, the HVI gradient is well-defined on each front by ignoring other fronts that dominate it. Since both approaches require the same number of evaluations the exact method from [97] is used here, as shown in Equation 6.8. Given that the numerical computation of the gradients requires a large number of evaluations (equal to the number of continuous decision variables) investigating alternatives that use fewer, or no, evaluations could be a promising future direction.

In this work, the step size parameter σ is initialised to $0.0025 \times (ub_r - lb_r)$ according to practical usage of the algorithm. Here ub_r and lb_r refer to the upper and lower bounds of continuous decision variable r respectively. The gradient-based update is as follows,

$$\mathbf{x}_j^{(i)} \leftarrow \mathbf{x}_j^{(i)} + \sigma^{(i)} \frac{\partial \mathcal{H}_{\mathbf{F}}(\mathbf{X})}{\partial \mathbf{x}_j^{(i)}} , \qquad i = 1, \ldots, \mu , \qquad j = 1, \ldots, d . \tag{6.8}$$

6.3 Algorithms

Each subsection here describes one of the considered algorithms. First the HIGA-MO-SC approach, as adapted from the standard HIGA-MO [97] algorithm. Second the SMS-EMOA-SC algorithm, previously introduced in [15]. Finally, a combination of the two in the form of a memetic algorithm, MEMO-SC, is considered.

6.3.1 HIGA-MO-SC

The HVI gradient method adapted to the context of building spatial design is described in Algorithm 4. There, HIGA-MO [97] is adjusted to work with the supercube representation and forms the HIGA-MO-SC algorithm. An initial population is generated with the problem specific initialisation procedure introduced in Section 5.3.1. Following this, the population is sorted according to nondominated sorting [32]. Each front is then updated separately as follows. First, the HVI gradient is computed for the continuous subspace as previously described in Section 6.2.1. Second, the HVI gradients are normalised using Equation 6.4. Third, step sizes are updated by employing Equations 6.5, 6.6, and 6.7. Finally, the old points are replaced with new points generated according to the normalised HVI gradients and the updated step sizes.

Algorithm 4 HIGA-MO-SC

```
1: input: μ, λ, σ, c, α, h
2: output: PFA based on all evaluated solutions
3: Initialise population X of μ parents as in Section 5.3.1
4: while Stop condition not met do
5:     while X ≠ ∅ do
6:         X_nds ← NDS_1(X)          ▷ Where NDS_1 returns the first front after nondominated sorting
7:         X ← X \ X_nds
8:         Compute the HVI gradient for X_nds according to Section 6.2.1
9:         Normalise HVI gradient of X_nds according to Equation 6.4
10:        Update step size of X_nds according to Equations 6.5, 6.6, and 6.7
11:        Move X_nds according to Equation 6.8
12:        X' ← X' ∪ X_nds
13:    end while
14:    X ← X'
15: end while
```

Note that HIGA-MO-SC as used here differs from HIGA-MO from [97] in two aspects. First, the step sizes are updated before moving points, rather than after. As a result the gradient information of the current iteration is immediately taken into account. Second, and most significantly, here gradients are numerically approximated. Therefore, they require a number of function evaluations equal to the number of continuous decision variables.

Discussion on whether to call this algorithm memetic or not is possible, since the HVI gradient operates on a population level. Here, it is important to note that there

are two equivalent views on the HVI gradient. One view is that the HVI gradient consists of the gradients of the hypervolume contributions. In that sense, a point can locally improve by increasing its hypervolume contribution. If this is performed simultaneously for all points – the second view – the effect is equivalent to following the set gradient of the HVI. A detailed discussion is provided in [40].

6.3.2 SMS-EMOA-SC

Algorithm 5 SMS-EMOA-SC

input: μ, MT, MC
output: PFA based on all evaluated solutions
Initialise population X of μ parents as in Section 5.3.1
while Stop condition not met **do**
    $\mathbf{x}' \leftarrow$ A uniform random individual from X
    if $U(0,1) \leq MT$ **then** ▷ Where $U(0,1)$ returns a uniform random number
        if $U(0,1) \leq 0.5$ **then**
            n_steps $\leftarrow 1$ ▷ Local move
        else
            n_steps $\leftarrow 3$ ▷ Explorative move
        end if
        Mutate binary variables in $\mathbf{x}'$ with n_steps as in Section 5.3.2
    else
        Apply polynomial mutation to each continuous variable in $\mathbf{x}'$ with probability MC
    end if
    Rescale the continuous variables of $\mathbf{x}'$ until the design reaches the desired spatial volume
    $X \leftarrow$ Select μ individuals from $X \cup \mathbf{x}'$
end while

The SMS-EMOA SuperCube (SMS-EMOA-SC) algorithm is the result of tuning the tailored SMS-EMOA in Section 5.5.3. Through the use of problem specific initialisation and mutation operators SMS-EMOA-SC as described in Algorithm 5 considers only feasible solutions. In discrete space the initialisation operator generates random building spatial designs composed of cuboid spaces consisting of a random number of cells, within the restrictions of the supercube representation. The continuous variables are initialised uniformly at random within their bounds. Either discrete mutations are applied with probability $MT = 0.4993$, or continuous mutations with probability $1 - MT$. Mutation in discrete space works by extending or contracting existing spaces to change their shape, while ensuring that these changes finally lead to another feasible

design. Since this mutation procedure can consist of multiple steps, it is possible to move into infeasible regions, and then back to feasible space. Consequently, disconnected feasible areas can be reached as well. For mutation of the continuous variables, polynomial mutation [32] is applied with probability $MC = 0.4381$. Both MT and MC are used with values as found by parameter tuning in Section 5.5.3, although somewhat different objective functions were considered there.

Note that mutation is applied either in discrete space, or in continuous space. When mutations are applied on the discrete variables a design may get a significantly altered shape. As a result the optimal settings for the continuous variables change, and mutating them at the same time may have little meaning. Further, all designs are rescaled in the continuous domain (as described in Section 4.4.3) to the same volume to be able to make a sensible comparison between them. Therefore, any changes in the discrete domain automatically also result in changes in the continuous domain. Finally, mutations in discrete space may – chosen uniformly at random – consist either of a single step, to make a local move, or of three steps, to make an explorative move.

6.3.3 MEMO-SC

Algorithm 6 shows how the SMS-EMOA-SC Section 6.3.2 and the HIGA-MO [97] algorithms are combined into a new memetic algorithm. The evaluation budget is split between the two approaches according to a given fraction $frac = 0.5$ to be used for global search. Aside from this, the behaviour is the same as for the separate algorithms.

Algorithm 6 MEMO-SC

```
1: input: μ, MT, MC, λ, σ, c, α, h
2: output: PFA based on all evaluated solutions
3: Initialise population X of μ parents as in Section 5.3.1
4: while Stop condition not met do
5:     if eval ≥ eval_max × frac then
6:         Generate a new population as in Algorithm 4
7:     else
8:         Generate a new population as in Algorithm 5
9:     end if
10: end while
```

Although different hybridisation strategies are possible, here a relay hybrid [94] is chosen. This is favoured over an alternate-hybrid for various reasons. Applying the relatively expensive HVI gradient at earlier stages of the optimisation process

may result in costly updates to points in suboptimal discrete subspaces. Furthermore, optimising the points in low quality discrete subspaces may even impede finding better solutions in overlapping discrete subspaces. For instance, only 10 % of the solutions in some subspace A may be able to improve over the Pareto front (PF) of subspace B. As such, the further the search is away from the PF of B, the more likely it is that a newly discovered solution of the higher quality subspace A is accepted into the population by the evolutionary algorithm. Despite these possible issues, evaluating alternatives to the considered relay hybrid, with appropriate consideration for the noted pitfalls, may still be worth investigating in future work.

6.4 Experiments

For the comparison of the three algorithms (SMS-EMOA-SC, HIGA-MO-SC, and MEMO-SC) two objectives are considered as discussed in Subsection 6.4.1. Following that, Subsection 6.4.2 describes the experimental setup.

6.4.1 Objective Functions

In this work two objectives are considered for the building spatial design problem, related to two disciplines: structural design, and building physics. For both objectives measurements are computed through simulations [23, 25]. Settings for each of the simulation models are described briefly in the following.

Note that for both structural design (SD), and building physics (BP), improved metrics are used here compared to the previous chapters (4, 5). For the SD objective, the number of wind load cases has been reduced to four and the magnitudes of the loads have changed. The BP objective now uses realistic heating and cooling performance like in [23], instead of only a measure of the outer surface area. Moreover, error control has been introduced in the solver of the BP simulations to prevent possible erroneous results compared to [23].

Structural Design

The structural design (SD) objective for a given building spatial design is obtained by taking the total strain energy, here defined as compliance, in N mm (newton millimetre) from a Finite Element (FE) analysis that has been performed on an SD model developed for that spatial design. An SD model is obtained by means of a design grammar, i.e. a set of design rules that add discipline specific details to a building

spatial design. Specifically, the SD grammar adds structural aspects – like structural components, loads, and constraints – to the spatial design [25].

The following SD grammar has been defined for the studies in this work: For every surface in the spatial design a concrete slab is added with thickness $t = 150\,\mathrm{mm}$ (millimetre), Young's modulus $E = 30\,000\,\mathrm{N\,mm^{-2}}$ (newton per square millimetre), and Poisson's ratio $\nu = 0.3$. Furthermore, each edge of a surface will be constrained if both endpoints of that edge have an equal z-coordinate that is at or below zero (i.e. ground level). Next, a live load case $p_{live} = 5.0\,\mathrm{kN\,m^{-2}}$ (kilo newton per square metre) in $-z$-direction is applied on each concrete slab with a surface normal oriented vertically. Finally, wind load cases are applied, with for each wind load case three load types: $p_{w,p} = 1.0\,\mathrm{kN\,m^{-2}}$ for pressure, $p_{w,s} = 0.8\,\mathrm{kN\,m^{-2}}$ for suction, and $p_{w,sh} = 0.4\,\mathrm{kN\,m^{-2}}$ for shear. Four wind load cases are defined, in positive and negative x- and y-direction respectively. The load types are assigned to all external surfaces of the building spatial design (except to the ground floor surface). This is carried out according to the orientation of the external surface normal vector with respect to the wind direction vector. Pressure is applied if they are opposing, suction if they have the same orientation, and shear if they are perpendicular to each other.

FE analysis starts with meshing all the components into finite elements and nodes. Here a structural component is divided into ten elements along every dimension, which results in 10^n elements for n-dimensional components. For each load case, loads and boundary conditions are then applied to the nodes, and stiffness relations between the nodes are obtained via finite element formulations. The discretised structural design is formulated as a sparse linear system, which is then solved by the simplicial-LLT solver from the C++ library Eigen [50]. For each load case, the strain energy for each element can be computed once the system has been solved. Finally, the objective is then easily computed as the sum of strain energies over all elements, over all load cases. Note that here, for each element, the strain energy is calculated by $\mathbf{u}^\top \mathbf{K} \mathbf{u}$, where $\mathbf{u}$ is the displacement vector of an element, and $\mathbf{K}$ is its stiffness matrix.

Building Physics

The building physics (BP) objective is computed as the sum of heating and cooling energy in kW h (kilo watt hour) that is required to keep the air of all spaces of the building spatial design within a certain temperature range during a given simulation time period. The BP design grammar adds thermal related aspects – like volumes of air, thermal separations (e.g. walls and floors), temperature set points, and temperature profiles – to the building spatial design.

The BP grammar starts by defining temperature profiles for the weather and the ground. The ground temperature is set to be constant at $T_g = 10\,°\mathrm{C}$. The temperature data of the weather is obtained from real world measured data by KNMI (Koninklijk Nederlands Meteorologisch Instituut) at De Bilt, The Netherlands [61]. Two periods are simulated, three full hot summer days starting 1976, July 2, and three full cold winter days starting 1978, December 30. The grammar initialises all spaces of the building spatial design with their volume, and assigns a heat capacity $C_s = 3600\,\mathrm{J\,K^{-1}\,m^{-3}}$ (joule per kelvin per cubic metre), a heating set point $T_h = 18\,°\mathrm{C}$, a cooling set point $T_c = 20\,°\mathrm{C}$,[1] a heating power $Q_h = 100\,\mathrm{W\,m^{-3}}$ (watt per cubic metre), a cooling power $Q_c = 100\,\mathrm{W\,m^{-3}}$, and a ventilation rate of one air change per hour. Subsequently, the thermal separations are added, with their heat conduction properties and their connections to the volumes and temperature profiles. All surfaces in the building spatial design are assigned a concrete slab with thickness $t = 150\,\mathrm{mm}$, density $\varrho = 2400\,\mathrm{kg\,m^{-3}}$ (kilogram per cubic metre), specific heat capacity $C = 850\,\mathrm{J\,K^{-1}\,kg^{-1}}$ (joule per kelvin per kilogram), and thermal conductivity $k = 1.8\,\mathrm{W\,K^{-1}\,m^{-1}}$ (watt per kelvin per metre). Additionally, each external surface is assigned insulation on the outside with thickness $t = 150\,\mathrm{mm}$, density $\varrho = 60\,\mathrm{kg\,m^{-3}}$, specific heat capacity $C = 850\,\mathrm{J\,K^{-1}\,kg^{-1}}$, and thermal conductivity $k = 0.04\,\mathrm{W\,K^{-1}\,m^{-1}}$ (values based on stone wool [13]). A warm-up period is defined for each simulation period, starting to run backwards from four days after the beginning of the actual simulation period and ending when the start of the period is reached.

For the simulation, the BP model is first abstracted as a Resistor-Capacitor (RC) network [63], where each volume or separation is modelled by a temperature point called a state. Between each temperature point a resistance is modelled, and a grounded capacitor is attached to each temperature point. The heat flux through the capacitors and resistors in the RC-network can be described by a set of first order ordinary differential equations (ODEs) [25]. This system is solved using time steps of 15 minutes using the error controlled explicit Runge-Kutta-Dopri5 solver by odeint [2]. The simulated heating or cooling of spaces is controlled at each time step by first predicting the energy demand for that time step with the system of ODEs. Then the predicted heating or cooling demand is accepted if it is lower than the available power

[1] Although these set points are close together, this does not result in issues relevant for the presented optimisation problem. Cooling may become active when it is colder outside than inside, but due to the 15 minute time steps will not result in a temperature drop below the heating set point. Further, these simulations result in a somewhat distorted view with regard to the quantitative energy performance (which is anyway not accurate because things like solar irradiation are not considered), but the qualitatively performance between spatial designs matches reality. This is sufficient because only qualitative comparisons between spatial designs are considered here.

in a space, otherwise it is set to the available power. All heating and cooling energies are summed over all spaces and time steps to finally yield the BP objective.

6.4.2 Setup

A number of aspects of the proposed approach are evaluated empirically. Specifically, a comparison is made between the three described methods: SMS-EMOA-SC, HIGA-MO-SC, and MEMO-SC. Moreover, two versions of both the HIGA-MO-SC and the MEMO-SC algorithms are considered, one with gradient step size adaptation and one without.

Note that due to the mixed-integer nature of the problem the pure HIGA-MO-SC approach cannot be expected to be competitive with the other methods. It is considered here solely to study the behaviour of the HVI gradient on the constrained landscape of this real world problem, and the value of step size adaptation. It may also be used to show that the exploration of the discrete subspace, which HIGA-MO-SC lacks, is essential to find high quality solutions, but this is not new information.

A problem with a supercube size 3333 is considered here. Meaning the supercube has three cells in with, depth, and height dimensions, and also consists of three spaces. Although in Section 4.6 it was found that for a mid-sized supercube like this constraint navigation is still reasonably simple, this problem size already consists of nine continuous variables. For the hypervolume indicator (HVI) gradient, including numerically computing the gradient (nine evaluations, one per continuous variable), this means each new point requires ten evaluations. Since the focus of this study is on analysing the behaviour of the HVI gradient, and not on constraint navigation, the problem size is considered to be sufficient here.

Given a $3 \times 3 \times 3$ supercube, 27 discrete variables (per space, so 81 in total), and nine continuous variables exist: three each for width, depth, and height. The continuous variables for width and depth are bounded in $]0.5, 20]$, while those for height are bounded in $]3, 20]$ (all in metres). This ensures all spaces in the building are sufficiently large for human occupation. During optimisation these variables are rescaled such that the volume of the building spatial designs is kept within one cubic millimetre of $V_0 = 300\,\mathrm{m}^3$ (cubic metre), as described in [23].

For these experiments each algorithm is executed 35 times with an evaluation budget of 10 000. The MEMO-SC approaches are set to switch halfway (i.e. $frac = 0.5$), and thus use 5000 evaluations each on evolutionary search and gradient search. This halfway switch is chosen in order to allow the evolutionary search to progress

sufficiently in discrete space, while also giving the gradient search enough time to advance and adjust step sizes as needed. Note that although the evaluation budgets are equal, the number of sampled points is not. During evolutionary search each evaluation equates to a sampled point, while during HVI gradient search ten evaluations are used per sampled point.

Each algorithm considers a population size $\mu = 25$. This value is chosen to ensure a high likelihood of having a well covered PFA. Moreover, it is not so large that it would prohibit applying gradient approximation to the whole population. Note that with 25 individuals the initialisation costs 25 evaluations, leaving 9975 for the rest of the process. This means HIGA-MO-SC is not split exactly in two halves of 5000. Since HIGA-MO-SC stops when it has an insufficient evaluation budget left to generate a new point (in this case 10 evaluations), it ultimately uses five evaluations less than the two other algorithms. However, this should have no significant impact on the results.

Settings for the SMS-EMOA-SC algorithm are given in Table 6.1. Parameters MT and MC control the probability to perform a discrete or continuous mutation, and the probability of mutation per continuous decision variable respectively (see Section 6.3.2 for details). A reference point of $(1.1e9, 1.1e9)$ is used as in Section 5.5. The settings for HIGA-MO-SC are available in Table 6.2, details on their values are available in Section 6.2. Finally, MEMO-SC uses settings from either of the other two algorithms, depending on whether it is in the global or local search phase.

μ	MT	MC
25	0.4993	0.4381

Table 6.1: Settings for SMS-EMOA-SC.

μ	λ	σ	c	α	h
25	25	0.0025	0.7	0.8	0.01

Table 6.2: Settings for HIGA-MO-SC.

6.5 Results

In Figure 6.1 results are shown for a single execution of the SMS-EMOA-SC, adaptive HIGA-MO-SC, and adaptive MEMO-SC algorithms. Both the Pareto front approximations (PFAs), and the points considered during the search (limited to those suffi-

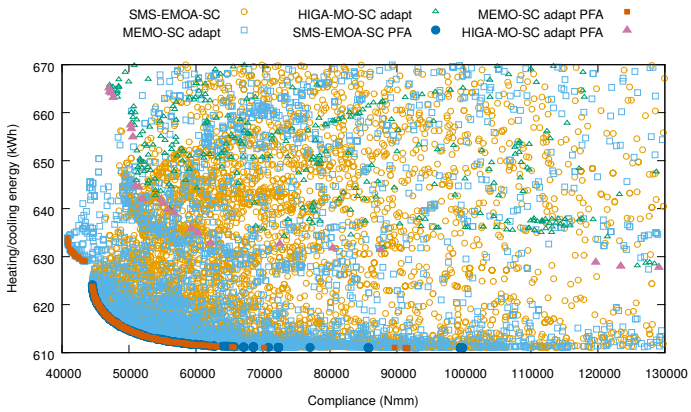
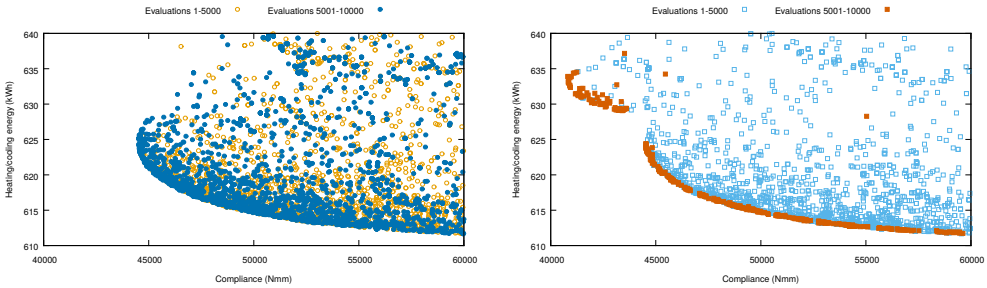


Figure 6.1: Scatter plot of the PFA region for a single execution of the SMS-EMOA-SC, adaptive HIGA-MO-SC, and adaptive MEMO-SC approaches.

ciently close to the PFAs) are shown. Evidently, SMS-EMOA-SC and MEMO-SC seem to perform similarly well. In contrast to these results – unsurprisingly – HIGA-MO-SC lags behind, unable to navigate the discrete landscape. Even so, HIGA-MO-SC is clearly able to navigate the continuous landscape within the discrete subspaces it is confined to upon initialisation.



(a) SMS-EMOA-SC. **(b)** Adaptive MEMO-SC.

Figure 6.2: Scatter plot of the PFA region for a single execution.

Figures 6.2a and 6.2b display the behavioural difference in the search strategies of the SMS-EMOA-SC and MEMO-SC approaches during the second half of the optimisation process. MEMO-SC strongly focuses on local improvements to the PFA, while SMS-EMOA-SC continues to explore as well as exploit. Another interesting observation is that for this specific execution MEMO-SC seems to find two partially overlapping discrete subspaces that both contribute to the PFA. This results in a PFA

consisting of two parts, one similar to what is found by SMS-EMOA-SC in Figure 6.2a, and an extra part in the upper-left corner of Figure 6.2b. Note that the differences in discrete subspaces that are discovered are an artifact of comparing single executions. Given a second execution, the discovered discrete subspaces might be reversed.

A visual comparison of the results over multiple repetitions is done using median attainment curves [49]. Figure 6.3 shows the high level overview of the results, including all of the approaches. Moreover, the results are split in a first and a second half, to indicate how much the algorithms improved during the second half. From this figure it is clear that, as expected, the pure HVI gradient methods are not competitive in a mixed-integer environment. Even so, it is also evident that these methods work, and effectively improve their Pareto front approximations (PFA). It also becomes clear form this figure that the use of step size adaptation has a significant effect on the optimisation progress.

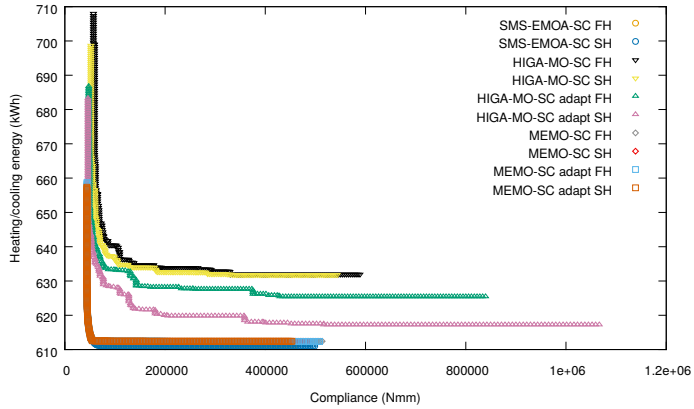


Figure 6.3: Median attainment curves per algorithm (35 repetitions each), first halves (FH) and second halves (SH).

When zoomed in on the knee point area of the median attainment curves in Figure 6.4, it can be seen that there is not much difference between the adaptive MEMO-SC, and the regular MEMO-SC algorithms. While SMS-EMOA-SC appears to be able to find better solutions in the heating and cooling energy objective, even during the first half of the search. This is a striking result, given that these three algorithms behave exactly the same during the first half of the search process. Note that despite their equivalent behaviour, given their separately generated random seeds, they can still find different results due to chance.

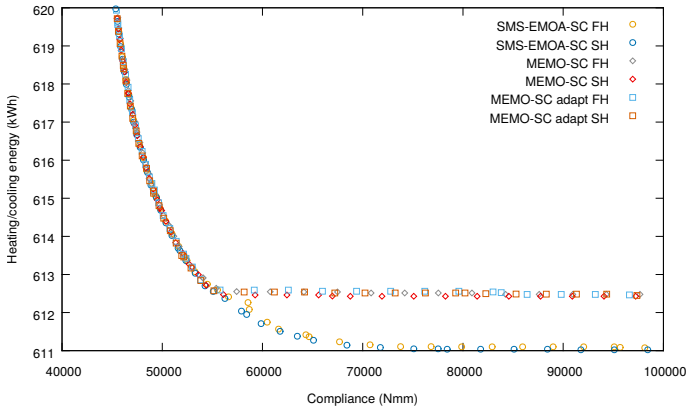


Figure 6.4: Median attainment curves per algorithm (35 repetitions each), first halves (FH) and second halves (SH); zoomed in on the knee point area.

To understand what is happening Figure 6.5 shows the nondominated solutions of every repetition for all considered algorithms. In this figure, multiple different PFAs – that are frequently found by all of the competitive approaches – can clearly be identified. These evidently represent the PFAs for different discrete subspaces. Looking back at Figure 6.4, it appears that despite using 35 repetitions, the number of times each algorithm ends up in each discrete subspace differs sufficiently to end up with differing median attainment curves. After all, the median attainment curve may be different even if one of the algorithms ends up in (for instance) the optimal discrete subspace only a single time more than the other algorithms.

Based on the points found in Figure 6.5 it is also possible to visualise the trade-off between the two objectives. In Figure 6.6 three example solutions are shown, together with some details on their features in Table 6.3. One for each objective, and one from the knee point area. For compliance it seems that long, evenly distributed walls with short floor spans are optimal for the distribution of strain across the structural elements. On the other hand, optimal energy efficiency is found by using a cubic shape and some spaces as padding to the outside, in order to provide insulation.

The union of all solutions found over all repetitions of all the approaches is taken. Based on the nondominated solutions of this collection (Figure 6.7), the objective values are normalised. From these nondominated solutions it is found that for compliance a range of $[0, 500\,000]$ can be considered, while in energy use a range of $[610, 660]$ is sufficient. All objective values then are normalised from those ranges to a $[0, 1]$ range.

Given the normalised objective values, statistics over the hypervolume indicator

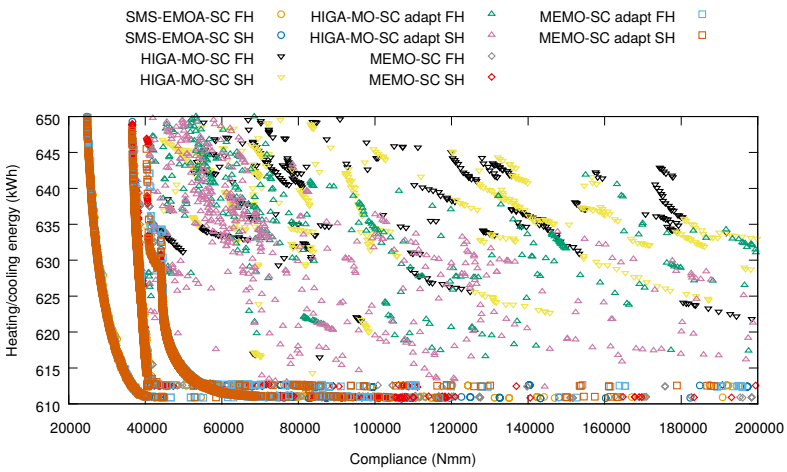


Figure 6.5: Nondominated solutions from each of the 35 repetitions per algorithm, first halves (FH) and second halves (SH); zoomed in.

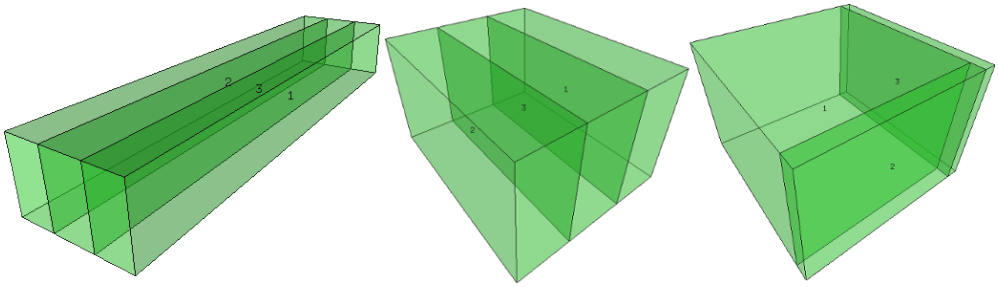


Figure 6.6: Example solutions: optimal compliance (left), a knee point solution (centre), and optimal energy efficiency (right).

(HVI) can be computed, with reference point (1,1). Table 6.4 shows these results per algorithm for the first half of the optimisation process. Considering that SMS-EMOA-SC and both MEMO-SC approaches are equivalent in the first half, it is not surprising to see their very similar performance here. Although SMS-EMOA-SC performs slightly better overall, this is purely based on chance.

Further, it should be noted that these results largely depend on which discrete subspace an algorithm ends up in. For instance, consider two partially overlapping Pareto front approximations PFA_1 and PFA_2, and two equivalent algorithms A_1, A_2, executed for 10 repetitions each. Now, by chance A_1 could end up in PFA_1 8 out of 10 times, while A_2 does the reverse. Seemingly A_1 would then be better in one objective, and A_2 in the other, although they are actually the same algorithm. In

	Compliance (N mm)	Heating/cooling energy (kW h)	Surface area (m)	Soil surface (m)	Height (m)	Longest edge (m)	Shortest edge (m)
Optimal compliance	24478.9	655.7	251	100	3.0	20.2	4.9
Knee point	37816.4	611.7	216	60	5.0	8.6	7.0
Optimal energy efficiency	435965.0	610.7	215	58	5.2	7.6	7.6

Table 6.3: Details on the features of the optimal compliance, knee point, and optimal energy efficiency spatial designs.

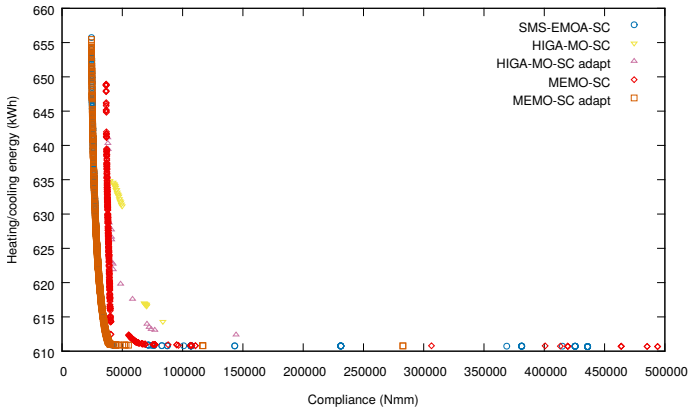


Figure 6.7: Nondominated solutions over all 35 repetitions per algorithm, second halves only.

other words, even for a simple case a reasonably large number of repetitions is required. Alternatively each discrete subspace could be analysed separately, but this is contrary to the goal of finding high quality discrete subspaces in the first place. Moreover, recall that the problem at hand does not merely consider two discrete subspaces, but many – often difficult to reach – subspaces.

Table 6.5 contains the results after completion of the second half of the search process. Once more, SMS-EMOA-SC seems to outperform all other approaches. Moreover, when comparing the results in Table 6.5 to those in Table 6.4 it can be observed that SMS-EMOA-SC shows the greatest improvement during the second phase of the search. Although this could be taken as surprising, when taken together with Figures 6.4 and 6.5, it can be postulated that SMS-EMOA-SC simply found the best PFA more often. Moreover, SMS-EMOA-SC is still able to make discrete moves in the second phase, and may therefore find a new, better, front while the MEMO-SC approaches cannot. Regardless of the reason SMS-EMOA-SC appears to be better

Algorithm	Min	Max	Mean	Median	Std. dev.
SMS-EMOA-SC	0.86052	**0.92815**	**0.88674**	**0.88899**	0.01764
HIGA-MO-SC	0.25969	0.75421	0.45662	0.45798	0.11305
HIGA-MO-SC adapt	0.40257	0.76252	0.57560	0.57260	0.09315
MEMO-SC	**0.86112**	0.90129	0.88068	0.87780	**0.01220**
MEMO-SC adapt	0.85973	0.92788	0.88049	0.87633	0.01892

Table 6.4: Statistics of the normalised HVI per algorithm after the first half, over 35 repetitions, best values in bold.

Algorithm	Min	Max	Mean	Median	Std. dev.
SMS-EMOA-SC	**0.86337**	**0.92910**	**0.89087**	**0.89048**	0.01602
HIGA-MO-SC	0.31499	0.77975	0.49822	0.48678	0.10857
HIGA-MO-SC adapt	0.49186	0.85363	0.69118	0.71397	0.08543
MEMO-SC	0.86146	0.90300	0.88163	0.87950	**0.01197**
MEMO-SC adapt	0.86193	0.92797	0.88168	0.87793	0.01843

Table 6.5: Statistics of the normalised HVI per algorithm for the second half, over 35 repetitions, best values in bold.

during the local search phase, the performance of the HIGA-MO-SC approach is also striking. It significantly improves in all metrics during the second phase, and is clearly a viable method for this problem, as long as it is given a good discrete subspace to work in.

All in all, it appears that both the SMS-EMOA-SC and MEMO-SC approaches are able to converge to good Pareto front approximations. Since this is already true after the first half of the search process, there is simply little to improve for either the HVI gradient in MEMO-SC, or the evolutionary approach in SMS-EMOA-SC during the second half. Further, it is observed that, for the problem here, the quality of the found PFA depends more on the discrete, than on the continuous decision variables.

6.6 Conclusion

6.6.1 Summary

In this work a building spatial design problem for two objectives has been considered. For this problem the shape of a building had to be optimised for structural performance and energy performance. Provided an existing mixed-integer representation, three algorithms have been applied to this optimisation problem: An algorithm based on the hypervolume indicator (HVI) gradient (HIGA-MO-SC), an adapted version of

SMS-EMOA (SMS-EMOA-SC), and a memetic algorithm combining the two methods (MEMO-SC).

Results showed that the HVI gradient method by itself could not compete with the evolutionary and memetic approaches. Considering the mixed-integer nature of the problem, this is not surprising. It has also been shown that the evolutionary approach performed slightly better during the local search phase than the memetic algorithm. However, this may be the result of larger global moves, rather than of its implied local search abilities.

Although the algorithm does not improve significantly in most cases, the HVI gradient remains useful in providing a guarantee of local convergence in the continuous subspace. The non-deterministic evolutionary algorithm cannot provide such guarantees. However, based on the results, in many cases it has a good practical performance in adjusting the continuous variables. Further, the effort spent on solving the integer problem appears to have a more significant impact on the overall result. As such, the answer to whether local search can improve over the results found during global search (RQ3) is positive, but whether it is worth the effort in practice will be situation dependent.

6.6.2 Future Work

Improvement of the MEMO-SC algorithm may be possible by focusing on moving nondominated points, rather than all points. This could reduce the number of used evaluations on points that might never reach the Pareto front, because they are either simply too far away, or worse, stuck in a discrete subspace that is completely dominated by another. Challenges herein are found in how it is ensured that there are sufficiently many points on the Pareto front, and subsequently, how to ensure they remain on the Pareto front during the search.

Other hybridisation strategies could also be explored for the MEMO-SC algorithm. In this work a relay hybrid – where first evolutionary search is applied, and then HVI gradient ascent – has been considered. An alternate-hybrid strategy, which continuously alternates between the two approaches, should also be investigated. Additionally, a comparison between HIGA-MO-SC and SMS-EMOA-SC on a single discrete subspace (such that only continuous variables are considered by both of them) could give interesting insights.

One limitation of the HVI gradient is that it cannot navigate mixed-integer space. To overcome this, surrogate models (approximations of the objective functions that

can be cheaply evaluated) can be considered. For instance, surrogate models are used for the mixed-integer case in [68]. Use of the HVI gradient is possible in this case by taking the HVI gradient of the surrogate model, rather than of the actual objective functions. Otherwise, it may be that methods integrating the HVI gradient are not well suited to problems with far more integer than continuous variables, as is considered here. To investigate this, a comparison to a problem with a small number of integer variables would be interesting as future work.

If the MEMO-SC algorithm, as it was presented in this work, is improved by the suggestions above (or by other means), such that it provides advantages beyond the evolutionary approach, new directions open up. Then, in the future, combining an improved MEMO-SC algorithm and the cooperation between superstructure and free representations presented in [23] can lead to an optimisation strategy covering all levels of the building spatial design problem. Specifically, it would allow exploration with a free representation using co-evolutionary design simulation (as in e.g. [23, 52]), followed by global search with the evolutionary algorithm, and local search with the HVI gradient method when using the superstructure representation.

Finally, despite the existence of a plethora of different measures to compare the quality of Pareto front approximations (PFAs), determining which PFA is better, or how two PFAs differ remains challenging. In particular this holds if, as in this work, there is a desire for statistical significance and comparisons are done over multiple repetitions per algorithmic approach. Moreover, mixed-integer landscapes further complicate the process, where different repetitions may end up in distinct, but possibly overlapping, discrete subspaces. Evidently, much work remains in the area of multi-objective quality measures.

Chapter 7

Mining Optimisation Data for Design Rules

Up to this point, a mixed-integer representation has been defined for the building spatial design problem in Chapter 3. Based on this representation, multi-objective evolutionary algorithms have been devised, along with problem specific operators in Chapters 4 and 5. Furthermore, the application of the hypervolume indicator gradient [40], to improve local search, was studied in Chapter 6, which resulted in a considerable amount of optimisation data.

Despite all this progress, the transfer of an optimisation result to a design expert is not merely a matter of stating "this solution is better than the previous one". For an optimised building spatial design to be used, the solution must be trusted by the design expert. To inspire such confidence in the optimised design, the optimised results should be made explainable. This can be achieved by learning heuristic design rules from the optimisation data. Given such rules, it becomes clear why the design is effective. Ideally, not only known rules that experts trust and understand are obtained, but also new insights. By combining known and new design rules it is possible for experts, and automated (e.g. co-evolutionary [23]) design systems, to improve their design process. These improved design processes can then be applied to similar problems, without another lengthy optimisation procedure. Learning these design rules, new and old, is the focus of RQ4, and as such of this chapter.

The process of learning innovative design rules from optimisation data was introduced in [33], and termed *innovization*. This concept has since been applied to a

variety of problems such as clutch brake design in [33], and truss design in [7]. Later, the learning process was interleaved with the optimisation process in [78], and further automated in [7, 31]. Furthermore, in [8] it was studied how an optimiser learns new concepts over time. Here it is investigated whether simple techniques used to verify optimisation results may also lead to innovative insights.

This work is a first step in applying innovization in building spatial design. The following contributions are made: Optimisation results are verified through data analysis of a subset of the 800 000 solutions found by multi-objective optimisation in Section 6.5. Handling a dataset of this size also results in new challenges. With this in mind, simple and computationally inexpensive analysis techniques are applied.

From here on, this chapter first introduces features to enable the discovery of heuristic design rules in Section 7.1. The preparation of the considered dataset is then described in Section 7.2. Section 7.3 evaluates the results from analysis of the data, and the implications that follow. Finally, Section 7.4 briefly summarises the study, and proposes possible directions for future work.

7.1 Features

The supercube representation introduced in Chapter 3 is a mixed-integer representation of the building spatial design problem, consisting of binary and positive real numbers. Raw data in this format is difficult to interpret in terms of building properties, making it difficult to learn directly from this data. To ease this process, this section introduces elementary features that allow building engineers to characterise a building spatial design. Such features are necessarily domain specific. However, the same process may be applied in other domains.

Given that the supercube representation is key to understanding the dataset and features, its essential components are briefly reintroduced in the following. Since it is used for building spatial design, the supercube representation considers a number of spaces that together form the building spatial design. Each space is defined as a cuboid (3D rectangle), such that the whole building consists of rectangular surfaces, like in Figure 7.1. Additional constraints ensure that the floors of all spaces are connected with the soil via other spaces, that is, in the given representation no floating or overhanging spaces may exist.

All considered features are listed in Table 7.1 with their definitions and explanations. Except for the last three, all other features are computed both for the building, and for individual spaces. Since the ordering of spaces is arbitrary, including values

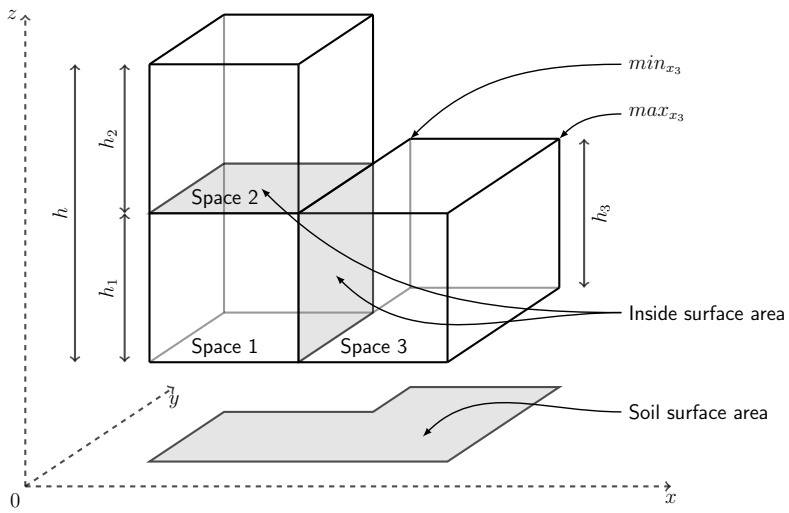


Figure 7.1: Example building spatial design, annotated with a selection of features.

for each of them in the feature set would be of little use. Therefore, statistics are taken over all spaces in a building for each feature. In particular, the minimum, maximum, mean, median, range, standard deviation and Gini index (average deviation from the mean) are considered. Since the last three features in Table 7.1 do not make sense for individual spaces (e.g. mean height of a space is equal to its height), they are only computed for the building as a whole.

Values for w, d, h are found by taking $max_* - min_*$, where $*$ corresponds to x, y, z respectively. In other words, they are simply the distance between the minimal and maximal coordinates of a given dimension. For example, the min_x and max_x of space 3 are marked in Figure 7.1. Note that these values are computed for the full design, as well as for individual spaces, as indicated for height in Figure 7.1 with h for the complete building spatial design, and h_1, h_2, h_3 for each individual space.

To differentiate between various surfaces, the following surface area definitions are used. First, to distinguish between different locations of the surfaces, a non-overlapping division is made between inside (*in_area*), outside (*out_area*), and soil (*soil_area*) surface area. Exterior surfaces are considered as outside, while interior surfaces are considered as inside. The ground floor which connects with the soil is excluded from the outside surface area, and taken as soil surface area. In Figure 7.1, examples of inside and soil surface area are highlighted (the rest is outside surface area). Second, to distinguish between walls and floors/ceilings, a division between

Feature	Definition	Explanation
vol	$w \times d \times h$	Volume of the space, or sum of spaces for the full design
short	$min(w, d)$	Shortest horizontal edge, indicator of span
long	$max(w, d)$	Longest horizontal edge, indicator of span
height	$max_z - min_z$	Height of the space or the full building spatial design
out	$sum(out\_area)$	Outside surface area, indicator of energy flow
in	$sum(in\_area)$	Inside surface area, indicator of energy flow
soil	$sum(soil\_area)$	Soil (ground floor) surface area, indicator of spread
horz	$sum(horz\_area)$	Horizontal surface area, indicator of total wall area
vert	$sum(vert\_area)$	Vertical surface area, indicator of floor and roof area
in_out	$in/(in + out)$	Ratio between inside- and outside surface area
out_vol	out/vol	Ratio between outside surface area and volume
long_short	$long/(long + short)$	Ratio between longest- and shortest horizontal edge
meanh	$sum(h \times roof\_area)/soil$	Mean height of the building
meanh_h	$meanh/height$	Ratio between the mean height and the height
height_soil	$height/soil$	Ratio between the height and the soil area

Table 7.1: Features, definitions, and explanations.

horizontal ($horz\_area$) and vertical ($vert\_area$) surface area is made. The horizontal surface area includes all floors and ceilings (so also the ground floor) while the vertical surface area consists of all walls, regardless of them being interior or exterior. Finally, the $roof\_area$ considered for $meanh$ is a part of the roof area in the building spatial design positioned at equal height.

Note that when considering a building as a whole, each surface is counted only once per considered distinction (e.g. horizontal/vertical). However, on the space level, surfaces are sometimes counted twice. That is, for two neighbouring spaces, both count their connecting surface as being part of, for instance, their horizontal surface areas. As a result, the sum of the surface areas of all spaces is not (necessarily) equal to the total surface area of the building.

In some cases, different features measure the same thing. For instance, the outside surface area of a building has an equal distribution (but not value) to the mean outside surface area of the spaces. Despite this, such features are kept to simplify data processing. In the analysis, only one representative should be used for these equivalent features, unless the differing values provide additional insights.

Additionally, some features may result in distributions similar to each other. This is particularly common for the range, standard deviation, and Gini index. However, even small differences may make one of them more valuable in distinguishing between solution classes than the other. Since, a priori, it is not known which is more useful in which situation, all of them are included.

Finally, it is noted that undefined (NAN) values may appear in a few cases. Some spaces may be disconnected (meaning they do not share a wall with another space). As a result, it can occur in a building design that none of the spaces has a neighbour, from which it follows that their inside surface area is zero. In these cases, the Gini indices of the interior surface area, and of the ratio between inside and outside surface areas will be undefined and marked as NAN (the Gini index divides by the sum of the set of spaces, which is zero in this case). However, since these are very low quality solutions, they are not considered in the analysis in the rest of this chapter. This will become clear in the next section.

7.2 Data Preparation

In order to learn heuristic rules for building spatial design, the dataset from the optimisation experiments in Section 6.5 is used. The dataset is a Pareto front and an archive from a building design optimisation that aimed for a building spatial design consisting of three spaces, with a total volume of $300\,\mathrm{m}^3$ (cubic metre). Note that while these may seem like simple building spatial designs, they already require 9 continuous and 81 binary variables to encode with the supercube representation (Section 3.2), leading to a large search space. The optimisation runs resulted in a dataset of around 800 000 solutions. Here the data is prepared for analysis in the following five steps. First, classes are defined to enable the discovery of different qualities in different groups of solutions. Second, the nondominated (ND) set is identified. Third, the knee point solution is identified. Fourth, solutions are assigned labels to link them to a class, based on the previously identified ND set and knee point. Fifth, a procedure is described to equalise the number of solutions in each class for those analysis techniques that demand this. Note that all steps are defined such that they should at least be generalisable for two-dimensional convex Pareto fronts with a pronounced knee-shape.

To be able to learn from the features defined in the previous section, the data is split into different classes. This is accomplished based on objective values, rather than features. Classification based on objective values allows for the verification of the optimisation procedure: Do design experts agree that the designs with good objective values are indeed good? In addition, it is often a combination of features that indicate a certain quality in the building spatial design, making feature based classification more complex. Further, by classifying on known good qualities of a building spatial design, finding innovative design rules would become very unlikely. Here, four categories of solutions are considered: the knee point area (KP), good in the compliance objective

(F1), good in the heating/cooling energy objective (F2), and relatively low quality solutions (BD). The aim is to data-mine for heuristic design rules that make it possible to differentiate between all of these distinct classes. For problems with more objectives additional classes F* can be added as needed.

The classification considers two primary aspects: (1) It should clearly distinguish between the classes in the objective space, and (2) It should be computationally efficient to enable processing of the large dataset of circa 800 000 points. The computational efficiency should also allow the proposed methods to generalise to larger building spatial designs than those considered here.

Since the considered classes are defined based on the nondominated (ND) set and the knee point, these have to be identified first. For ND set computation the well-known log-linear time algorithm based on sorting is employed [65]. Based on the ND set, the knee point is derived as follows. First the objective values of the ND set are normalised to a $[0, 1]$ range, where outliers beyond 1.5 times the interquartile range are set to the appropriate boundary value. Next, the Euclidean distance to the origin $(0, 0)$ is computed for each normalised ND point. The point with the smallest distance is then taken as the knee point (indicated with 'kp' in Figure 7.2), which is a reasonable approximation for the given dataset.

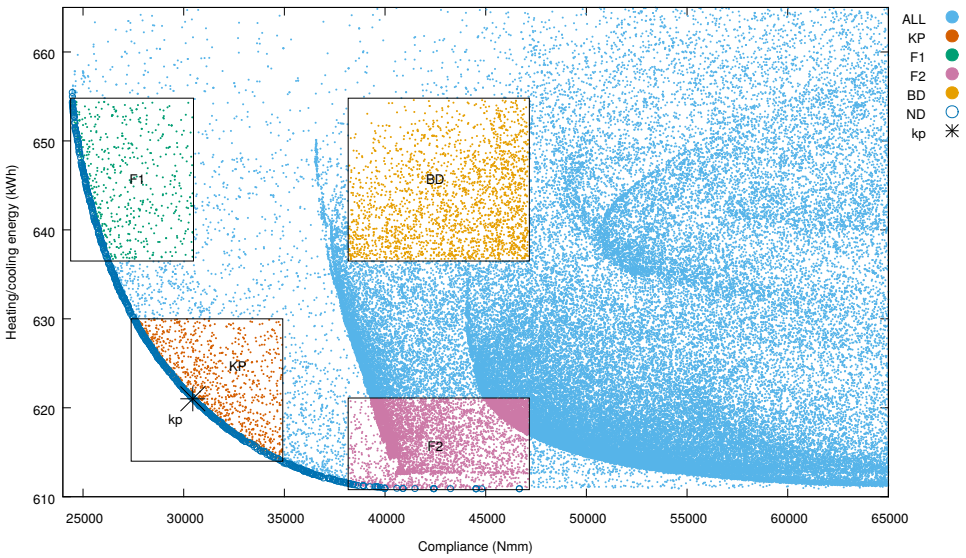


Figure 7.2: Division of data into different classes: All points (ALL), knee point area (KP), objective one (F1), objective two (F2), bad solutions (BD) included in the analysis; relative to the nondominated set (ND), and the knee point (kp). A subset of the full dataset is shown.

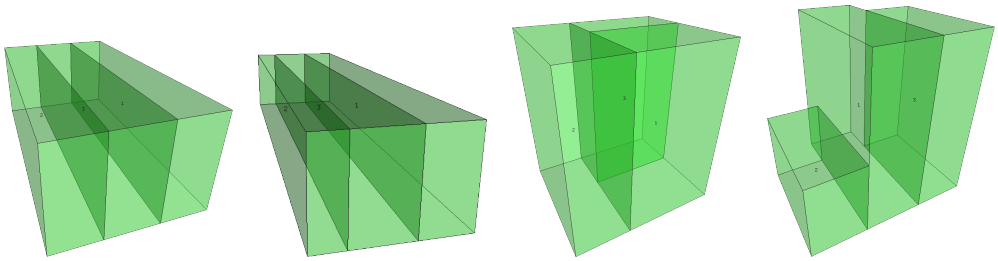
The data is then classified based on the knee point $p = (p_1, p_2)$, and the ND set. For this, the ND set is first reduced to the ND points that were not considered an outlier after normalisation, but the non-normalised values are used. In order to classify in a computationally efficient manner, each class is defined by a bounding box. These bounding boxes are found based on the length of the range of the ND set in objective one r_1, and objective two r_2. For the knee point area class (KP) the lower bound of the box is set to $(0, 0)$, while the upper bound is set to $(p_1 + r_1 \times 0.2, p_2 + r_2 \times 0.2)$. For class F1 a lower bound of $(p_1 + r_1 \times 0.35, 0)$, and an upper bound of $(p_1 + r_1 \times 0.75, p_2)$ are taken. Similarly, F2 is found with the bounds $(0, p_2 + r_2 \times 0.35)$, and $(p_1, p_2 + r_2 \times 0.75)$. Lastly, BD uses the bounds $(p_1 + r_1 \times 0.35, p_2 + r_2 \times 0.35)$, and $(p_1 + r_1 \times 0.75, p_2 + r_2 \times 0.75)$. Following this, points are assigned a label based on the box they are located in. Any remaining unlabelled points are excluded from the analysis. Note that the parameters $0.2, 0.35, 0.75$ are heuristic in their nature. It is possible to change them slightly, but overlap of the regions should be avoided.

The result of the classification process is visualised in Figure 7.2. Note that gaps are left between the different classes to improve the chances of being able to distinguish between them. If the classes would directly neighbour each other, points on the border are likely to have very similar features. This would impede learning what makes a solution perform well (or not) in one objective or the other. Future work could study how these points can be included in the analysis.

In Figure 7.3 a randomly selected example of a building spatial design is shown for each class. Although the examples for KP and F1 look similar, the design for F1 is far more elongated. This result can be expected, as the short spans (here coupled with elongated spaces) allow F1 designs to reduce the strain energy, at the cost of a larger surface area, which reduces thermal efficiency. The F2 design shows the reverse, with a much more compact design. Finally, the BD design is not very evenly arranged in the spatial sense, and shows relatively poor performance in both objectives.

After processing the dataset[1] 70 088 of the 806 430 solutions are labelled, including 5978 KP, 3400 F1, 48 482 F2, and 12 228 BD solutions respectively. Given the mixed-integer nature of the representation, multiple discrete subspaces can be seen in Figure 7.2, indicated by the apparition of different curves in the point cloud. Since the dataset is not homogeneous, the resulting classes do not have an equal number of points. For some types of analysis, however, it is critical to have equally sized classes. In such situations, excess solutions are removed from the larger classes uniformly at random. In all other situations, all labelled data is used.

[1] The dataset is available under `http://moda.liacs.nl/index.php?page=code`



(a) KP. **(b)** F1. **(c)** F2. **(d)** BD.

Figure 7.3: Typical examples of the different classes.

7.3 Results

Two techniques are used for data analysis: Box plots and decision trees. Box plots give insight into the distribution of feature data for different solution classes. As such, it may be possible to identify features that allow for a clear distinction between two or more classes. Besides, the decision tree can provide information about distinguishing features, since it generates clear rules based on such features. Moreover, it gives confidence measures for the classification of solutions to different classes. Finally, by using the learned decision tree on new data, it is possible to validate whether those rules can indeed be used reliably.

7.3.1 Box Plots

To generate box plots all labelled data is used, with each feature normalised to a $[0, 1]$ range, without removing outliers. In the plots, each class is then visualised by an individual box, such that any differences become clearly visible.

In Figure 7.4 a subset of the features is shown that appears to allow for a significant amount of distinction between the different classes. Notice, for example, how the mean of the most extended horizontal edge (long.mean) enables differentiation between objective one (F1), and objective two (F2).

Surprisingly the soil surface area (soil.mean), and the horizontal surface area (not in the figure) both showed exactly the same distributions. This occurs because all buildings considered in the labelled dataset are single-storey buildings. For such single-storey buildings, the horizontal surface area is equal to the soil surface area plus the roof area. Since these two areas are equal, the horizontal surface area is exactly twice the soil surface area, which results in their equal distributions.

It appears then that, in general, single-storey buildings have a good performance

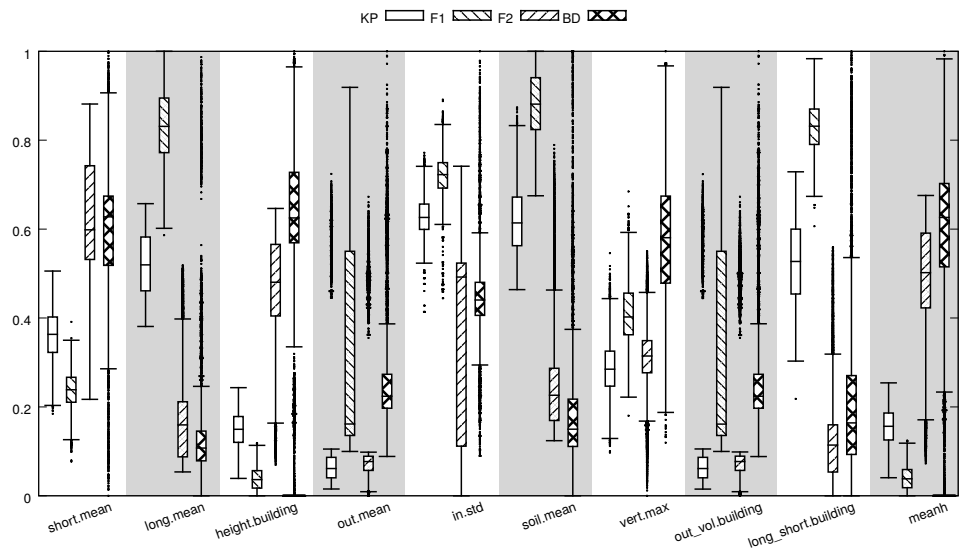


Figure 7.4: Boxplot of a selection of distinguishing features.

for the given objectives, even if they are not necessarily optimal. After all, the labelled solutions are all relatively close the Pareto front approximation. Naturally, this result may not generalise to designs with a larger number of spaces. This also indicates it may be interesting to include an even worse class of solutions in future analysis to see how things differ with even worse solutions. Additionally, a feature indicating the number of storeys of a building spatial design could be useful as well in this case. Even if just to identify this type of situation more easily.

7.3.2 Decision Trees

In order to use decision trees to their full potential, the data should be equally distributed among the classes. As such, this is carried out as described previously (Section 7.2). Since the smallest class contains 3400 solutions, the other classes are reduced to the same number of data points, resulting in a total of 13 600 solutions. This total is split into a training set of 10 200 solutions, and a test set of 3400 solutions by sampling uniformly at random. Note that as a result of random sampling, the representation of each class is not necessarily exactly equal in either of the training and test sets, but still sufficiently close. The training and test sets then consist of approximately 2550, respectively 850 solutions per class. Only labelled solutions are used, no normalisation

is applied, and no outliers of individual features are removed. In the future it may be of interest to do the same study with unlabelled solutions to see if the generated rules generalise.

Given the prepared dataset, the decision tree in Figure 7.5 was generated with the CART algorithm [27] implemented in the R package *rpart* [95] with default settings. In each node probabilities of belonging to each class are given (from left to right: BD, F1, F2, KP), as well as the percentage of the data concerned. From this figure, it can be found that the longest horizontal edge, the outer surface area, the ratio between the longest and shortest horizontal edge, and the ratio between the inner and outer surface area provide important information to distinguish between different classes of solutions.

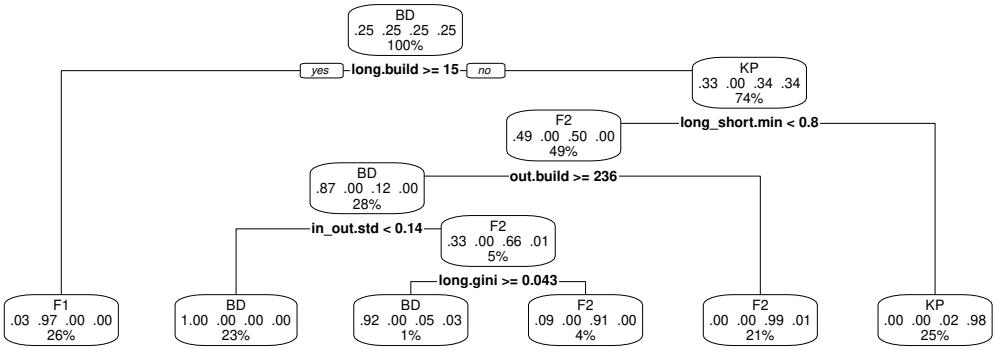


Figure 7.5: Decision tree based on data.

These rules indicate properties of a building that contribute to qualities present in different solution classes. The first split shows that relatively long buildings (long.build) are likely to be efficient in objective one (compliance). This split intuitively makes sense, since buildings that are more stretched out are likely to have short spans. Note that this is under the assumption that not just the building is stretched out, but the spaces as well (e.g. F1 in Figure 7.3).

In the other branch buildings are a bit more compact. Additionally, it can be seen that buildings where the minimal ratio of the spaces between the longest and shortest horizontal edge (long_short.min) is relatively high, are very likely to be solutions in the knee point area. This indicates that although the building as a whole is more compact, the individual spaces remain somewhat elongated to balance between the two objectives.

The primary split between low quality solutions and the second objective (energy)

is made based on the outer surface area of the entire building (out.build). Since a larger outer surface area is an indicator of a more significant loss of energy to the outside, this appears to be a sensible rule. Further, these rules provide clear pointers on how to navigate towards the PF. It may be possible to incorporate this in problem specific operators to speed up the optimisation process.

From the decision tree in Figure 7.5 it appears classification of solutions is possible with high precision. To validate this, the tree was used to classify the 3400 solutions in the test set. Table 7.2 shows the resulting predictions. All assignments were made with a confidence of at least 90 %, showing that it is possible to classify designs quite reliably. A particularly notable result is the classification of the majority of the solutions in the F2 and KP classes, which, for this dataset, is done with near perfect confidence. Not only does this provide confidence in the optimisation process, but these rules could even be useful during optimisation. By classifying new solutions based on these rules it may be possible to identify which solutions are more likely to perform well, such that expensive simulations might only be needed for those.

Prob.	.0000	.0008	.0046	.0048	.0051	.0162	.0276	.0345	.0483	.0926	.9074	.9241	.9655	.9784	.9949	.9952
BD	1758	0	0	0	728	0	54	0	0	0	0	0	0	860	0	0
F1	1649	860	0	0	0	0	0	0	0	0	0	0	891	0	0	0
F2	891	0	0	748	0	860	0	0	54	0	119	0	0	0	728	0
KP	728	0	860	0	0	0	0	891	0	119	0	54	0	0	0	748

Table 7.2: Decision tree results on the test set. Columns relate to the predicted probability of belonging to a specific class, whereas rows refer to classes. Each cell then contains the number of solutions that belong to a solution class, with a particular probability.

Based on first discussions with a design expert, it can be concluded that interesting heuristics are learned that accurately describe high quality building spatial designs. However, it seems to remain difficult to foresee the consequences of changes in feature values with respect to the objective values. In order to improve this, visual aids would be helpful. For instance, a slider controlling the weights of the structural and thermal objectives could be used to change the spatial design in real-time.

7.4 Conclusion

7.4.1 Summary

In this chapter optimisation data from previous experiments (Section 6.5) has been analysed to learn what makes a good building spatial design perform well with respect

to compliance and energy performance. This information could then be used to inform a design expert why a proposed new design should be considered. Moreover, if new design rules are learned, the expert can use them in the future.

To be able to analyse the optimisation data, this chapter included a process to go from optimisation data to practically analysable data. To this end, features have been defined that describe a building spatial design in a meaningful way for a design expert. Following that, the data was subdivided in multiple classes, representing solutions that are good in objective one (structural performance), objective two (energy performance), both objectives, or neither objective.

Data analysis has been performed through the use of box plots and decision trees. The box plots provided a clear overview of which features are likely to be useful in differentiating between the classes. Then, the decision tree produces specific rules to assign solutions to one of the previously defined classes. Further, these rules have been tested with new solutions (not used to produce the decision tree), which resulted in high precision ($\geq 96\%$) classification of solutions. Finally, from discussions with a design expert it was identified that the learned rules accurately describe what constitutes a good building spatial design.

With respect to RQ4 it can be said that it is indeed possible to learn valuable information from optimisation data, or to confirm existing empirical knowledge. Clearly, the optimised designs have been proven to conform with rules known by design experts, learned from the data.

7.4.2 Future Work

Besides generating insight, the design rules could also be useful in steering the multi-objective optimisation process. For future work, it would be interesting to investigate which moves in the optimisation process result in improvements. In other words, given an existing design, what changes to its features will, with high probability, result in an improved design. Furthermore, it may be possible to apply learned rules in co-evolutionary design processes [23]. Or, one could use the results from data mining to implement a mechanism to discard inferior solutions without the to use expensive simulations during optimisation.

The current work analyses data for a specific type of building. To generalise the conclusions, the same methods should be evaluated on a larger variety of building types. Given the computationally efficient nature of the used approach, it is probable that larger building spatial designs can be handled, however this must still be verified.

Additionally, currently only a subset of the optimisation data is labelled. As a result, it is unclear whether the learned rules generally allow the identification of, for instance, solutions that perform well in the compliance objective. It may be the case that some areas of the objective space, that have not been considered here, have similar characteristics in some features. This should be studied in the future. A challenge is how to do proper analysis with both sparse and dense areas in the objective space.

In the same line of including the currently unlabelled solutions into the analysis, it would be interesting to consider how the inclusion of solutions that lay between the currently considered classes would influence the ability to learn useful rules. Moreover, including solutions beyond the current worst class may provide new insights in how solutions change as they come closer to the Pareto front.

To improve the usefulness of the learned design rules for a design expert, it would be beneficial to have them more involved in the process. As such, an option might be introduced that allows them to use sliders to get a feel for the relation between different feature and objective values. The development of effective tools for these ideas is a challenging future direction towards a user-friendly workflow from problem specification to results and insights.

Chapter 8

Towards General Multi-Objective Mixed-Integer Optimisation

So far this thesis has focused on the development of mixed-integer methods for multi-objective building spatial design. However, general methods for multi-objective mixed-integer (MOMI) optimisation are also needed, as stipulated by RQ5. This chapter takes first steps towards this goal.

Multi-objective optimisation for either only continuous or only integer variables is widely studied, the mixed-integer case is however largely neglected. In single-objective optimisation the mixed-integer case was successfully tackled by algorithms such as the Mixed-Integer Evolution Strategy (MIES) [69]. This chapter aims to extend the MIES algorithm for the multi-objective case.

It should be noted that other multi-objective mixed-integer approaches exist, such as the Enhanced Directed Search (EDS) method [67, 100]. Another technique is the direct zigzag method [99], but it was reported to be outperformed by the EDS method [100]. However, both the direct zigzag and the EDS methods do not distinguish between integer and nominal discrete (encoded by integers) variables like MIES does. Naturally, handling these variables separately may be advantageous.

Another approach to multi-objective mixed-integer optimisation is found in [105]. This work applies Bayesian Global Optimisation (BGO) to the problem. Although mixed-integer BGO does differentiate between integer and nominal discrete variables,

BGO would require an extensive introduction and is therefore not included in this thesis. As such, this alternative is not elaborated on.

As in canonical evolution strategies (ES) [81, 91], one of the core principles of the MIES algorithm is automatic step size adaptation, i.e., the online adaptation of the strength of the stochastic perturbations. However, step size adaptation mechanisms for the single-objective case do not necessarily directly transfer to the multi-objective case. Furthermore, in [101] the authors analysed step size adaptation in evolutionary multi-objective optimisation for continuous problems, and reported the best performance when recombination is not used.

This chapter analyses mutation only approaches and step size adaptation in multi-objective mixed-integer evolution strategies. To this end, comparisons are made on both the performance in terms of diversity and convergence (combined in the hypervolume) to the Pareto front, as well as in terms of step size adaptation for the different variable types.

Section 8.1 introduces the considered algorithms and variations. Next, in Section 8.2 the experimental setup is introduced. Thereafter, the results are discussed in Section 8.3. Finally, Section 8.4 summarises the chapter and provides an overview of directions for future work.

8.1 Algorithms

Here three versions of a proposed algorithm will be introduced and compared. The core idea is the combination of the mixed-integer evolution strategy (MIES) [69] and the $\mathcal{S}$-metric selection evolutionary multi-objective algorithm (SMS-EMOA) [39]. Then, a variant is considered that does not use recombination, and one that selects offspring in a tournament during mutation. Each of the three variants is described in the following.

Firstly, an algorithm is considered that combines the canonical mixed-integer evolution strategy (MIES) as proposed in [69] with $\mathcal{S}$-metric selection and nondominated sorting as used in SMS-EMOA [39], as well as the $(\mu + 1)$ strategy considered there. In the multi-objective case recombination may have a disruptive effect on the step size adaptation mechanism. This is the result of different individuals navigating towards different parts of the Pareto front. However, this first variant will retain recombination for comparison purposes.

Secondly, an alternative is considered that only uses mutation, but is otherwise equivalent to the first algorithm. Instead of mutating the offspring resulting from recombination, mutation is applied to a uniformly at random selected individual.

Thirdly, an approach is considered that uses a tournament between mutants of the same parent as a local selection mechanism, but is otherwise equivalent to the second algorithm. In the tournament λ_k mutants are generated for the selected individual, rather than one. The mutant with the greatest hypervolume contribution is chosen as the winner, and enters $\mathcal{S}$-metric selection as usual. The idea is that competition between offspring may benefit step size adaptation. That is, the mutant with the better step size should also have the better performance, and should thus be selected.

In [69] no bounds were considered for continuous and integer step sizes. However, since these step size adaptation mechanisms were not designed with multi-objective optimisation in mind, step sizes might behave erratically and grow excessively. This could be one of the negative effects of recombining individuals that are navigating towards different parts of the Pareto front. To prevent this, step sizes for continuous and integer variables are given an upper bound equal to half the used variable range (as given in the next section). Step sizes of nominal discrete variables were already bounded in [69], and are bounded equivalently here.

8.2 Experimental Setup

To evaluate the algorithms three problems are considered: the multi-sphere (msphere) function in Equation 8.1, the multi-barrier (mbarrier) function in Equation 8.2, and the multi-objective optical filter (moptfilt) problem from [1, 5]. For the multi-objective case both the sphere and barrier functions can be adjusted with an offset for each term, such that continuous, integer, and nominal discrete optima are different in the second objective. The settings considered for each of these problems are shown in Table 8.1. Here $\mathbf{r}, \mathbf{z}, \mathbf{d}$ represent vectors of continuous, integer, and nominal discrete variables respectively, and n_r, n_i, n_d the dimensionality for each of them.

$$\begin{aligned} f_{sphere_1}(\mathbf{r}, \mathbf{z}, \mathbf{d}) &= \sum_{i=1}^{n_r} r_i^2 + \sum_{i=1}^{n_z} z_i^2 + \sum_{i=1}^{n_d} d_i^2 \rightarrow min \\ f_{sphere_2}(\mathbf{r}, \mathbf{z}, \mathbf{d}) &= \sum_{i=1}^{n_r} (r_i - 2)^2 + \sum_{i=1}^{n_z} (z_i - 2)^2 + \sum_{i=1}^{n_d} (d_i - 2)^2 \rightarrow min \end{aligned} \tag{8.1}$$

$$
\begin{aligned}
f_{barrier_1}(\mathbf{r},\mathbf{z},\mathbf{d}) &= \sum_{i=1}^{n_r} \left(r_i^2 + \theta \sin(r_i)^2\right) \\
&\quad + \sum_{i=1}^{n_z} A\left[z_i\right]^2 + \sum_{i=1}^{n_d} B_i\left[d_i\right]^2 \rightarrow min \\
f_{barrier_2}(\mathbf{r},\mathbf{z},\mathbf{d}) &= \sum_{i=1}^{n_r} \left((r_i-2)^2 + \theta \sin(r_i-2)^2\right) \\
&\quad + \sum_{i=1}^{n_z} (A\left[z_i\right]-2)^2 + \sum_{i=1}^{n_d} (B_i\left[d_i\right]-2)^2 \rightarrow min
\end{aligned}
\tag{8.2}
$$

problem	n_r	r range	n_z	z range	n_d	d range
f_{sphere}	5	$[0, 20]$	5	$[0, 20]$	5	$[0, 20]$
$f_{barrier}$	5	$[0, 20]$	5	$[0, 20]$	5	$[0, 20]$
$f_{optfilt}$	11	$[0, 1]$	N/A	N/A	11	$\{0, 1\}$

Table 8.1: Settings of the benchmark functions.

For the barrier function $\theta = 1$, and A is generated by Algorithm 6 from [69] with the parameter $C = 20$. This results in a vector of increasing integers that are occasionally swapped to create barriers for the optimisation algorithm, as shown in Equation 8.3. Further, $B_{i\in\{1,\ldots,n_d\}}$ is a set of n_d random permutations of the integer sequence $\{0, 1, \ldots, 20\}$, shown in Equation 8.4. Both A and B remain fixed for all experiments. Unlike in [69], here smooth wave-like barriers are used in the continuous part, rather than staircase-like barriers.

$$
A = \begin{bmatrix} 0 & 1 & 2 & 4 & 6 & 3 & 5 & 7 & 8 & 9 & 11 & 12 & 10 & 14 & 15 & 16 & 13 & 17 & 19 & 20 & 18 \end{bmatrix} \tag{8.3}
$$

A variant of the optical filter problem from [1, 5] with mixed variables and a second objective is considered here. Pairs of continuous and (binary) nominal discrete variables are used. When a binary variable is active, the corresponding continuous variable is used in the objective functions, otherwise it is ignored. If all bits are inactive a penalty of $(250, 1250)$ is returned.

$$B = \begin{bmatrix}
15 & 19 & 3 & 14 & 10 & 20 & 9 & 12 & 11 & 13 & 18 & 5 & 17 & 1 & 6 & 2 & 16 & 7 & 0 & 4 & 8 \\
14 & 11 & 9 & 20 & 16 & 15 & 0 & 10 & 2 & 13 & 3 & 4 & 1 & 5 & 17 & 6 & 7 & 12 & 8 & 18 & 19 \\
20 & 17 & 15 & 4 & 0 & 14 & 11 & 5 & 8 & 7 & 16 & 9 & 12 & 3 & 13 & 6 & 18 & 1 & 2 & 19 & 10 \\
14 & 5 & 18 & 6 & 9 & 11 & 8 & 2 & 20 & 7 & 12 & 13 & 3 & 0 & 10 & 15 & 16 & 4 & 1 & 17 & 19 \\
16 & 13 & 3 & 20 & 10 & 15 & 4 & 8 & 7 & 1 & 0 & 19 & 14 & 5 & 12 & 6 & 2 & 18 & 17 & 9 & 11 \\
19 & 4 & 11 & 17 & 16 & 12 & 0 & 7 & 6 & 18 & 8 & 1 & 5 & 14 & 10 & 15 & 2 & 3 & 9 & 20 & 13 \\
6 & 1 & 3 & 14 & 8 & 4 & 2 & 15 & 10 & 9 & 13 & 5 & 16 & 18 & 19 & 11 & 7 & 12 & 0 & 20 & 17 \\
16 & 1 & 17 & 15 & 0 & 12 & 11 & 18 & 13 & 7 & 19 & 14 & 2 & 8 & 9 & 3 & 10 & 20 & 4 & 6 & 5 \\
13 & 6 & 14 & 17 & 11 & 2 & 4 & 19 & 7 & 20 & 8 & 18 & 3 & 10 & 5 & 1 & 0 & 12 & 9 & 16 & 15 \\
18 & 20 & 15 & 4 & 11 & 9 & 0 & 6 & 5 & 8 & 12 & 17 & 19 & 3 & 14 & 16 & 10 & 7 & 1 & 13 & 2 \\
10 & 12 & 3 & 18 & 19 & 9 & 1 & 17 & 11 & 15 & 5 & 8 & 13 & 6 & 2 & 20 & 7 & 14 & 0 & 16 & 4 \\
6 & 19 & 0 & 14 & 1 & 7 & 17 & 12 & 16 & 18 & 11 & 4 & 13 & 8 & 15 & 3 & 20 & 9 & 10 & 2 & 5 \\
1 & 13 & 11 & 5 & 10 & 15 & 20 & 2 & 6 & 14 & 18 & 12 & 8 & 7 & 0 & 9 & 3 & 17 & 4 & 16 & 19 \\
18 & 7 & 13 & 3 & 6 & 8 & 20 & 11 & 2 & 15 & 12 & 9 & 4 & 19 & 0 & 5 & 1 & 14 & 10 & 17 & 16 \\
20 & 1 & 14 & 10 & 15 & 13 & 11 & 5 & 2 & 9 & 18 & 3 & 12 & 4 & 8 & 7 & 19 & 6 & 17 & 0 & 16 \\
5 & 6 & 9 & 19 & 14 & 3 & 12 & 17 & 13 & 11 & 15 & 10 & 0 & 2 & 4 & 20 & 18 & 16 & 1 & 8 & 7 \\
19 & 15 & 5 & 12 & 18 & 6 & 1 & 14 & 2 & 16 & 0 & 11 & 4 & 7 & 13 & 8 & 3 & 9 & 10 & 17 & 20 \\
3 & 13 & 17 & 9 & 6 & 12 & 4 & 20 & 14 & 18 & 5 & 10 & 2 & 8 & 19 & 11 & 1 & 7 & 0 & 15 & 16 \\
7 & 17 & 0 & 20 & 8 & 18 & 12 & 11 & 13 & 15 & 3 & 4 & 10 & 5 & 1 & 6 & 19 & 14 & 16 & 9 & 2 \\
4 & 1 & 3 & 15 & 19 & 8 & 16 & 14 & 10 & 6 & 18 & 5 & 0 & 7 & 20 & 17 & 11 & 9 & 13 & 12 & 2 \\
11 & 3 & 16 & 5 & 4 & 14 & 10 & 17 & 0 & 19 & 13 & 8 & 12 & 15 & 1 & 20 & 18 & 6 & 9 & 7 & 2
\end{bmatrix} \tag{8.4}$$

The original objective considers the transformation of a light wave by means of a filter that consists of layers of different materials from a limited set of materials (discrete variables). The layers can have different widths (continuous variables). The transformed waveform is compared to a target waveform and the root mean square error is measured, and to be minimised.In addition, a second objective is considered, the minimisation of the filter thickness:

$$f_{optfilt_2}(\mathbf{r}, \mathbf{d}) = \sum_{i=1}^{n_r} r_i d_i \rightarrow min. \tag{8.5}$$

8.2.1 Algorithm Settings

The canonical approach uses dominant recombination for the variables, and intermediate recombination for the step sizes as in [69]. All three approaches use single step size mode in all domains (continuous, integer, and nominal discrete), meaning a single step size per domain. Furthermore, $\mu = 10$, and a reference point $(2500, 2500)$ are considered for all approaches and objective functions. The tournament based approach uses a tournament of size 2. Step sizes are initialised to 25 % of the variable range for continuous and integer variables, and $\frac{1}{n_d}$ for nominal discrete variables. Step sizes are

bounded to $[0, 10]$ for continuous, $[1, 10]$ for integer (upper bounds equal half of the range as mentioned before), and $[\frac{1}{n_d}, 0.5]$ for nominal discrete variables. An evaluation budget of 10 000 is used in order to be able to analyse the hypervolume and step size convergence during various phases in the optimisation process.

8.3 Results

For both the msphere and mbarrier problems the canonical MIES with $\mathcal{S}$-metric selection shows the fastest convergence in Figure 8.1, and outperforms the other approaches throughout the optimisation process. The mutation only approach has a slower start, but ultimately reaches only slightly worse hypervolume values. For both the msphere and mbarrier problems the tournament approach is clearly worse than the other two. Any possible advantages of the additional selection pressure in the tournament approach are clearly mitigated by the larger number of evaluations used per generation. On the moptfilt problem all three approaches quickly converge to a stable situation.

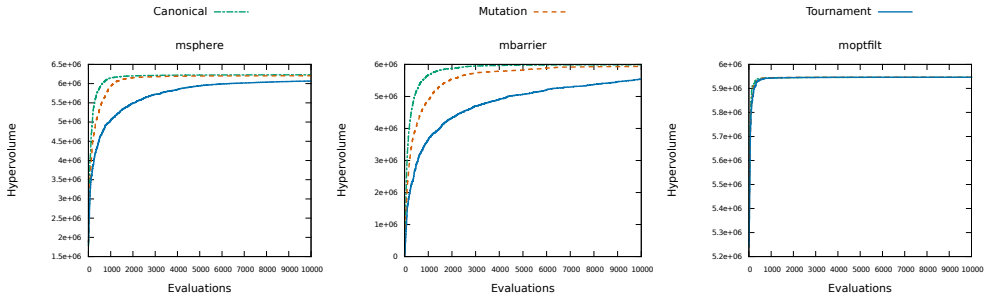


Figure 8.1: Mean hypervolume convergence over 25 repetitions.

Although it has more variables (22), it may be easier due to the smaller variable ranges that need to be searched.

The median attainment curves [49] in Figure 8.2 show that the canonical and mutation approaches find similar Pareto front approximations on the msphere and mbarrier functions, with the canonical approach remaining slightly better, as expected given the observed hypervolume convergence. All three approaches find very similar Pareto front approximations for the moptfilt problem, which suggests that they are close to the true Pareto front.

From Figure 8.3 it appears that step sizes σ for continuous and ς for integer variables stabilise reasonably well, whereas step sizes ζ for the nominal discrete variables show more erratic behaviour. However, Figure 8.4 shows that step sizes generated

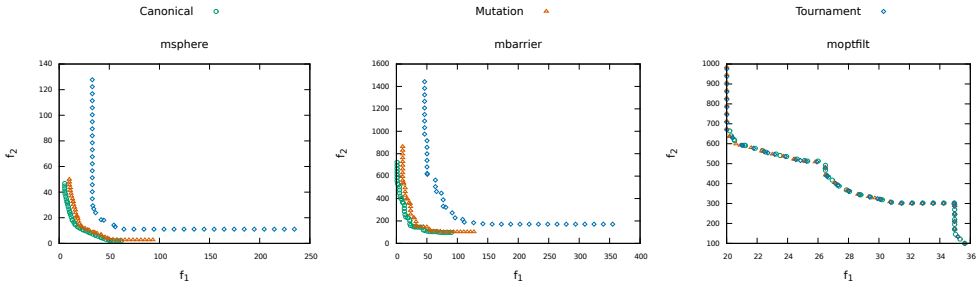


Figure 8.2: Median attainment curves over 25 repetitions.

for the offspring vary widely. The exception is the step size for continuous variables where the mutation only and tournament approaches do seem to stabilise. Although the tournament approach does so much later, this is likely due to its slower convergence. Thus, it appears only using mutation does indeed contribute to step size adaptation, but integer and nominal discrete step size adaptation have to be adjusted for the multi-objective case. Further, despite step sizes not adapting well when using recombination, it does result in better algorithm performance.

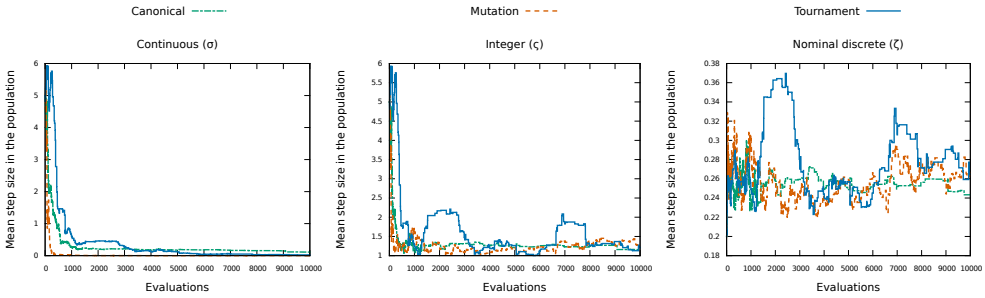


Figure 8.3: Mean step size in the population for each variable type, single run on the msphere problem.

8.4 Conclusion

8.4.1 Summary

In this chapter first steps have been taken towards general multi-objective mixed-integer optimisation, in accordance with RQ5. To this end the mixed-integer evolution strategy (MIES) [69] has been extended with multi-objective components from

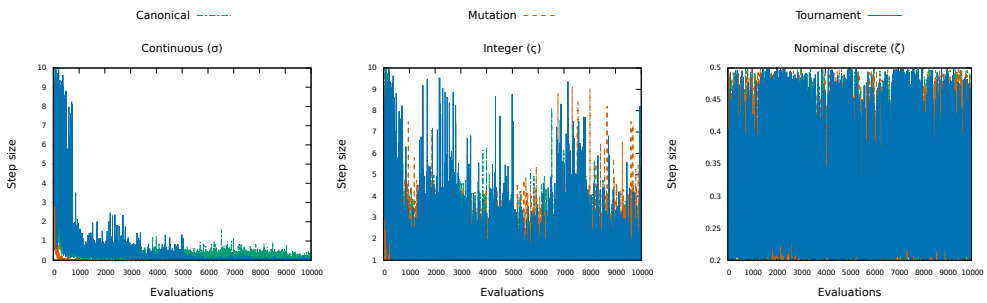


Figure 8.4: Step size per generated individual for each variable type, single run on the msphere problem.

SMS-EMOA [39]. Three variants of the considered algorithm have been introduced. Firstly, a variant that used MIES in its canonical form. Secondly, a variant that excluded recombination. Thirdly, a variant without recombination, but with a mutation tournament.

These approaches have been evaluated on three multi-objective mixed-integer problems, two test functions, and the real world problem of optical filter design. The two test functions considered all three types of variables: continuous, integer, and nominal discrete. In the optical filter problem only continuous and nominal discrete variables have been used.

The results, surprisingly,showed that the canonical variant of the algorithm performs best. Recombination seems to have a significant advantage in exploration, and is thus useful, despite its expected disruptive effect in step size adaptation for multi-objective problems. Further, the experiments showed that when only mutation is used, step sizes only adapt well for the continuous variables. Finally, mutation tournaments did not lead to sufficient additional selection pressure to offset the extra evaluation costs.

8.4.2 Future Work

Multiple directions of future work come to mind considering the experimental results. For instance, more work is needed to improve step size adaptation for the multi-objective case. First off, adaptation mechanisms have to be developed that allow integer and nominal discrete step sizes to stabilise. Second, step size adaptation that works together with recombination also has to be developed.

Another interesting question is what recombination is actually doing that makes it so effective. Is it, for instance, particularly useful for the navigation of the discrete landscape? Additionally, once the workings of recombination are better understood, it may be considered to switch partway through the optimisation process to a mutation-only approach. After all, if one is more effective in exploration, and the other in exploitation, this would be a beneficial strategy.

In addition to questions arising from the experiments presented here, the MOMIES algorithm has to be developed further in other directions as well. Throughout this thesis, the importance of constraint handling has become overtly apparent. As such, integrating constraint handling into the MOMIES algorithm is a natural next step to investigate.

Chapter 9

Applications in Building Design

Acknowledgement

This chapter is primarily based on several articles to which the author contributed, but was not the main contributor. Specifically, Section 9.1 contains parts from [21] (later extended in [25]) and [23], Section 9.2 is based on [22], and Section 9.3 summarises [24]. For all of these works Sjonnie Boonstra was the main contributor, and the author was second contributor.

So far this thesis has focused on algorithm development. First for building spatial design, and then for general applications (Chapter 8). However, the final research question (RQ6) asks how these algorithms are beneficial to real world problems. To this end, three applications of the developed algorithms are showcased in this chapter. The first two employ the problem specific building spatial design algorithm (SMS-EMOA-SC), while the third uses the general multi-objective mixed-integer evolution strategy (MOMIES).

Superstructures (such as the supercube from Chapter 3) allow optimisation algorithms to efficiently explore a limited search space. Real world design processes, however, often also consider options outside such predefined boundaries. In order to benefit from both the efficient search of an optimisation algorithm, and to explore a greater diversity of solutions, a combination of two methods is proposed in the first application domain. Specifically, co-evolutionary design simulation and evolutionary

optimisation are combined. A working example of this combination is implemented and evaluated.

Building Information Modelling (BIM) [37] is frequently used in the building design practice. However, integrating BIM with an optimisation algorithm is not straightforward. In this second application, the interaction between a BIM environment and the previously introduced SMS-EMOA-SC algorithm (Section 6.3.2) is investigated, and gaps between the optimisation environment and the BIM environment from practice are identified.

During building spatial design optimisation, the quality of the design has to be assessed. To assess the structural performance, for instance, a building structural design is needed. Generating a building structural design can be realised by heuristic grammars. The quality of fixed, pre-defined grammars is, however, not guaranteed. A new grammar is developed that can quickly generate high quality structural assignments for a given building spatial design. This is achieved by optimising the rules that operate the grammar. In this third application, the performance of this new grammar is assessed by comparing it to a baseline of structural assignments set by the MOMIES algorithm (Section 8.1).

This chapter continues in Section 9.1 with the exploration of a combination of co-evolutionary design simulation and evolutionary optimisation. Then, in Section 9.2 the gaps between optimisation tools and their practical use in a BIM environment are investigated. Section 9.3 follows with a showcase application of the MOMIES algorithm. Finally, Section 9.4 summarises these contributions, and discusses future research directions.

9.1 Combining Co-Evolution and Optimisation

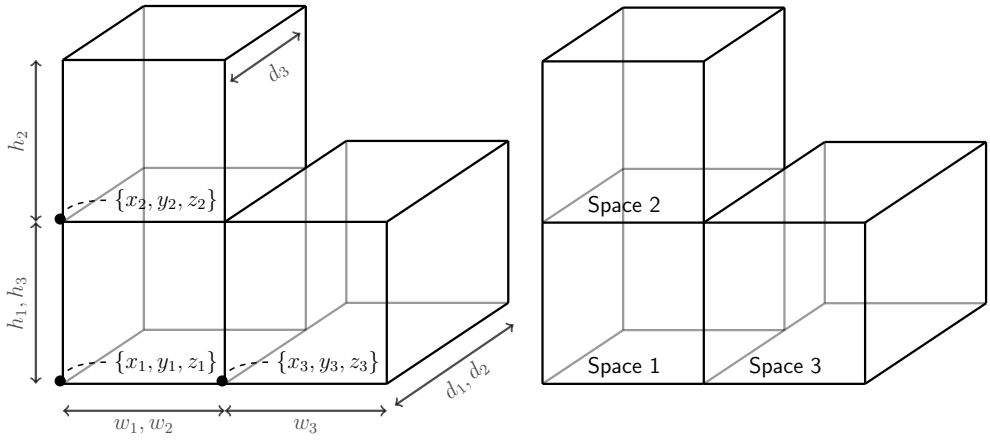
To benefit from both exploration and optimisation, a relay hybrid search using both a superstructure and a superstructure free representation is proposed. To this end, SMS-EMOA-SC (Section 6.3.2) is used with the supercube representation (Section 3.2), and co-evolutionary design (CD) simulations [52] are used with a superstructure free representation. First the superstructure free representation is introduced, together with methods to convert it to the supercube representation and back. This is followed by a description of the CD method, the relay method, and finally a case study.

9.1.1 Superstructure Free Representation

Although in Chapter 3 a superstructure representation was selected for building spatial design optimisation, a design process can also benefit from a free representation. For instance, during optimisation the number of spaces is constant in the supercube representation (Section 3.2), but this is not necessarily the case in the design process. To allow the free exploration of all possible building spatial designs, here a superstructure free representation is introduced.

This representation, called movable sizeable, describes a building by a vector of spaces $\mathbf{s}$, as shown in Equation 9.1. Here, s_i represents a space, L the coordinates of the space origin and D the geometry of the space. Here, w, d and h indicate the width in x, depth in y, and height in z coordinates respectively. Also see Figure 9.1, where an example of the movable sizeable representation is given, as well as the corresponding building spatial design.

$$\begin{aligned} \mathbf{s} = \{s_1, s_2, \ldots, s_{N_{spaces}}\} \quad \text{where} \quad & s_i = [L, D] \\ & L = [x, y, z] \\ & D = [w, d, h] \end{aligned} \tag{9.1}$$



(a) Movable sizeable representation. **(b)** Spatial design.

Figure 9.1: The movable sizeable representation corresponding to a spatial design.

9.1.2 Conversion

For a combined search process using both the supercube and the movable sizeable representations, it has to be possible to convert a design from one representation to the other. To this end, two conversion mechanisms have been developed: One that converts from supercube to movable sizeable, and another that converts from movable sizeable to supercube.

Conversion from the supercube representation to the movable sizeable representation works as follows. First, for each space in the supercube, the smallest and largest indices are extracted per dimension (w, d, h) according to Equation 9.2. Then, Equation 9.3 computes the coordinates of the origin of a space by taking the sum over all indices below its minimal index. Note that if the minimal index is one, there is nothing to take the sum of, and the sum is thus zero. Finally, the sum of all values from the minimal to the maximal index results in the dimensions of a space, per Equation 9.4.

$$\begin{aligned} i^\ell_{min} &= \min(\{i|b^\ell_{i,j,k}\}) & i^\ell_{max} &= \max(\{i|b^\ell_{i,j,k}\}) \\ j^\ell_{min} &= \min(\{j|b^\ell_{i,j,k}\}) & j^\ell_{max} &= \max(\{j|b^\ell_{i,j,k}\}) \\ k^\ell_{min} &= \min(\{k|b^\ell_{i,j,k}\}) & k^\ell_{max} &= \max(\{k|b^\ell_{i,j,k}\}) \end{aligned} \tag{9.2}$$

$$x^\ell = \sum_{p=1}^{i^\ell_{min}-1} w_p \qquad y^\ell = \sum_{q=1}^{j^\ell_{min}-1} d_q \qquad z^\ell = \sum_{r=1}^{k^\ell_{min}-1} h_r \tag{9.3}$$

$$w^\ell = \sum_{i=i^\ell_{min}}^{i^\ell_{max}} w_i \qquad d^\ell = \sum_{j=j^\ell_{min}}^{j^\ell_{max}} d_j \qquad h^\ell = \sum_{k=k^\ell_{min}}^{k^\ell_{max}} h_k \tag{9.4}$$

Next, the conversion procedure from the movable sizeable to the supercube representation is described. In order to find the dimensioning variables w_i, d_j, h_k, first the minimal (e.g. x) and maximal (e.g. $x+w$) coordinates of every space are stored. These values are then sorted in ascending order (per dimension), and duplicates are removed. The dimensioning variables are then found as $w_i = x_{i+1} - x_i$, with $i \in 1, \ldots, n-1$, and n being the number of values. For the variables d_j and h_k the same procedure is applied, but based on y and z respectively. To find which binary variables $b^\ell_{i,j,k}$ should be active for a space ℓ, the coordinates of the binary variable (or cell) are taken according to Equation 9.3, where instead of the minimal indices the indices relevant to the binary variable are used. Then, given such a coordinate, a binary variable (cell) belongs to a space, and is thus set to one, if (for example in the case of the x dimension, but this goes for y and z too) $x_i \geq x_{space}$, and $x_i < x_{space} + w_{space}$.

9.1.3 Co-Evolutionary Design Simulation

Co-evolutionary design (CD) simulations (as proposed in [73] – among others) have already been used in [52] to optimise the structures belonging to building spatial designs. The method was shown to find better designs quickly, but was also found to be sensitive to local optima. The process followed by the CD simulations was to delete poorly performing spaces, and then recover the original number of spaces and the volume by dividing spaces and rescaling the design. Given those promising results, the superstructure free representation will also be used in combination with CD simulations.

In the case study the CD simulation will use a simple selection and modification rule set. As such, only a single solution is generated after each co-evolutionary cycle. The selection and modification rule starts by evaluating the performance of each space in a design with respect to each discipline (again structural and energy performance). These performances are then normalised based on the minimal and maximal points. Here the minimal and maximal points are defined as the points containing the lower, and respectively upper bound performance values of each discipline. Subsequently, the space associated with the worst performance is removed, e.g. space s_4 in the example in Equation 9.5. Here the worst performance is measured by the shortest distance to the maximal (anti ideal) point for both disciplines. Recall that in this equation, $\mathbf{s}$ denotes a set of spaces $\{s_1, \ldots, s_n\}$, with every space s_i consisting of a set of coordinates $\{x_i, y_i, z_i\}$ and a set of dimensions $\{w_i, d_i, h_i\}$, as previously introduced in Equation 9.1.

$$\mathbf{s}\{s_1, s_2, s_3, s_4\} \rightarrow \mathbf{s}\{s_1, s_2, s_3\} \tag{9.5}$$

Next, a space is split to restore the original number of spaces, e.g. s_1 is split into s_5, s_6 in Equation 9.6. To decide which space is split, the spaces are ranked according to their distance to the minimal (ideal) point. The best performing space is chosen, unless splitting it would result in a space with a side smaller than twice the constraint on the supercube's minimal division length (see next paragraph). In that case, the next best ranked space is considered for splitting, until one is found that satisfies the constraint. By default, spaces are split in a vertical oriented plane (normal in (x, y) direction) along a line parallel to the x axis, or parallel to the y axis when the space's side in x direction is longer than its side in the y direction. A space is never split in a horizontal plane (normal in z direction). Finally, after splitting, the complete design is scaled up equally in the x and y directions to match the original design volume.

Scaling of the design completes the selection and modification rule set, resulting in a new design in Equation 9.7. The full procedure is also depicted in Figure 9.2.

$$s_1\{\{x_1, y_1, z_1\}, \{w_1, d_1, h_1\}\} \rightarrow \begin{cases} s_5\{\{x_1, y_1, z_1\}, \{\frac{1}{2}w_1, d_1, h_1\}\} \\ s_6\{\{x_1 + \frac{1}{2}w_1, y_1, z_1\}, \{\frac{1}{2}w_1, d_1, h_1\}\} \end{cases} \tag{9.6}$$

$$\mathbf{s}\{s_2, s_3, s_5, s_6\} \tag{9.7}$$

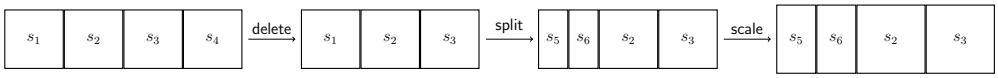


Figure 9.2: A movable sizeable design changed according to CD modification rules.

The minimal division length in the supercube ensures liveable spaces that are neither too low nor too thin to use. Exact settings of the minimal values of width, depth, and height divisions are given in the case study settings. Although this constraint does not directly apply to the MS representation, taking it into consideration vastly reduces the chances of (but not entirely prevents) generating designs that would be considered infeasible in the SC representation. Any such infeasible designs are manually excluded for this study.

9.1.4 Combination

To allow for both a wider exploration, and efficient optimisation of interesting design alternatives, a hybrid search employing both the supercube, and the movable sizeable representations is introduced. A straightforward combination of methods is the relay scheme, as shown in Figure 9.3. In this scheme, methods are executed in relays, and apart from the first, each method is initialised with the results from the previous method. For a detailed exposition of possible hybridisation schemes the interested reader is referred to [94].

In this case, CD is used as the starting method because it makes sense to explore first, before starting a time consuming optimisation process. Even so, starting with SMS-EMOA-SC could be studied in the future. Beginning with the CD method means the user needs to provide an initial design in the movable sizeable format, i.e., a collection of spaces with their locations and dimensions.

In each iteration, a method generates a number of design solutions, of which one or

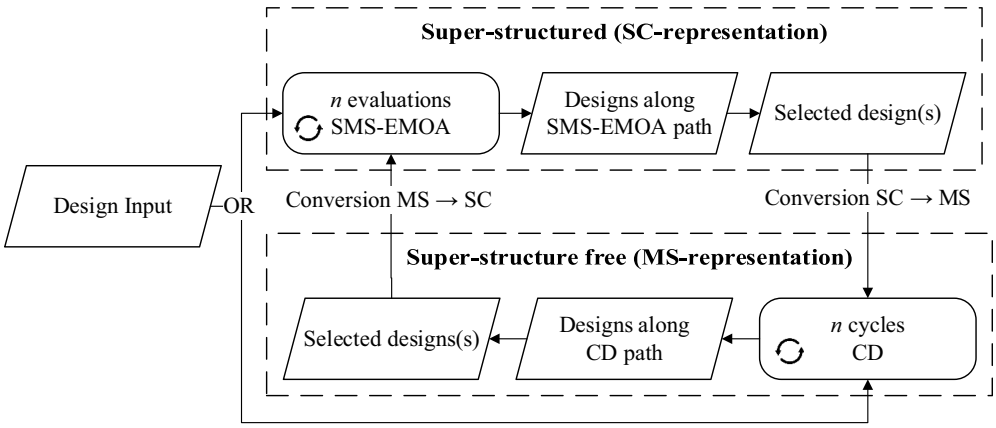


Figure 9.3: Relay hybridisation for the supercube (SC) and movable sizeable (MS) representations.

multiple have to be selected for the next method to continue with. Pareto optimality could help in this selection. However, the number of designs that are contained in a Pareto front approximation (PFA) varies. Thus, selecting the entire PFA could lead to excessive computational costs if a CD simulation is performed for each point, and in the case of SMS-EMOA-SC it could result in an initial population larger or smaller than the desired population size. Therefore, the so-called knee point solution is used. The knee point solution is selected here as the point that lies closest to the minimal (ideal) point. SMS-EMOA-SC starts with a supercube sized to contain this design, and adds an additional layer of cells on each side to allow for more variation. This means that for each iteration of the relay method the supercube size used by SMS-EMOA-SC may be different, and can thus result in the discovery of different designs. Then, given the size of the supercube and the desired population size, the population is initialised randomly according to the operator from Section 5.3.1. Completely random initialisation is chosen over the inclusion of initial solutions based on the design produced by CD, because preliminary results showed that such solutions are too dominant over randomly initialised designs, and hamper SMS-EMOA-SC in its search. For CD no additional initialisation process is needed beyond the conversion of the knee point design to the movable sizeable format.

An alternative to selection based on the knee point is hypervolume-based subset selection [28, 64]. This method selects a representative subset (a specified number of points) from a PFA and could be studied in the future as another approach to select the design – or set of designs – to continue from.

9.1.5 Case Study

To test the relay method it is used in a case study for the optimisation of a building spatial design. In addition, its performance is compared to optimisation with only SMS-EMOA. This section continues with a description of the considered objectives, and the experimental settings.

Objectives

As in the rest of this thesis, the considered objectives again concern the structural performance, and the thermal performance. To be able to evaluate these objectives for a given spatial design, additional properties are needed. These properties are assigned based on individual grammars for each objective. The settings of the structural grammar are the same as those presented in Section 6.4.1. For the thermal grammar there are minor differences, as described in the following.

Temperature regulation in the thermal grammar is done by cooling when the temperature is above $T_c = 25\,^{\circ}\mathrm{C}$, or heating when the temperature is below $T_h = 20\,^{\circ}\mathrm{C}$. For the outside temperature real world data from the Royal Dutch Meteorological Institute (KNMI) [61] is used. Specifically a time period from 01-07-2014 01:00 to 31-07-2014 24:00 is simulated. It should be noted that 2014 is considered considered to be an exceptionally warm year [60].

Considering the structural and thermal grammars, the objectives are as follows. The compliance is to be minimised, and is computed as the total strain energy for all elements and load cases in N m (newton metre). Further, the thermal performance in kW h (kilo watt hour) also has to be minimised, and is found as the sum of the heating and cooling energy used over the simulated time period.

Settings

The case study compares an individual run of SMS-EMOA-SC to the relay method which combines SMS-EMOA-SC and co-evolutionary design (CD) simulation. SMS-EMOA-SC is used here as described in Section 6.3.2, and also with the settings specified there. The relay method works as described before in Subsection 9.1.4.

As in previous experiments, the constraints presented in Section 3.2 are considered here for all methods. Further, to prevent unrealistic designs from being generated, the dimensioning variables of the supercube are constrained. This will ensure the cells are never too small. The width and depth variables use a minimum of 500 mm, whereas the height variables consider a minimum of 3000 mm.

The modification rules used in the CD method already work within the bounds of most constraints. There are two exceptions, however. Constraints on the dimensioning variables are not followed, since the MS representation does not know the concept of cells. Further, Constraint C2 C2 may also be violated by the modification rules. To prevent infeasible designs from being transferred to SMS-EMOA-SC, these constraints are check manually for this study.

Both the individual SMS-EMOA-SC method, and the relay method share a number of settings. They consider a volume $V_0 = 300\,\mathrm{m}^3$ (cubic metre), for a building spatial design consisting of eight spaces.

Specific to the individual run of SMS-EMOA-SC is the use of a supercube with three, two, and five cells in the width, depth, and height dimensions respectively. This supercube is then initialised at random with the operator from Section 5.3.1. Furthermore, SMS-EMOA-SC is executed for 5000 evaluations.

The relay method starts with the CD method and a building spatial design consisting of two spaces in x, one space in y and four spaces in z direction. All of these spaces measure 3500 mm in width, 3570 mm in depth, and 3000 mm in height. Five iterations are considered for the relay method, each consisting of 10 evaluations for CD, and 1000 for SMS-EMOA-SC. This results in a total of 5050 evaluations, comparable to the 5000 from the individual execution of SMS-EMOA-SC.

9.1.6 Results

Results of both the individual SMS-EMOA-SC run, and the relay method are shown in Figure 9.4. A first observation is that the solutions in the Pareto front approximation (PFA) found by SMS-EMOA-SC dominate most of the solutions found by the relay method. The relay method did, however, find better performing solutions in some of its iterations with regard to structural design (SD). For instance, iteration 1 of the relay method finds solutions with an SD performance of around 20 N m, versus approximately 25 N m for the best solution found by SMS-EMOA-SC. However, only two out of five iterations were able to compete with SMS-EMOA-SC on this part of the PFA. This shows that restarting the search with a new supercube shape is of limited use when 1000 evaluations are used per iteration. Regardless, a greater diversity of designs is explored by the relay method, even if not many are competitive.

When the first iteration of the relay method was allowed to continue running until 5000 evaluations were reached, however, it was able to outperform most of the PFA discovered by SMS-EMOA-SC individually. This suggests two things. Firstly, more

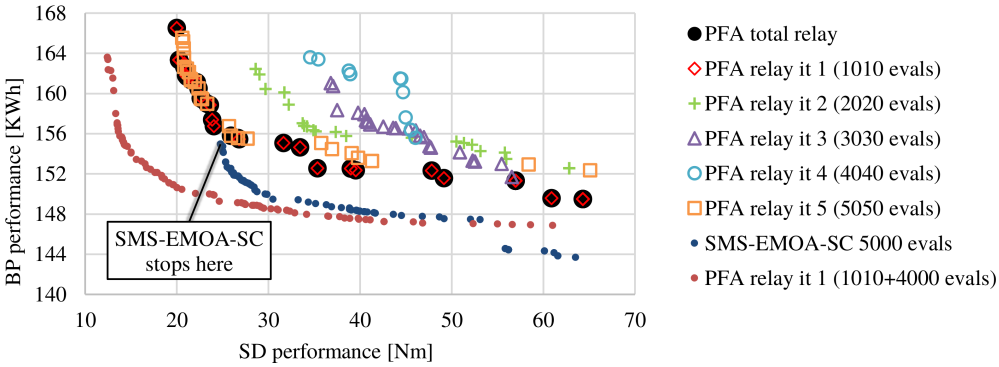


Figure 9.4: Pareto front approximations for each iteration of the relay method, a single run of SMS-EMOA-SC, and an extended run based on the first iteration of the relay.

than 1000 evaluations are needed by SMS-EMOA-SC to converge with the considered settings. Secondly, the CD method is able to suggest interesting search regions. The question remains, however, why the relay method did not find further improvements after the first iteration.

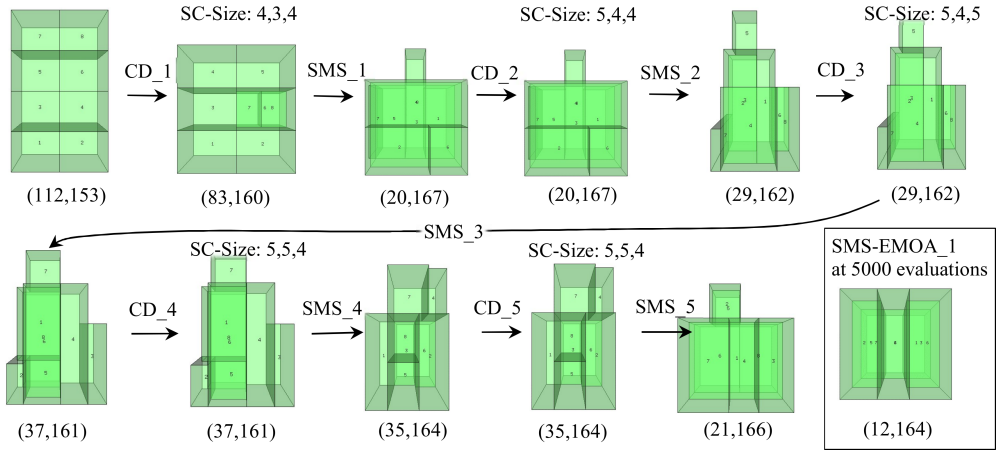


Figure 9.5: The path of the relay method, i.e. selected designs (knee points) after each optimisation run. Here SC stands for supercube.

Figure 9.5 provides some insights into the search process of the relay method. For each iteration, it shows both the input and output designs for CD as well as SMS-EMOA-SC. Here it becomes clear that although CD discovers an improved knee point design in the first iteration, in subsequent iterations it did not manage to do so. As a result, the variations in the supercube size are in fact solely based on the output of

SMS-EMOA-SC itself. This also explains why the designs in Figure 9.5 only change after the first iteration during execution of SMS-EMOA-SC, and not when CD is performed. What is also notable, is that the supercubes for iterations four and five are the same. Despite that, they unveiled very different PFAs (Figure 9.4). Evidently, at least in these early stages of the optimisation process, there is great variation in the progress of SMS-EMOA-SC towards the Pareto front.

9.2 Spatial Design Optimisation in Practice

Acknowledgement

The author gratefully acknowledges the contributions of Jeffrey van den Heuvel, whose efforts in the interaction between the optimisation tool and BouwConnect have lead to many new insights.

Recently, optimisation has received more attention in the building design practice. This attention results from stricter demands on performance measures, but also from the increasing number of objectives and disciplines that are taken into consideration. Since all of these aspects result in an increased complexity, building design optimisation can no longer be handled adequately by humans alone, and optimisation techniques are necessary.

BIM adds structure to the data used in the AEC (Architecture, Engineering, and Construction) industry, which simplifies the collaboration between different disciplines. However, BIM is also complex due to all the information it includes, which makes integration with optimisation techniques difficult. Further, changes to a BIM model can result in changes to the number of variables to be considered during optimisation.

Given the prominence of both building design optimisation research, and the use of BIM in building design practice, here the interaction between them is studied. Particularly, the aim is to identify the barriers that keep designers from using optimisation methods in a BIM context. To this end a case study is presented on the interaction between a designer using a BIM environment and an optimisation method.

Despite these challenges, here the combination of BIM and optimisation is investigated. First the considered BIM environment is discussed, along with its integration with building spatial design optimisation. This is followed by a study of how a designer interacts with an optimiser. Finally, the results of this case study are evaluated.

9.2.1 Integrating BouwConnect BIM and Optimisation

BouwConnect BIM[1] is used in the following study. This BIM tool has a product library that describes product parts with data objects. Each data object gives information on things such as the material, shape, and function. Further, a building model environment makes it possible to place products in a building spatial design. This environment is also able to compute metrics for energy, daylight, ventilation, and fire safety. All of this is, naturally, based on the spatial design and the included products.

The connection between BouwConnect and the optimiser is facilitated by an XML (eXtensible Mark-up Language) file. From this file the supercube representation can be extracted to be used by the optimiser. On the other hand, when transferring data to the BIM environment, building elements that describe a space are extracted form the XML file.

9.2.2 Case Study

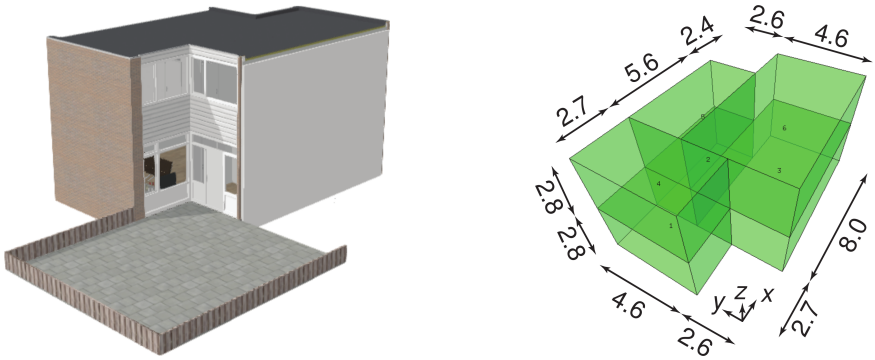
In this case study, a building design from BouwConnect BIM is modified based on optimisation results. To do so, the following procedure is considered. An existing design is selected, the design is then transferred to SMS-EMOA-SC (Section 6.3.2), and optimised. Based on the optimisation results, the design in the BIM environment is then modified. Finally, a comparison is made between the initial and the new designs, while also analysing the process.

Figure 9.6 depicts the selected design, which cuboid shapes fit well with the limitations of the supercube representation. Note that the light grey wall in Figure 9.6a is shared with a neighbouring building, and the same goes for the wall opposite to it. Figure 9.6b includes the dimensions (in metres) of the design, as used in the experiments.

Although some parts of the conversion process from BouwConnect to the optimiser were automated, others could not. Particularly, BouwConnect defines building elements, whereas the supercube representation considers spaces. As a result, spaces had to be defined manually based on the building elements, before they could be used by the optimiser.

The objectives, as well as the settings used form SMS-EMOA-SC considered in this case study are the same as those described in Section 6.4. Further, the constraints from Section 3.2 are also used again. Based on the design in Figure 9.6 a fixed volume $V_0 = 357\,\mathrm{m}^3$ is considered, for a supercube with six spaces, and three divisions each

[1] `www.bouwconnect.nl`



(a) BIM model in BouwConnect. **(b)** Spatial layout with dimensions in metres.

Figure 9.6: The initial design considered in the case study.

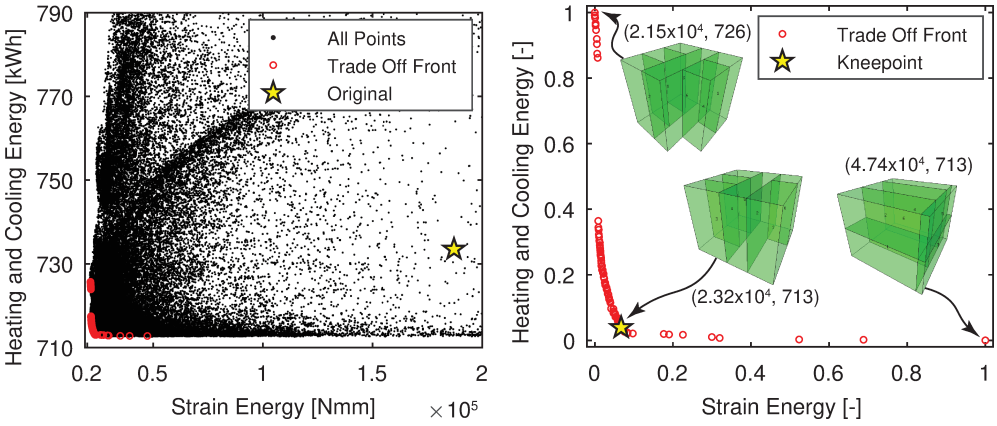
in width, depth, and height. Note that although the supercube settings are based on the original design, SMS-EMOA-SC starts from designs that are randomly initialised within those supercube restrictions. Finally, SMS-EMOA-SC is executed 10 times, for 10 000 evaluations.

To select an optimal design from the Pareto front approximation (PFA) produced by SMS-EMOA-SC, the following procedure is employed. First, the PFA is taken over all results from the ten runs. These solutions are then normalised to a $[0, 1]$ range, after which the point closest to the origin $(0, 0)$ is selected. The chosen solution is imported into BouwConnect and, based on the original, information is added that is necessary for the calculations BouwConnect performs.

9.2.3 Results

Figure 9.7a shows a scatter plot of the performances for designs found during the ten runs of SMS-EMOA-SC, as well as the original design, and the Pareto front approximation (PFA). Vast improvements are shown over the original design for both objectives considered by the optimiser. In Figure 9.7b a closer look at the trade-off between solutions in the PFA is provided. The visualisations of the knee point design and the extremal solutions for both objectives also give insight in how the design changes for different objectives.

The selected knee point design, together with its dimensions in millimetres, is visualised in Figure 9.8. Since this design is considerably different from the original (Figure 9.6) the designer had to make a number of additional design decisions in BouwConnect. Particularly notable is that the walls parallel to the y direction are assumed



(a) Scatter plot without outliers ($< 5\,\%$). **(b)** Normalised Pareto front approximation.

Figure 9.7: Optimisation results for ten runs of SMS-EMOA-SC.

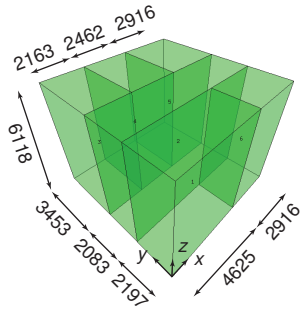


Figure 9.8: The knee point design found by SMS-EMOA-SC, including dimensions in millimetres.

to be fully shared with other buildings, although they are longer than in the original design, and as such result in a larger shared surface. Consequently, the orientation of some windows also had to be changed. Another change was the reassignment of the functions of each space, since the spaces are distributed very differently in the new design.

A comparison is made between the original and the optimised design based on various characteristics generated by BouwConnect and SMS-EMOA-SC, as shown in Table 9.1. Most notable is that, although the volume of the design is maintained, the floor area has been reduced by 50 %. On the other hand, SMS-EMOA-SC was very effective in reducing the objectives it considered, strain energy, and heating and cooling

energy. This also resulted in a reduction of the total annual energy use[2]. The energy index,[2] on the other hand, takes into account the floor area in its computation. Since the floor area became smaller, the energy index be came worse (higher). Specifically, the index changed from $32.82/118 = 0.28\,\mathrm{MW\,h\,m^{-2}}$ to $25.45/59 = 0.43\,\mathrm{MW\,h\,m^{-2}}$. The differences between the values for the heating and cooling energy and the total annual energy use can also be explained. Heating and cooling energy is computed over just six days (details in Section 6.4.1), while the total energy use considers a full year and takes ventilation, hot water, and lighting into account on top of heating and cooling.

source	characteristic	original design	optimised design
SMS-EMOA-SC	volume	$357\,\mathrm{m^3}$	$357\,\mathrm{m^3}$
BouwConnect	floor area	$118\,\mathrm{m^2}$	$59\,\mathrm{m^2}$
SMS-EMOA-SC	strain energy	$1.871 \times 10^5\,\mathrm{N\,mm}$	$0.2322 \times 10^5\,\mathrm{N\,mm}$
SMS-EMOA-SC	heating and cooling energy	$733.5\,\mathrm{kW\,h}$	$713.3\,\mathrm{kW\,h}$
BouwConnect	total annual energy use[2]	$32.82\,\mathrm{MW\,h}$	$25.45\,\mathrm{MW\,h}$
BouwConnect	energy index[2]	1.85	2.03

Table 9.1: Comparison of characteristics between the original and optimised designs.

An important part of this study is the identification of practical aspects that complicate the use of optimisation for building spatial designs in BIM environments. With this in mind, a first remark is that the optimised design is not practical, as evidenced by the floor area being reduced by half, even though for practical applicability it would be essential to keep this (near) constant. While maintaining the floor area, rather than the volume, would be a simple change for an optimisation expert, changing the behaviour of the optimiser is non-trivial for designers using a BIM environment.

Due to the mismatch in how the two tools describe the building design, the transfer of designs between them could only be automated to a limited extend. Specifically, BouwConnect describes the design with building elements (which is usual in BIM), whereas the supercube representation considers spaces. To be able to fully automate the conversion process, either the BIM environment, or the optimisation environment will have to be adapted to take this difference into account.

Adding information that is essential to the computations in BouwConnect to the optimised designs also turned out to be labour intensive. To streamline this process, properties from similar existing designs could be transferred to the new design in an automated manner. In cases where some properties of the new design differ from any

[2] According to Dutch regulation NEN-7120

known design, these could be indicated to a human designer to be resolved manually. Even though that would still require some work, it reduces the number of manual design decisions.

9.3 Structural Design Optimisation

Structural design is one of the two disciplines – together with architecture (aesthetics, spatial design) – that has the most influence on building designs during the early design stages. These two disciplines also interact strongly. After all, in practice a building spatial design requires a structural design to be meaningful. Which parts of the spatial design are occupied is determined by the structural design, and as such also influences what the design looks like [58].

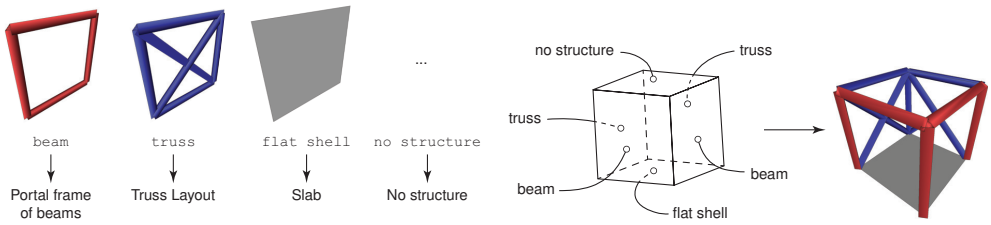
Here a grammar is introduced that aims to generate high quality structural designs in an efficient manner. This contrasts with the handcrafted, and simplistic grammar used earlier in Section 6.4.1. With a grammar that is capable of quickly generating structural grammars it is possible to support architects, structural engineers, and spatial design optimisation algorithms. Architects benefit from feedback on the structural performance of their designs, while engineers are aided in their job by the grammar. For optimisation algorithms the benefit lies in being able to efficiently assess the performance of spatial designs. This is particularly helpful because optimisation frequently requires the evaluation of a large number of designs.

This section continues with an introduction of some required structural design concepts. Then the design response grammar is introduced, and evaluated in a case study. Finally, the results are presented and discussed.

9.3.1 Structural Design

Here the structural design is described in largely the same way as in the simulator from [25]. To produce a structural design for a spatial design, the spatial design is segmented in rectangles and lines to appropriately handle assignments concerning multiple spaces or surfaces. Every rectangle (a surface belonging to one or more spaces) may then be assigned one of four structural types. Specifically, a beam, truss, flat shell, or no structure as shown in Figure 9.9a. Using these types, a structural design can be generated as in the example in Figure 9.9b.

Following the assignment of types to the rectangles, the rectangle and line rules from [24] are employed. These rules are needed because neighbouring rectangles or



(a) Possible structural types. **(b)** Generated structure based on types.

Figure 9.9: Structural types and their assignment to rectangles.

lines may have conflicting type assignments. To resolve such conflicts these rules define a procedure to follow, and a preference among the types in case one has to be chosen.

During simulation, loads are assigned to the rectangles to compute the performance (see [25, 24] for details). In some cases the rectangle where a load should be applied has no structure assigned to it. To appropriately transfer the load to the relevant components, such a rectangle is assigned a low stiffness flat shell. By using a low stiffness, these components have no influence on the overall stiffness. Note that rectangles without structure are not always replaced by a low stiffness flat shell, but only when needed.

9.3.2 Design Response Grammar

To efficiently provide high quality structural assignments for building spatial designs, a design response grammar is proposed here. This grammar assigns structural types based on a design response, instead of the grammars used in the rest of this thesis, which used very basic assignment rules. The response used here is the strain energy of a substitute component. Such substitute components are used because building spatial designs do not have a structural response of their own. Based on the response, the grammar will then replace these substitute components by beams, trusses, flat shells, or nothing. Substitute components work similarly to other components, and use the rectangle and line segment rules described in [24]. These rules check whether the line or rectangle belongs to a wall or floor, and resolve type conflicts between neighbouring rectangles and lines. Such conflicts may occur when different types have been assigned. In this case, a type is selected based on a ranking of the types, and can thus be resolved for the relevant rectangles/lines. In this ranking, the substitute component always comes last. As a result, already assigned types are always preferred over the substitute.

Each of the four types that can replace the substitute component is appropriate for some kind of loading. Trusses correspond with shear loading, beams with in-plane normal loading, and flat shells with out-of-plane loading. Finally, when non of these loading types apply the no structure type is appropriate. These loading types relate to the stiffness response of the substitute component. The stiffness is separated into bending, normal, and shear values, which are associated with the out-of-plane bending, in-plane normal, and in-plane shear loads respectively.

Equations for the out-of-plane behaviour already exist, as described in [10]. The in-plane behaviours are separated here according to Equation 9.8. Here, ν is Poisson's ratio, and E denotes Young's modulus in $\mathrm{N}^2\,\mathrm{mm}$. For the interested reader, a detailed description is available in [25, 24].

$$\underset{\text{total in-plane}}{\frac{E}{1-\nu^2}\begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & \frac{1-\nu}{2}\end{bmatrix}} = \underset{\text{normal in-plane}}{\frac{E}{1-\nu^2}\begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & 0\end{bmatrix}} + \underset{\text{shear in-plane}}{\frac{E}{1-\nu^2}\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & \frac{1-\nu}{2}\end{bmatrix}} \tag{9.8}$$

The basic idea of the grammar is that the structural design model is updated iteratively. Based on the spatial design, the building is partitioned into rectangles. Initially, each rectangle is assigned a substitute component to be able to compute a response (i.e. a performance measure) for each rectangle. Based on the performances, the substitute rectangles are clustered, and new component types are assigned to the cluster(s) with the highest mean strain energy. For each rectangle in a cluster the shear, normal, and bending responses are then checked to either assign a truss, beam, or respectively flat shell. If none of them are assigned, the rectangle receives the no structure type. The loop then starts again, until a stopping criteria is reached.

A more detailed description is given in the following based on Algorithm 7. The initialisation (line 1) simply assigns a substitute component to every rectangle. Then the loop begins on line 3, and continues for η_{conv} iterations. Such an iteration i starts by generating a structural design (based on rectangle and line rules from [24]), computing the total design response $U_{tot,i,j}$ (the sum of shear, normal, and bending strain energies) of all substitute rectangles j, and clustering them based on this response.

Here clustering is done by considering multiple runs of the k-means algorithm [72] and selecting the best result as follows. Cluster sizes from 4 to 10 are evaluated for 50 runs each. Then, for each cluster size k, the run with the lowest variance $\sigma_{sum,k} = \sum_{i=1}^{k} \sigma_i$ is stored. Next, for these values the second order change $\sigma''_{sum,k}$

Algorithm 7 Iterative replacement of substitute components

1: Assign substitute type to all rectangles
2: $i = 0$
3: **while** $i < \eta_{conv}$ **do**
4: Evaluate structural design generated based on rectangle and line rules [24]
5: Compute design response $U_{tot,i,j}$ for all substitute rectangles j
6: Cluster substitute rectangles based on $U_{tot,i,j}$
7: **while** $n_{subs,0} - \lceil n_{subs,0}/\eta_{conv} \rceil \cdot i < n_{subs,i}$ **do**
8: **for all** rectangles j in the cluster with the highest mean value **do**
9: Compute individual responses $U_{bend,i,j}$, $U_{norm,i,j}$, $U_{shear,i,j}$
10: **if** $U_{tot,i,j} < \eta_{noise} \cdot U_{tot,mean,0}$ **then**
11: **for all** $k \in [0, 1, 2]$ **do**
12: **if** $c_k = 1$ AND $U_{shear,i,j}/U_{tot,i,j} \geq \eta_{shear}$ **then**
13: Assign truss to rectangle j and go to line 21
14: **else if** $c_k = 2$ AND $U_{norm,i,j}/U_{tot,i,j} \geq \eta_{norm}$ **then**
15: Assign beam to rectangle j and go to line 21
16: **else if** $c_k = 3$ AND $U_{bend,i,j}/U_{tot,i,j} \geq \eta_{bend}$ **then**
17: Assign flat shell to rectangle j and go to line 21
18: **end if**
19: **end for**
20: **end if**
21: **if** j has type substitute **then**
22: Assign no structure to rectangle j
23: **end if**
24: **end for**
25: **end while**
26: $i = i + 1$
27: **end while**

is computed according to Equation 9.9, and finally, the one with the highest value is chosen.

$$\sigma''_{sum,k} = \sigma_{sum,k+1} + \sigma_{sum,k-1} - 2\sigma_{sum,k} \tag{9.9}$$

Continuing with Algorithm 7, a while loop is entered on line 7 that repeats based on the initial number of substitute rectangles $n_{subs,0}$, and the number of substitute rectangles in the current iteration $n_{subs,i}$. Then, for every rectangle in the cluster with the highest mean value (line 8), first the individual bending, normal, and shear responses ($U_{bend,i,j}, U_{norm,i,j}$ and $U_{shear,i,j}$ respectively) are computed. If the total response $U_{tot,i,j}$ of this rectangle is less than some fraction η_{noise} of the mean response of the initial structural model $U_{tot,mean,0}$ (Equation 9.10), it may be assigned a new

type. This check aims to avoid type assignments based on numerical noise that may occur when the magnitude of the design response is small.

$$U_{tot,mean,0} = \frac{\sum_{j=0}^{n_{subs,0}} U_{tot,0,j}}{n_{subs,0}} \tag{9.10}$$

Next, line 11 loops over an ordered list c which holds a predetermined permutation of the set $\{1, 2, 3\}$ which corresponds to the types (1 for truss, 2 for beam, and 3 for flat shell), and indicates the order that decides which type is considered first for assignment. Each of the types (truss, beam, and flat shell) is assigned based on whether the ratio between the corresponding response (shear, normal, and bending respectively), and the total response exceeds a given type dependent threshold ($\eta_{shear}, \eta_{norm}, \eta_{bend}$). If a type is assigned, the algorithm continues after the for loop, otherwise it tries the next type in c.

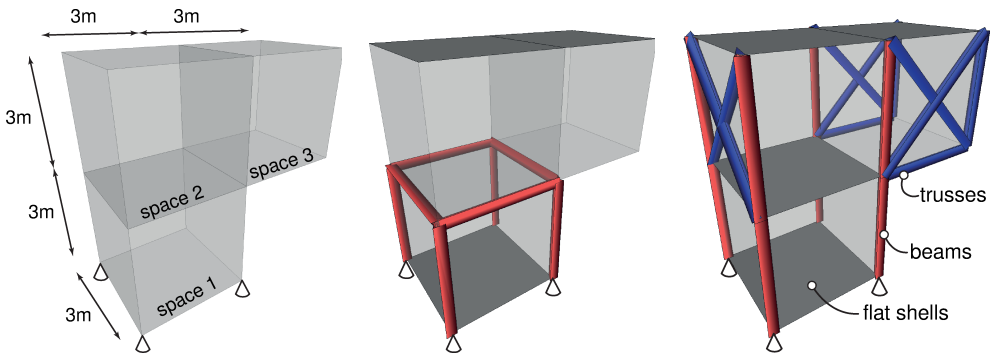
Finally, on line 21 the algorithm checks whether a rectangle still has the substitute type, in which case it is assigned the no structure type instead. This can happen if the noise threshold (η_{noise}) was not exceeded, or if none of the response thresholds ($\eta_{shear}, \eta_{norm}, \eta_{bend}$) were exceeded. After incrementing the loop counter (line 26), the next iteration starts.

An example of how the described design response grammar might generate a structural design for a given spatial design is shown in Figure 9.10. Starting from the substitute design in Figure 9.10a, beams are assigned to some rectangles in the intermediate design (Figure 9.10b), and after the final iteration the design in Figure 9.10c is produced. Note that for this example parameters of the grammar have been selected for illustrative purposes to show all structural types.

9.3.3 Case Study

To assess the performance of the design response grammar it is evaluated by generating a structural design for a given building spatial design. To find good parameter settings for the design response grammar 95 832 possible parameter configurations are considered. These configurations on average use three evaluations each (depending on η_{conv}), resulting in a total of 287 496 evaluations. In addition, the grammar is compared to structural assignments optimised by the MOMIES algorithm introduced in Chapter 8. Next, since the aim is to produce a grammar that can generate high quality structural designs at a low computational cost, parameter configurations are selected based on their performance and evaluated on a new design.

Two minimisation objectives are considered in the case study. Total strain energy



(a) Substitute. **(b)** Intermediate. **(c)** Final.

Figure 9.10: Example of the iterative process in which the design response grammar generates a structural design.

measured in Newton metres (N m), and structural volume measured in cubic metres (m^3). Strain energy is computed by finite element analysis, while the structural volume is computed as the sum of volumes of all components in the structural model.

To compute these two objectives the following settings are considered. A live load of $0.005\,\mathrm{N\,mm^{-2}}$ is considered. Further, wind loads from four directions (north, south, east, west) are considered with magnitudes of $0.001\,\mathrm{N\,mm^{-2}}$, $0.0004\,\mathrm{N\,mm^{-2}}$, and $0.0008\,\mathrm{N\,mm^{-2}}$ for pressure, shear, and suction respectively. Finally, each structural type has its specific settings as shown in Table 9.2.

type	thickness	width	height	cross sectional surface	Young's modulus	Poisson's ratio
flat shell	150 mm	–	–	–	$30\,000\,\mathrm{N\,mm^{-2}}$	0.3
beam	–	150 mm	150 mm	–	$30\,000\,\mathrm{N\,mm^{-2}}$	0.3
truss	–	–	–	$22\,500\,\mathrm{mm^2}$	$30\,000\,\mathrm{N\,mm^{-2}}$	–
substitute	150 mm	–	–	–	$0.03\,\mathrm{N\,mm^{-2}}$	0.3

Table 9.2: Settings used for each structural type.

As a test problem a structural design will be generated for the low rise building in Figure 9.11. This is a typical spatial design of a low rise building, and will as such also give insight into the practical applicability. Next, to evaluate the parameter configurations for the design grammar that will be selected based on the test results, the portal building shown in Figure 9.12 is considered.

For the test with the low rise building, full enumeration of the possible parameter combinations for the design response grammar is considered. The relevant parameters

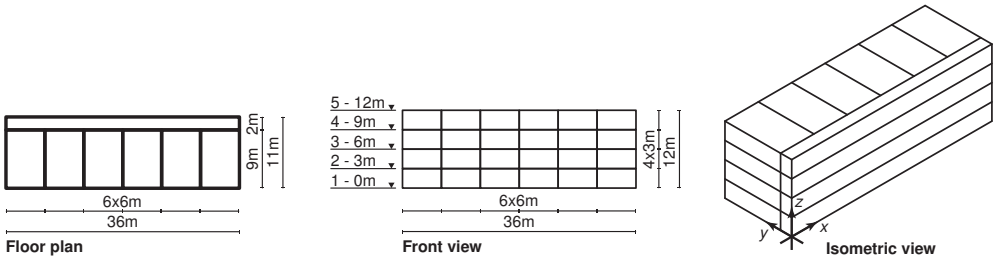


Figure 9.11: Spatial design of a low rise building with four floors, six spaces per floor, and a horizontal walkway on each floor.

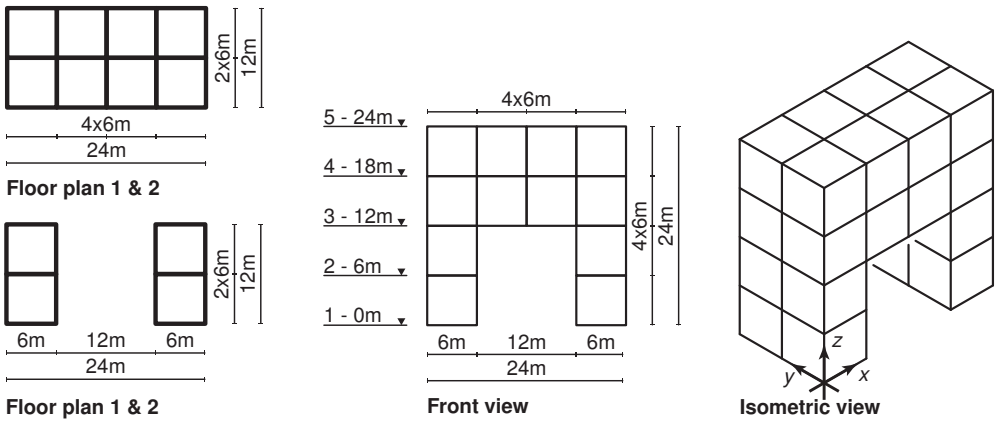


Figure 9.12: Spatial design of a portal building with four floors, 24 spaces, and a passage through its centre.

are shown in Table 9.3, and result in 95 832 possible parameter configurations. Based on their performance for the low rise building, three parameter configurations will be selected to be evaluated on the portal building to get a first indication about how well they generalise.

In order for the MOMIES algorithm to optimise the structural assignments in a structural design, the problem needs to be encoded in decision variables. The encoding used here is as follows. Each variable can take a value from the set $\{1, 2, 3, 4\}$. These numbers correspond to the different structural types, 1 for no structure, 2 for truss, 3 for beam, and 4 for flat shell. The low rise design consists of 168 of these variables, while the portal design has 104 decision variables.

MOMIES is used with a reference point of $(8 \times 10^{10}\,\mathrm{N\,mm}, 700\,\mathrm{m}^3)$ is for the low rise building, and $(1 \times 10^{10}\,\mathrm{N\,mm}, 300\,\mathrm{m}^3)$ for the portal building. These reference points also serve as the penalty value that is assigned as performance for designs that

description	parameter	values
shear threshold	η_{shear}	$[0.0, 0.1, \ldots, 1.0]$
bending threshold	η_{bend}	$[0.0, 0.1, \ldots, 1.0]$
normal threshold	η_{norm}	$[0.0, 0.1, \ldots, 1.0]$
noise threshold	η_{noise}	$\{0.025, 0.050, 0.075\}$
number of iterations	η_{conv}	$\{1, 2, 3, 4\}$
checking order	c	all permutations of $\{1, 2, 3\}$

Table 9.3: Design response grammar parameter ranges.

are structurally unstable. Further, MOMIES uses dominant crossover for the decision variables, and intermediate crossover for the step size. A single step size value for all variables is used, initialised to $1/n_d$, where n_d is the number of decision variables. Finally, a population size of $\mu = 50$ is used with an evaluation budget of 10 000, and five repetitions.

9.3.4 Results

Figure 9.13 shows the solutions discovered by MOMIES during the optimisation of the structural assignments. Some variance can be seen in the Pareto front approximations (PFAs) of the different runs, but all of them found high quality solutions. On the right side of the figure, examples of structural designs found by the algorithm are shown. Specifically, one design that performs well for each individual objective, a compromise solution, and a poor performing solution. All of these designs show – to the human eye – rather chaotic assignments of structural types. Evidently, homogeneous assignments are not needed to achieve good performance. However, this does not follow common practice. For practical use it may therefore be worth investigating whether optimisation of more consistent designs would still result in competitive designs.

The results for full enumeration of the possible parameter configurations for the design response grammar are shown in Figure 9.14. Moreover, they are compared to the PFA resulting from the combination of all runs by the MOMIES algorithm. Better solutions are found by the design response grammar in the knee point region, as well as both extremal regions. These areas are, however, only sparsely populated by the design response grammar. MOMIES on the other hand, has a wide selection of solutions in the knee point area, which may provide a decision maker with more freedom to choose. Moreover, for the design response grammar 287 496 evaluations were used, which is almost six times the 50 000 evaluations used by MOMIES. On the other hand, if the well performing parameter configurations generalise, this is a one

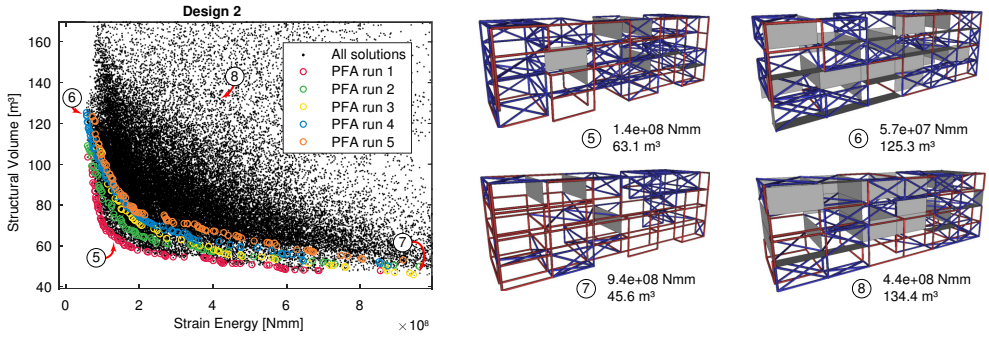


Figure 9.13: Optimisation results of a low rise building by MOMIES (left), and examples of corresponding structural designs (right).

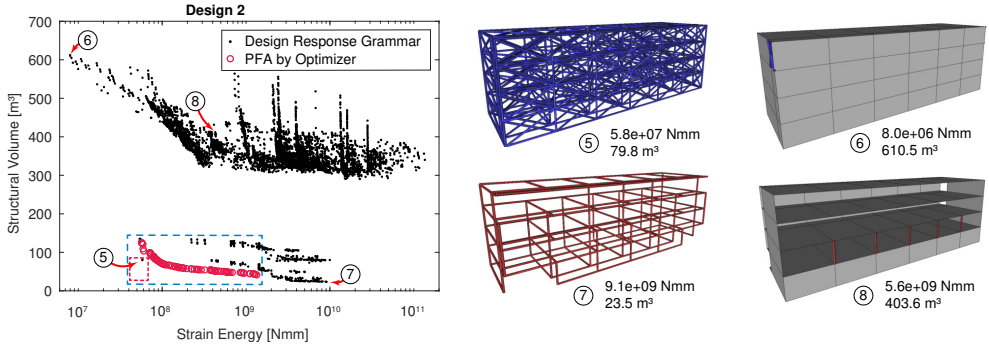


Figure 9.14: Results of the design response grammar for a low rise building compared to the PFA of MOMIES (left), and examples of corresponding structural designs (right).

time cost for the design response grammar, whereas MOMIES has to start from scratch for every new design. Even if MOMIES were to use fewer evaluations, say 10 000, that would still be far more expensive than the design response grammar which would only need a handful of evaluations. Another advantage of the design response grammar is the practicality of the produced structural designs. These designs are far more consistent than those produced by MOMIES, and thus more practical to construct.

Based on the results of the design response grammar, the parameter configurations in Table 9.4 were chosen for their good performance. Figure 9.15 visualises the results achieved by these parameter configurations, compared to the MOMIES algorithm. MOMIES clearly shows better performance, but the results of the design response grammar are competitive with the PFA. Most importantly, this shows that the design response grammar is indeed capable of finding high quality solutions for minimal com-

putational cost. Each of the parameter configurations of the design response grammar used only two evaluations, whereas MOMIES used 50 000.

In summary, it is concluded that the discrete component of the MOMIES algorithm introduced in Chapter 8 is effective in real world applications. Furthermore, with the design response grammar promising steps were made towards the goal of developing a grammar capable of producing high quality structural designs at a low cost.

parameter	η_{shear}	η_{bend}	η_{norm}	η_{noise}	η_{conv}	c
configuration 1	0.0	1.0	1.0	0.025	1	$(3, 2, 1)$
configuration 2	0.1	1.0	0.0	0.025	1	$(1, 2, 3)$
configuration 3	0.2	1.0	0.0	0.025	1	$(1, 2, 3)$

Table 9.4: Selected parameter configurations for the design response grammar.

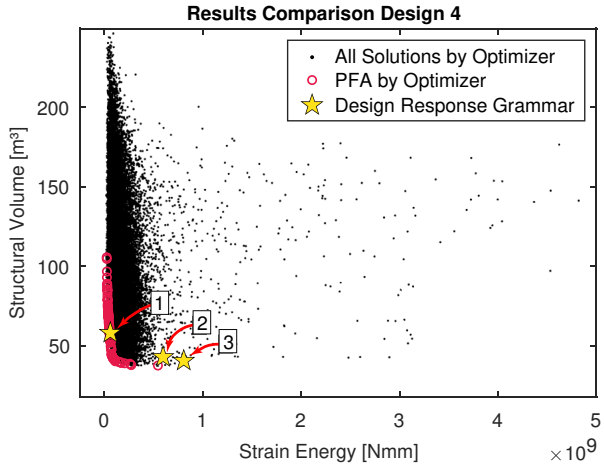


Figure 9.15: Comparison of results by the design response grammar and MOMIES on the portal design.

9.4 Conclusion

9.4.1 Summary

Over the course of this chapter applications of the algorithms developed in this thesis have been explored with the goal of answering RQ6. This question asks how the developed algorithms benefit applications in the real world design process. To this end,

the first section of this chapter investigated the interaction of the superstructure based SMS-EMOA-SC algorithm (Section 6.3.2) with a superstructure free co-evolutionary design process that more closely resembles the design practice. In the second section, interaction between an optimisation algorithm (again SMS-EMOA-SC) and a building information modelling environment was studied for practicality. Finally, the third section looked at the use of the generally applicable multi-objective mixed-integer evolution strategy (MOMIES, Chapter 8) in structural design optimisation.

The superstructure based SMS-EMOA-SC has been compared to superstructure free co-evolutionary design, as well as a relay hybrid method that combines the two. By using a superstructure free representation a more exploratory search that is natural to the design process becomes possible. On the other hand, a superstructure representation enables effective optimisation in interesting design regions. However, the restrictions of the superstructure prevent the discovery of other interesting design regions, that a free representation allows. The combination of these two aspects has been shown to be possible with a relay hybrid. Unfortunately, the modification rules of the co-evolutionary method had limited effect, and improved versions of those would likely result in a more useful hybrid method.

In line with the use of building information modelling (BIM) in the building design practice, the integration of optimisation and BIM has been investigated. A building spatial design has been transferred from a specific BIM environment (BouwConnect) to the supercube representation (Section 3.2) to be optimised by SMS-EMOA-SC. After optimisation, the spatial design was transferred back to the BIM environment. SMS-EMOA-SC proved to be effective at optimising its given objectives, but less so for specific (but related) regulations used in practice. Furthermore, the transfer of the building spatial design between the optimisation and BIM environments turned out to be laborious due to the need for manual adjustments.

To assess the use of the MOMIES algorithm introduced in Chapter 8 it has been applied to structural design optimisation. Here it functioned to set a baseline for an automated design grammar that has been developed with the aim of producing high quality structural assignments at a minimal cost. MOMIES proved to be an effective optimiser in the nominal discrete domain and found a good approximation of the knee point region of the Pareto front. At the same time, after full enumeration of its parameter configurations, high quality settings were found for the design response grammar. With these settings it was evaluated on a new design case, and shown to find solutions near the knee point region discovered by MOMIES, while using only a few evaluations.

9.4.2 Future Work

A successful first step was made in the hybrid use of the superstructure and superstructure free representations in optimisation. However, various new research directions also presented themselves. To effectively explore various interesting design regions, new modification rules have to be developed for the co-evolutionary design method. SMS-EMOA-SC, in addition, would benefit from better diversity handling. This would allow it to include an initial design in its search, without being biased towards it to the extend that it does not explore other options sufficiently.

Improvements to the interaction between optimisation tools and BIM environments are also necessary. SMS-EMOA-SC proved itself as an effective optimiser, but unless the considered objectives closely match the regulations used in practice, it remains of limited use. Furthermore, the transfer of designs between BIM and optimisation environments has to be automated. Currently this involves too many manual design decisions, while these could be automated or guided based on similar existing designs. Many of these improvements may seem practical in nature, but will require strong interaction with practitioners to get right.

The design response grammar is already effective at finding high quality solutions with a minimal number of evaluations, but it may be possible to find even better parameter configurations for it. For instance, a number of parameters that are continuous in practice were discretised to make full enumeration of the parameter configurations possible. If these parameters are kept as continuous values, an optimiser could be applied to handle the larger number of options, and in turn find better solutions.

With regard to the used MOMIES algorithm, it proved effective in optimising in the nominal discrete domain for a real world problem. However, it should still be evaluated more extensively on continuous, integer, and mixed problems.

Chapter 10

Conclusions

This thesis investigated many different aspects of multi-objective mixed-integer optimisation problems related to the built environment. In this conclusion, first the contributions of each chapter are briefly summarised, together with answers to their corresponding research questions. After that, paths to future discoveries are discussed that opened up as a result of this work.

10.1 Summary

Chapter 1 gave a high level introduction to the topic of study in this thesis.

Chapter 2 then provided a starting point to this study by introducing key concepts used throughout the thesis. Specifically, it covered the basics of optimisation for a single objective, as well as for multiple objectives. Evolutionary computation was introduced next as an approach to solve such optimisation problems. Finally, it covered the main application domain of this work: building spatial design.

Chapter 3 introduced a representation for building spatial designs in response to the first research question:

RQ1 How can elements of the solution space be represented?

The introduction of the supercube representation makes it possible to encode an arbitrary number of spaces, and to control their dimensions. To be able to check

the feasibility of building spatial designs, a number of constraints on the representation were introduced as well. These constraints are polynomial expressions directly formulated on the binary and continuous decision variables. Hence they can be exactly computed in a straightforward manner, and, at least in principle, can be used in equation based solvers.

Chapters 4 and 5 looked at how to manage the constraints used in the supercube in order to answer the second research question:

> *RQ2 How can the discovery of feasible designs be ensured?*

Penalty functions that act on constraint violating solutions are shown to aid the search process in Chapter 4. They are, however, also found not to scale sufficiently for larger supercube sizes. Chapter 5 presents problem specific initialisation and mutation operators that navigate only the feasible space, and thus do not suffer from the larger infeasible regions present in larger sized supercubes.

Chapter 6 investigated the benefits of a local search strategy with the third research question in mind:

> *RQ3 How can local search contribute to the improvement of solutions found during global search in a multi-objective setting?*

Hypervolume indicator gradient ascent multi-objective optimisation (HIGA-MO) is shown to work for a real world problem, and with numerically approximated gradients. Even so, the number of evaluations needed for numerical approximation of the considered high dimensional problem, prevents it from outperforming the considered evolutionary algorithm (SMS-EMOA-SC). This means that as a local search component HIGA-MO would likely outperform SMS-EMOA-SC at some point due to the convergence properties of the hypervolume gradient, but the number of evaluations needed would be impractical.

Chapter 7 evaluated the value of the data that is generated during optimisation based on the fourth research question:

> *RQ4 What can be learned about building spatial design from the optimisation process?*

Based on optimisation data, key properties of high quality building spatial designs can be learned, by automatically extracting design rules from the data. These design rules can then be communicated to design experts to prove to them that the discovered solutions are trustworthy, since they typically correspond to well-known design rules used by experts.

Chapter 8 explored the possibilities of a general mixed-integer algorithm for multi-objective optimisation to answer the fifth research question:

> *RQ5 How can a generally applicable multi-objective mixed-integer algorithm be developed?*

Promising first steps are made towards a general purpose multi-objective mixed-integer evolution strategy (MOMIES). MOMIES is shown to perform well in practice, and recombination has been discovered to be more valuable in multi-objective optimisation than previously believed, although the reasons are still unclear. Step size adaptation for the multi-objective case requires more thought, as the current methods still show erratic behaviour and are not capable of reliably tracking optimal mutation step sizes.

Chapter 9 presented various applications of the developed algorithms in response to the sixth research question:

> *RQ6 How applicable are the developed algorithms to real world problems?*

In application, SMS-EMOA-SC proved to be a useful component in a hybrid algorithm that also explored which supercube configurations were most interesting to work with. Furthermore, SMS-EMOA-SC was shown to be a good optimiser, but the settings and objectives considered in the earlier academic cases do not always align with the practicalities of the real world. In the nominal discrete space of structural design optimisation MOMIES proved to be a highly effective algorithm.

In summary, this thesis has made tools available for the multi-objective optimisation of building spatial designs. To this end a representation has been introduced, as well as specialised operators for the building spatial design problem. Furthermore, the potential benefits of a memetic algorithm, where stochastic global search is combined with deterministic local search, has been explored. Practical utilisation of these methods has also been studied. Both by analysis of the optimisation data to help human designers, as well as through real world studies of the algorithms. Finally, a

general algorithm, not restricted to the building spatial design problem, has also been developed.

10.2 Future Work

Although considerable progress has been made in the context of this thesis, much more work remains. Some perspectives for this future research are included here, and are briefly discussed.

New or modified representations that contain fewer infeasible or duplicate solutions than the supercube representation would be very helpful in tackling larger building designs, and in speeding up the search process in general. Chapter 3 already suggested a number of possible directions, such as the use of an integer representation instead of the current binary one. However, as also mentioned before, caution is needed when considering what is or is not a duplicate since this is largely problem dependent. With a new representation it is also likely that new operators are needed. As such, representations and operators should be considered in concert to avoid poorly compatible combinations.

Local search with the hypervolume indicator (HVI) gradient successfully improved the Pareto front approximation found during global search. It was, however, not able to outperform SMS-EMOA-SC. The primary shortcoming is found in the many evaluations needed to approximate the gradient for the considered high dimensional problem. To overcome this, methods are needed that reduce the computational costs of an algorithm using the HVI gradient. One interesting direction is the use stochastic gradient descent methods, which is currently used with success in the optimisation of weights in deep neural networks . These networks, like building spatial designs (in the supercube representation), require the adaptation of a large number of continuous variables for a given discrete structure. Another option would be the use of surrogate models that can be learned from previously obtained evaluation data. These would both allow cheap numerical approximation of the HVI gradient on the model, rather than the true function, and enable the navigation of discrete space. Another way to reduce computation costs is to move fewer points based on the HVI gradient. Only nondominated points could be considered, rather than all points, for instance.

Since differences in performance are larger between discrete subspaces than within them, median attainment curves struggle to adequately present the performance differences for algorithms that sometimes end up in one discrete subspace, and sometimes in another. New multi-objective performance measures for mixed-integer problems that

take these things into account are thus a must.

Integrating the insights that can be gained from optimisation data into the optimisation process has the potential to improve, and possibly speed up the optimisation process. These insights could be used to only perform expensive evaluations for candidate solutions that look at least somewhat promising, instead of for all candidates. Another new direction in this area could be the use of learned design qualities in guiding the variation operators. By biasing the operators to directions that are likely to produce improved designs, the search could be speeded up. Naturally, the risk is that unexpected design alternatives will not be found. How to balance this bias with the needed exploratory components will therefore require careful consideration.

Finally, another promising path for future work is continue the development and analysis of MOMIES, the general purpose multi-objective mixed-integer optimisation technique that resulted from this thesis. To be a full fledged general optimiser, MOMIES will also have to include a constraint handling component capable of operating in the multi-objective mixed-integer environment. In addition, step size adaptation has to be tailored for multi-objective optimisation to improve its effectiveness. This is particularly true for the integer and nominal discrete adaptation mechanisms, since they were shown to have particularly irregular behaviour. Recombination was shown to be surprisingly effective for the multi-objective case, but it remains unclear why. Investigating this is especially interesting since it was previously shown that the value of recombination can be questioned in the multi-objective setting [101]. A deeper understanding could elucidate when it is of value to use recombination, and when to avoid it.

Bibliography

[1] J. A. Aguilera, J. Aguilera, P. Baumeister, A. Bloom, D. Coursen, J. A. Dobrowolski, F. T. Goldstein, D. E. Gustafson, and R. A. Kemp. Antireflection coatings for germanium ir optics: a comparison of numerical design methods. *Applied Optics*, 27(14):2832–2840, 1988.

[2] Karsten Ahnert and Mario Mulansky. Odeint - solving ordinary differential equations in C++. *AIP Conference Proceedings*, 1389(1):1586–1589, 2011.

[3] Shady Attia, Elisabeth Gratia, André De Herde, and Jan L. M. Hensen. Simulation-based decision support tool for early stages of zero-energy building design. *Energy and Buildings*, 49:2–15, 2012.

[4] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz. *Handbook of Evolutionary Computation*. IOP Publishing Ltd., 1997.

[5] Thomas Bäck and Martin Schütz. Evolution strategies for mixed-integer optimization of optical multilayer systems. In *Evolutionary Programming*, pages 33–51, 1995.

[6] Robert Baldock and Kristina Shae. Structural topology optimization of braced steel frameworks using genetic programming. In Ian F. C. Smith, editor, *Intelligent Computing in Engineering and Architecture: 13th EG-ICE Workshop 2006, Ascona, Switzerland, June 25-30, 2006, Revised Selected Papers*, pages 54–61, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[7] Sunith Bandaru and Kalyanmoy Deb. Automated innovization for simultaneous discovery of multiple rules in bi-objective problems. In Ricardo H. C. Takahashi, Kalyanmoy Deb, Elizabeth F. Wanner, and Salvatore Greco, editors, *Evolutionary Multi-Criterion Optimization*, pages 1–15, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[8] Sunith Bandaru and Kalyanmoy Deb. Temporal innovization: Evolution of design principles using multi-objective optimization. In António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos Coello Coello, editors, *Evolutionary Multi-Criterion Optimization*, pages 79–93, Cham, 2015. Springer International Publishing.

[9] Thomas Bartz-Beielstein, Christian W. G. Lasarczyk, and Mike Preuß. Sequential parameter optimization. In *2005 IEEE congress on evolutionary computation*, volume 1, pages 773–780. IEEE, 2005.

[10] Jean-Louis Batoz and Mabrouk Ben Tahar. Evaluation of a new quadrilateral thin plate bending element. *International Journal for Numerical Methods in Engineering*, 18(11):1655–1677, 1982.

[11] A. D. Belegundu and S. D. Rajan. A shape optimization approach based on natural design variables and shape functions. *Computer Methods in Applied Mechanics and Engineering*, 66(1):87–106, 1988.

[12] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies–a comprehensive introduction. *Natural computing*, 1(1):3–52, 2002.

[13] R. Blok and N. Nieuwenhuizen. *Tabellen voor bouw-en waterbouwkundigen*. Thieme Meulenhof BV, 2006.

[14] Koen van der Blom and Thomas Bäck. A new foraging-based algorithm for online scheduling. In *GECCO '18: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 53–60, New York, NY, USA, 2018. ACM.

[15] Koen van der Blom, Sjonnie Boonstra, Hèrm Hofmeyer, Thomas Bäck, and Michael T. M. Emmerich. Configuring advanced evolutionary algorithms for multicriteria building spatial design optimisation. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1803–1810. IEEE, 2017.

[16] Koen van der Blom, Sjonnie Boonstra, Hèrm Hofmeyer, and Michael T. M. Emmerich. Multicriteria building spatial design with mixed integer evolutionary algorithms. In Julia Handl, Emma Hart, Peter R. Lewis, Manuel López-Ibáñez, Gabriela Ochoa, and Ben Paechter, editors, *Parallel Problem Solving from Nature – PPSN XIV*, volume 9921 of *Lecture Notes in Computer Science*, pages 453–462, Cham, 2016. Springer International Publishing.

[17] Koen van der Blom, Sjonnie Boonstra, Hèrm Hofmeyer, and Michael T. M. Emmerich. A super-structure based optimisation approach for building spatial designs. In M. Papadrakakis, V. Papadopoulos, G. Stefanou, and V. Plevris, editors, *VII European Congress on Computational Methods in Applied Sciences and Engineering – ECCOMAS VII*, volume 2, pages 3409–3422, Athens, Greece, 2016. National Technical University of Athens.

[18] Koen van der Blom, Sjonnie Boonstra, Hèrm Hofmeyer, and Michael T. M. Emmerich. Analysing optimisation data for multicriteria building spatial design. In Kalyanmoy Deb, Erik Goodman, Carlos A. Coello Coello, Kathrin Klamroth, Kaisa Miettinen, Sanaz Mostaghim, and Patrick Reed, editors, *Evolutionary Multi-Criterion Optimization*, pages 671–682, Cham, 2019. Springer International Publishing.

[19] Koen van der Blom, Sjonnie Boonstra, Hao Wang, Hèrm Hofmeyer, and Michael T. M. Emmerich. Evaluating memetic building spatial design optimisation using hypervolume indicator gradient ascent. In Leonardo Trujillo, Oliver Schütze, Yazmin Maldonado, and Paul Valle, editors, *Numerical and Evolutionary Optimization – NEO 2017*, pages 62–86. Springer, Cham, 2018.

[20] Koen van der Blom, Kaifeng Yang, Thomas Bäck, and Michael T. M. Emmerich. Towards multi-objective mixed integer evolution strategies. In Michael T. M. Emmerich, André H. Deutz, Sander C. Hille, and Yaroslav D. Sergeyev, editors, *Proceedings LeGO – 14th International Global Optimization Workshop*, volume 2070, pages 020046–1–020046–4. AIP Publishing, 2019.

[21] Sjonnie Boonstra, Koen van der Blom, Hèrm Hofmeyer, Robert Amor, and Michael T. M. Emmerich. Super-structure and super-structure free design search space representations for a building spatial design in multi-disciplinary building optimisation. In *Electronic proceedings of the 23rd International Workshop of the European Group for Intelligent Computing in Engineering*, pages 1–10. Jagiellonian University ZPGK, 2016.

[22] Sjonnie Boonstra, Koen van der Blom, Hèrm Hofmeyer, Joost van den Buijs, and Michael T. M. Emmerich. Coupling between a building spatial design optimisation toolbox and bouwconnect BIM. In *35th CIB W78 2018 Conference: IT in Design, Construction, and Management*, pages 95–102. Springer, 2018.

[23] Sjonnie Boonstra, Koen van der Blom, Hèrm Hofmeyer, and Michael T. M. Emmerich. Combined super-structured and super-structure free optimisation of building spatial designs. In C. Koch, W. Tizani, and J. Ninić, editors, *24rd International Workshop of the European Group for Intelligent Computing in Engineering*, pages 23–34. University of Nottingham, 2017.

[24] Sjonnie Boonstra, Koen van der Blom, Hèrm Hofmeyer, and Michael T. M. Emmerich. Conceptual structural system layouts via design response grammars and evolutionary algorithms. *Automation in Construction*, pages 1–24, 2019. (submitted).

[25] Sjonnie Boonstra, Koen van der Blom, Hèrm Hofmeyer, Michael T. M. Emmerich, Jos van Schijndel, and Pieter de Wilde. Toolbox for super-structured and super-structure free multi-disciplinary building spatial design optimisation. *Advanced Engineering Informatics*, 36:86–100, 2018.

[26] Sjonnie Boonstra, Koen van der Blom, Hérm Hofmeyer, and Michael T. M. Emmerich. Co-evolutionary design processes applied to building spatial design optimization. In *Advances in Structural and Multidisciplinary Optimization. Proceedings of the 13th World Congress of Structural and Multidisciplinary Optimization (WCSMO13)*, pages 1–6. Springer, 2020. (in print).

[27] Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[28] Karl Bringmann, Tobias Friedrich, and Patrick Klitzke. Generic postprocessing via subset selection for hypervolume and epsilon-indicator. In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipič, and Jim Smith, editors, *Parallel Problem Solving from Nature – PPSN XIII*, pages 518–527, Cham, 2014. Springer International Publishing.

[29] A. L. Custódio, M. Emmerich, and J. F. A. Madeira. Recent developments in derivative-free multiobjective optimization. *Computational Technology Reviews*, 5(1):1–31, 2012.

[30] Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148, 1995.

[31] Kalyanmoy Deb, Sunith Bandaru, David Greiner, António Gaspar-Cunha, and Cem Celal Tutum. An integrated approach to automated innovization for discovering useful design principles: Case studies from engineering. *Applied Soft Computing*, 15:42–56, 2014.

[32] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[33] Kalyanmoy Deb and Aravind Srinivasan. Innovization: Innovating design principles through optimization. In *GECCO '06 Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1629–1636, New York, NY, USA, 2006. ACM.

[34] M. Dellnitz, O. Schütze, and T. Hestermeyer. Covering pareto sets by multilevel subdivision techniques. *Journal of Optimization Theory and Applications*, 124(1):113–136, Jan 2005.

[35] W. B. Dolan, P. T. Cummings, and M. D. Le Van. Algorithmic efficiency of simulated annealing for heat exchanger network design. *Computers & Chemical Engineering*, 14(10):1039–1050, 1990.

[36] Stefan Droste and Dirk Wiesmann. Metric based evolutionary algorithms. In Riccardo Poli, Wolfgang Banzhaf, William B. Langdon, Julian Miller, Peter Nordin, and Terence C. Fogarty, editors, *Genetic Programming*, pages 29–43, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[37] Chuck Eastman, Paul Teicholz, Rafael Sacks, and Kathleen Liston. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. Wiley & Sons, 2011.

[38] Matthias Ehrgott. *Multicriteria Optimization*, volume 491. Springer Science & Business Media, 2005.

[39] Michael Emmerich, Nicola Beume, and Boris Naujoks. An EMO algorithm using the hypervolume measure as selection criterion. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, pages 62–76, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[40] Michael Emmerich and André Deutz. Time complexity and zeros of the hypervolume indicator gradient field. In Oliver Schuetze, Carlos A. Coello Coello, Alexandru-Adrian Tantar, Emilia Tantar, Pascal Bouvry, Pierre Del Moral, and Pierrick Legrand, editors, *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation III*, pages 169–193, Heidelberg, 2014. Springer International Publishing.

[41] Michael Emmerich, André Deutz, and Nicola Beume. Gradient-based / evolutionary relay hybrid for computing pareto front approximations maximizing the S-metric. In Thomas Bartz-Beielstein, María José Blesa Aguilera, Christian Blum, Boris Naujoks, Andrea Roli, Günter Rudolph, and Michael Sampels, editors, *Hybrid Metaheuristics*, pages 140–156, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[42] Michael Emmerich, Monika Grötzner, Bernd Groß, and Martin Schütz. Mixed-integer evolution strategy for chemical plant optimization with simulators. In I. C. Parmee, editor, *Evolutionary Design and Manufacture: Selected Papers from ACDM '00*, pages 55–67, London, 2000. Springer London.

[43] Michael Emmerich, Monika Grötzner, and Martin Schütz. Design of graph-based evolutionary algorithms: A case study for chemical process networks. *Evolutionary Computation*, 9(3):329–354, 2001.

[44] Michael T. M. Emmerich and André H. Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural Computing*, 17(3):585–609, Sep 2018.

[45] European Commission. *Challenging and Changing Europe's Built Environment: A Vision for a Sustainable and Competitive Construction Sector By 2030*. European Construction Technology Platform, 2005.

[46] Jörg Fliege and Benar Fux Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3):479–494, Aug 2000.

[47] Christodoulos A. Floudas. *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. Oxford University Press, 1995.

[48] Lawrence J. Fogel. Autonomous automata. *Industrial Research*, 4:14–19, 1962.

[49] Carlos M. Fonseca, Viviane Grunert da Fonseca, and Luís Paquete. Exploring the performance of stochastic multiobjective optimisers with the second-order

attainment function. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, pages 250–264, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[50] Guennebaud, G. and Jacob, B. and others. Eigen v3: a C++ linear algebra library, 2010. [Online; accessed 7-May-2018].

[51] Victor Adrián Sosa Hernández, Oliver Schütze, and Michael Emmerich. Hypervolume maximization via set based newton's method. In Alexandru-Adrian Tantar, Emilia Tantar, Jian-Qiao Sun, Wei Zhang, Qian Ding, Oliver Schütze, Michael Emmerich, Pierrick Legrand, Pierre Del Moral, and Carlos A. Coello Coello, editors, *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*, pages 15–28, Cham, 2014. Springer International Publishing.

[52] Hèrm Hofmeyer and Juan Manuel Davila Delgado. Coevolutionary and genetic algorithm based building spatial and structural design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 29(04):351–370, 2015.

[53] John H. Holland. Outline for a logical theory of adaptive systems. *Journal of the Association for Computing Machinery*, 9(3):297–314, 1962.

[54] Christina J. Hopfe, Michael T. M. Emmerich, Robert Marijt, and Jan L. M. Hensen. Robust multi-criteria design optimisation in building design. *Proceedings of Building Simulation and Optimization, Loughborough, UK*, pages 118–125, 2012.

[55] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[56] Roy Jackson. Optimization of chemical reactors with respect to flow configuration. *Journal of Optimization Theory and Applications*, 2(4):240–259, Jul 1968.

[57] Terry Jones and Stephanie Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *ICGA*, volume 95, pages 184–192, 1995.

[58] Lachmi Khemlani, Anne Timerman, Beatrice Bennen, and Yehuda E. Kalay. Intelligent representation for computer-aided building design. *Automation in Construction*, 8(1):49 – 71, 1998.

[59] Rafal Kicinger, Tomasz Arciszewski, and Kenneth De Jong. Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers & Structures*, 83(23):1943–1978, 2005.

[60] Koningklijk Nederlands Metereologisch Instituut. Jaar 2014 – uitzonderlijk warm, zeer zonnig, en vrij droog, 2015. [Online; accessed 18-February-2019].

[61] Koningklijk Nederlands Metereologisch Instituut. Daggegevens van het weer in Nederland, 2018. [Online; accessed 7-May-2018].

[62] John R. Koza, David Andre, Forrest H. Bennett, III, and Martin A. Keane. Use of automatically defined functions and architecture-altering operations in automated circuit synthesis with genetic programming. In *Proceedings of the 1st Annual Conference on Genetic Programming*, pages 132–140, Cambridge, MA, USA, 1996. MIT Press.

[63] Rick Kramer, Jos van Schijndel, and Henk Schellen. Simplified thermal and hygric building models: A literature review. *Frontiers of Architectural Research*, 1(4):318–325, 2012.

[64] Tobias Kuhn, Carlos M. Fonseca, Luís Paquete, Stefan Ruzika, Miguel M. Duarte, and José Rui Figueira. Hypervolume subset selection in two dimensions: Formulations and algorithms. *Evolutionary Computation*, 24(3):411–425, 2016. PMID: 26135717.

[65] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *J. ACM*, 22(4):469–476, October 1975.

[66] Adriana Lara, Gustavo Sanchez, Carlos A. Coello Coello, and Oliver Schütze. HCS: A new local search strategy for memetic multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 14(1):112–132, Feb 2010.

[67] David Laredo Razo. EDS: A continuation method for mixed-integer multiobjective optimization problems. Master's thesis, CINVESTAV-IPN, Mexico City, 2015.

[68] Rui Li, M. T. M. Emmerich, J. Eggermont, E. G. P. Bovenkamp, T. Bäck, J. Dijkstra, and J. H. C. Reiber. Metamodel-assisted mixed integer evolution strategies and their application to intravascular ultrasound image analysis. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 2764–2771, June 2008.

[69] Rui Li, Michael T. M. Emmerich, Jeroen Eggermont, Thomas Bäck, Martin Schütz, Jouke Dijkstra, and Johan H. C. Reiber. Mixed integer evolution strategies for parameter optimization. *Evolutionary computation*, 21(1):29–64, 2013.

[70] Xingtao Liao, Qing Li, Xujing Yang, Weigang Zhang, and Wei Li. Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural and Multidisciplinary Optimization*, 35(6):561–569, 2008.

[71] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[72] David J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge university press, 2003.

[73] Mary Lou Maher and Hsien-Hui Tang. Co-evolution as a computational and cognitive model of design. *Research in Engineering Design*, 14(1):47–64, Feb 2003.

[74] Adanay Martín and Oliver Schütze. Pareto tracer: a predictor-corrector method for multi-objective optimization problems. *Engineering Optimization*, 50(3):516–536, 2018.

[75] Joaquim R. R. A. Martins and Andrew B. Lambe. Multidisciplinary design optimization: A survey of architectures. *AIAA journal*, 51(9):2049–2075, 2013.

[76] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 829–836, New York, NY, USA, 2011. ACM.

[77] Pablo Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program 158-79, Technical Report*, pages 1–68, 1989.

[78] Amos H. C. Ng, Catarina Dudas, Henrik Boström, and Kalyanmoy Deb. Interleaving innovization with evolutionary multi-objective optimization in production system simulation for faster convergence. In Giuseppe Nicosia and Panos Pardalos, editors, *Learning and Intelligent Optimization*, pages 1–18, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[79] Matti Palonen, Mohamed Hamdy, and Ala Hasan. MOBO a new software for multi-objective building performance optimization. In *Proceedings of the 13th Internationcal Conference of the IBPSA*, pages 2567–2574, 2013.

[80] Erik Pitzer and Michael Affenzeller. A comprehensive survey on fitness landscape analysis. In János Fodor, Ryszard Klempous, and Carmen Paz Suárez Araujo, editors, *Recent Advances in Intelligent Engineering Systems*, pages 161–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[81] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog Verlag, Stuttgart, 1973.

[82] Ingo Rechenberg. *Evolutionsstrategie '94, Werkstatt Bionik und Evolutionstechnik Band 1*. Frommann Holzboog, Stuttgart, 1994.

[83] Helge Rosé, Werner Ebeling, and Torsten Asselmeyer. The density of states — a measure of the difficulty of optimisation problems. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature — PPSN IV*, volume 1141 of *Lecture Notes in*

Computer Science, pages 208–217, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[84] G. Sand, J. Till, T. Tometzki, M. Urselmann, S. Engell, and M. Emmerich. Engineered versus standard evolutionary algorithms: A case study in batch scheduling with recourse. *Computers & Chemical Engineering*, 32(11):2706 – 2722, 2008. Enterprise-Wide Optimization.

[85] S. Schäffler, R. Schultz, and K. Weinzierl. Stochastic method for the solution of unconstrained vector optimization problems. *Journal of Optimization Theory and Applications*, 114(1):209–222, Jul 2002.

[86] Oliver Schütze, Carlos A. Coello Coello, Sanaz Mostaghim, El-Ghazali Talbi, and Michael Dellnitz. Hybridizing evolutionary strategies with continuation methods for solving multi-objective problems. *Engineering Optimization*, 40(5):383–402, 2008.

[87] Oliver Schütze, Víctor Adrián Sosa Hernández, Heike Trautmann, and Günter Rudolph. The hypervolume based directed search method for multi-objective optimization problems. *Journal of Heuristics*, 22(3):273–300, Jun 2016.

[88] Hans-Paul Schwefel. *Evolutionsstrategie und numerische Optimierung*. PhD thesis, Technische Universität Berlin, 1975.

[89] Hans-Paul Schwefel. Numerische optimierung von computer-modellen mittels der evolutionsstrategie. *Interdisciplinary Systems Research*, 26:319–354, 1977.

[90] Hans-Paul Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA, 1981.

[91] Hans-Paul Schwefel. *Evolution and Optimum Seeking*. John Wiley & Sons, Inc., New York, NY, USA, 1993.

[92] Zbigniew Sekulski. Least-weight topology and size optimization of high speed vehicle-passenger catamaran structure by genetic algorithm. *Marine Structures*, 22(4):691–711, 2009.

[93] Ofer M. Shir, Michael T. M. Emmerich, and Thomas Bäck. Adaptive niche radii and niche shapes approaches for niching with the CMA-ES. *Evolutionary Computation*, 18(1):97–126, 2010.

[94] E.-G. Talbi. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5):541–564, Sep 2002.

[95] Terry Therneau and Beth Atkinson. *rpart: Recursive Partitioning and Regression Trees*, 2018. R package version 4.1-13.

[96] Philip Voll, Matthias Lampe, Gregor Wrobel, and André Bardow. Superstructure-free synthesis and optimization of distributed industrial energy

supply systems. *Energy*, 45(1):424–435, 2012. The 24th International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy, ECOS 2011.

[97] Hao Wang, André Deutz, Thomas Bäck, and Michael Emmerich. Hypervolume indicator gradient ascent multi-objective optimization. In Heike Trautmann, Günter Rudolph, Kathrin Klamroth, Oliver Schütze, Margaret Wiecek, Yaochu Jin, and Christian Grimme, editors, *Evolutionary Multi-Criterion Optimization*, pages 654–669, Cham, 2017. Springer International Publishing.

[98] Hao Wang, Yiyi Ren, André Deutz, and Michael Emmerich. On steering dominated points in hypervolume indicator gradient ascent for bi-objective optimization. In Oliver Schütze, Leonardo Trujillo, Pierrick Legrand, and Yazmin Maldonado, editors, *NEO 2015: Results of the Numerical and Evolutionary Optimization Workshop NEO 2015 held at September 23-25 2015 in Tijuana, Mexico*, pages 175–203, Cham, 2017. Springer International Publishing.

[99] Honggang Wang. Direct zigzag search for discrete multi-objective optimization. *Computers & Operations Research*, 61:100 – 109, 2015.

[100] Honggang Wang, David Laredo, Oliver Cuate, and Oliver Schütze. Enhanced directed search: a continuation method for mixed-integer multi-objective optimization problems. *Annals of Operations Research*, Sep 2018.

[101] Simon Wessing, Rosa Pink, Kai Brandenbusch, and Günter Rudolph. Toward step-size adaptation in evolutionary multiobjective optimization. In Heike Trautmann, Günter Rudolph, Kathrin Klamroth, Oliver Schütze, Margaret Wiecek, Yaochu Jin, and Christian Grimme, editors, *Evolutionary Multi-Criterion Optimization*, pages 670–684, Cham, 2017. Springer International Publishing.

[102] Michael Wetter and Elijah Polak. Building design optimization using a convergent pattern search algorithm with adaptive precision simulations. *Energy and Buildings*, 37(6):603 – 612, 2005.

[103] Dirk Wiesmann. From syntactical to semantical mutation operators for structure optimization. In Juan Julián Merelo Guervós, Panagiotis Adamidis, Hans-Georg Beyer, Hans-Paul Schwefel, and José-Luis Fernández-Villacañas, editors, *Parallel Problem Solving from Nature — PPSN VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 234–243, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[104] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[105] Kaifeng Yang, Koen van der Blom, Thomas Bäck, and Michael Emmerich. Towards single- and multiobjective bayesian global optimization for mixed integer problems. In Michael T. M. Emmerich, André H. Deutz, Sander C. Hille, and

Yaroslav D. Sergeyev, editors, *Proceedings LeGO – 14th International Global Optimization Workshop*, volume 2070, pages 020044–1–020044–4. AIP Publishing, 2019.

[106] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.

[107] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms — a comparative case study. In Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature — PPSN V*, pages 292–301, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

Samenvatting

In dit proefschrift (getiteld: Multi-Criteria Gemengd-Geheeltallige Evolutionaire Algoritmen voor Gebouw Ruimtelijk Ontwerp) is onderzocht hoe evolutionaire algoritmen toegepast kunnen worden op complexe problemen. Deze complexiteit komt voort uit meerdere aspecten. Allereerst gaat het om een multi-criteria probleem. Dit betekent dat er oplossingen gevonden moeten worden die een afweging maken tussen meerdere verschillende eisen. Ten tweede is de representatie van dit probleem gemengd-geheeltallig. Dit introduceert complexiteit omdat het doorzoeken van reële getallen efficiënt gedaan wordt met andere technieken dan het doorzoeken van discrete getallen. Oftewel, meerdere technieken moeten gecombineerd worden. Het derde aspect is het omgaan met restricties. Zulke restricties zijn er om te garanderen dat gevonden oplossingen valide zijn.

Al deze aspecten worden meegenomen in het oplossen van een praktisch probleem: De optimalisatie van gebouw-ruimtelijke ontwerpen in een vroeg ontwerpstadium. Dit is van belang omdat het aanpassen van het ontwerp in latere fases veel werk kost, er moeten dan namelijk meerdere aspecten opnieuw worden bekeken. Door vroeg in het ontwerpproces te optimaliseren kan er dus tijd bespaard worden. Daarnaast leidt de optimalisatie ook tot betere prestaties. In deze scriptie worden prestaties op het gebied van temperatuur (koeling en verwarming), evenals structurele aspecten meegenomen.

Om te beginnen moet er een representatie voor de mogelijke oplossingen ontwikkeld worden voor het ruimtelijke ontwerp (Hoofdstuk 3). Deze zogeheten *supercube* representatie maakt het mogelijk om een arbitrair aantal ruimtes te beschrijven, en de dimensies daarvan in te stellen. Om de validiteit van de ruimtelijke ontwerpen te kunnen controleren zijn er ook een aantal restricties voor de representatie geïntroduceerd. Deze restricties zijn polynomiale expressies die op directe basis van de binaire en reële variabelen geformuleerd zijn. Om deze reden kunnen ze op eenvoudige wijze precies berekend worden en daardoor, in principe, gebruikt worden in op vergelijkingen

gebaseerde oplossers.

Vervolgens is de vraag hoe er met de restricties op de supercube omgegaan kan worden, om het vinden van valide ontwerpen te garanderen. Hoofdstuk 4 laat zien dat het gebruik van straffuncties die reageren op het overschrijden van de restricties kan helpen in het zoekproces. Helaas wordt er ook aangetoond dat ze niet voldoende schalen naar grotere supercube afmetingen. In Hoofdstuk 5 worden probleemspecifieke initialisatie- en mutatieoperatoren gepresenteerd die enkel door de valide zoekruimte navigeren, en daardoor geen last hebben van de grote invalide regio's die voorkomen bij grotere supercube afmetingen.

Hoofdstuk 6 bekijkt hoe, in een multi-criteria setting, een lokale zoekstrategie bij kan dragen aan de verbetering van oplossingen die tijdens het globale zoekproces gevonden zijn. De resultaten laten zien dat *hypervolume indicator gradiënt stijging multi-criteria optimalisatie* (HIGA-MO) werkt in een praktijkgerichte setting, evenals met numeriek benaderde gradiënten. Ondanks dit positieve resultaat, weerhoudt het aantal evaluaties dat nodig is voor de numerieke benadering in deze hoog-dimensionale setting HIGA-MO ervan betere resultaten te produceren dan het gebruikte evolutionaire algoritme (SMS-EMOA-SC). Dit betekent dat als lokale zoekcomponent HIGA-MO waarschijnlijk uiteindelijk wel beter zou zijn dan SMS-EMOA-SC, dankzij de convergentie-eigenschappen van de hypervolume gradiënt, maar het aantal benodigde evaluaties zou ondoenlijk zijn.

De optimalisatieprocessen genereren een schat aan data. De vraag is dan ook: Wat kan er van deze data geleerd worden over gebouw-ruimtelijk ontwerp (Hoofdstuk 7)? Gebaseerd op de optimalisatiedata kunnen door middel van het automatisch extraheren van ontwerpreglementen sleuteleigenschappen van hoge kwaliteit gebouw-ruimtelijke ontwerpen geleerd worden. Deze ontwerpreglementen kunnen vervolgens gebruikt worden om aan een ontwerpdeskundige te laten zien dat de ontdekte oplossingen betrouwbaar zijn. Deze reglementen komen over het algemeen namelijk overeen met welbekende ontwerpreglementen die gebruikt worden door de ontwerpdeskundigen.

Volgend op de ontwikkeling van optimalisatiemethodes voor het gebouw-ruimtelijk ontwerpprobleem, wordt er in Hoofdstuk 8 verkend of het mogelijk is een generiek multi-criteria gemengd-geheeltallig evolutionaire strategie (MOMIES) te ontwikkelen. Er zijn veelbelovende eerste stappen gemaakt die laten zien dat dit in de praktijk goed werkt. Verder is er ontdekt dat recombinatie waardevoller is in multi-criteria optimalisatie dan voorheen werd gedacht, hoewel het nog onduidelijk is hoe dit komt. Daarnaast is er nog nader onderzoek nodig naar de aanpassing van de stapgrootte in de

multi-criteria setting, de huidige methodes laten namelijk nog onberekenbaar gedrag zien en zijn niet betrouwbaar in het volgen van de optimale mutatiestapgrootte.

Als laatste bijdrage van dit proefschrift wordt er onderzocht in hoeverre de ontwikkelde algoritmes toepasbaar zijn op praktische problemen (Hoofdstuk 9). SMS-EMOA-SC heeft bewezen een waardevol component te zijn in een hybride algoritme dat ook verkende welke supercube-configuraties het meest waardevol waren om mee te werken. Verder is duidelijk geworden dat SMS-EMOA-SC een goede optimalisatiemethode is, maar dat de instellingen en de doelfuncties die eerder gebruikt werden in de academische experimenten niet altijd direct aansluiten op de realiteit van de praktische problemen. Voor de optimalisatie van nominaal-discrete variabelen van structurele ontwerpen is MOMIES een zeer effectief algoritme gebleken.

Samenvattend heeft deze scriptie technieken beschikbaar gemaakt voor multi-criteria optimalisatie van gebouw-ruimtelijke ontwerpen. Om dit te bereiken is er een representatie geïntroduceerd, evenals gespecialiseerde variatieoperatoren voor het gebouw-ruimtelijk ontwerpprobleem. Daarnaast zijn de potentiële voordelen van een memetisch algoritme onderzocht, waar stochastische globale zoekprocessen gecombineerd worden met deterministische lokale zoekprocessen. Ook is de praktische toepassing van de methodes onderzocht. Zowel door de analyse van de optimalisatiedata, als door de toepassing op praktische problemen. Tot slot is er een generiek algoritme ontwikkeld, dat niet gelimiteerd is tot het gebouw-ruimtelijk ontwerpprobleem.

Curriculum Vitae

Koen was born on the 27th July 1990 in Berkel en Rodenrijs. After graduating with a B ICT from The Hague University of Applied Sciences in 2012, he completed his MSc at Leiden University in 2014. Under the supervision of Michael Emmerich, Hèrm Hofmeyer (Eindhoven University of Technology), and Thomas Bäck he conducted the research presented in this doctoral thesis at the same university. Currently, still in Leiden, he is a post-doctoral researcher in meta-algorithmics supervised by Holger Hoos.

Glossary

(building) spatial design

The design of the external and internal geometrical shape (of a building)

binary space / the space of binary numbers / $\mathbb{B}^n \in \{0,1\}^n$

The space of n-tuples of integers restricted to two possible values, usually zero and one

categorical / nominal discrete space

The space of tuples of categorical variables (usually encoded by integers)

decision / variable / search / design space

The space of candidate solutions in the chosen representation

discrete space

The space of tuples of integers and categorical variables (usually encoded by integers)

feasible space

The space of solutions that do not violate any constraints

hypervolume (indicator) / $\mathcal{S}$-metric

The volume covered by a set of points relative to a reference point

infeasible space

The space of solutions that violate one or more constraints

integer space / the space of integer numbers / $\mathbb{Z}^n$

The space of n-tuples of whole numbers; positive numbers, and possibly also negative numbers and zero

mixed-integer space

The space of tuples containing a combination of real and discrete values

objective space / solution space

The space of solution/objective performances, typically a subset of $\mathbb{R}^m$, where m is the number of objective functions

real space / the space of real numbers / continuous space / $\mathbb{R}^n$

The space of real vectors of dimension n

space

Part of a building spatial design, e.g. a room, corridor, atrium, etc.

structural component

Structural component of a building, e.g. a beam, strut, slab, etc.

structural design

The set of structural components with their position and properties, which circumvents and distributes forces

subspace

Part of a more general collection; integer space is a subspace of real space, for instance

supercube

A superstructure to encode building spatial designs

superstructure

A design space representation that represents a relevant subset of the entire design space and each element of the superstructure is a solution that is encoded by a vector of constant length

superstructure free

A design space representation that is not encoding solutions by using a superstructure

Acronyms

AEC
Architecture, Engineering, and Construction

BGO
Bayesian Global Optimisation

BIM
Building Information Modelling

BP
Building Physics

CAD
Computer Aided Design

CD
Co-evolutionary Design

CFD
Computational Fluid Dynamics

EA
Evolutionary Algorithm

EDS
Enhanced Directed Search

EMO
Evolutionary Multi-Objective Optimisation

EMOA
Evolutionary Multi-Objective Optimisation Algorithm

ES

Evolution Strategy

FE

Finite Element

FEM

Finite Element Method

HIGA-MO

Hypervolume Indicator Gradient Ascent Multi-Objective Optimisation

HIGA-MO-SC

Hypervolume Indicator Gradient Ascent Multi-Objective Optimisation SuperCube

HVI

Hypervolume (Indicator)

MEMO

Memetic Multi-Objective Optimisation

MEMO-SC

Memetic Multi-Objective Optimisation SuperCube

MIES

Mixed-Integer Evolution Strategy

MINLP

Mixed-Integer Nonlinear Programming

MOMI

Multi-Objective Mixed-Integer

MOMIES

Multi-Objective Mixed-Integer Evolution Strategy

MOP

Multi-Objective Optimisation Problem

NSGA-II

Nondominated Sorting Genetic Algorithm II

ODE

Ordinary Differential Equations

PF

Pareto front

PFA

Pareto front approximation

RC

Resistance/Capacitance

SD

Structural Design

SMS-EMOA

$\mathcal{S}$-Metric Selection Evolutionary Multi-Objective Algorithm

SMS-EMOA-SC

$\mathcal{S}$-Metric Selection Evolutionary Multi-Objective Algorithm SuperCube

XML

eXtensible Mark-up Language

Symbols

α ratio between desired and current volume

$\mathbf{B}$ binary matrix

$b^{\ell}_{i,j,k}$ indicator of activity for a specific cell belonging to a specific space for the supercube

C heat capacity

c accumulation coefficient

CV number of constraint violations

D geometry of a space in the movable sizeable representation

d number of decision variables

$\mathbf{d}$ vector of nominal discrete decision variables

d_j size of a specific depth division of the supercube

E Young's modulus

$\mathbf{e}_j$ j-th standard basis in $\mathbb{R}^d$

$f(\cdot)$ objective function

$\mathbf{F}$ mapping between $\mathbf{X}$ and $\mathbf{Y}$

$\mathbf{f}(\cdot)$ vector of objective functions

FS fixed number of steps

$G(\cdot,\cdot)$ Gaussian distribution

$g(\cdot)$ equality constraint function

G_{norm} normalised subgradient

$H(\cdot)$ hypervolume of a set of objective vectors

$h(\cdot)$ inequality constraint function

$\mathcal{H}_{\mathbf{F}}(\cdot)$ hypervolume of a set of decision vectors

h_k size of a specific height division of the supercube

I inner product of normalised HVI subgradients in two consecutive iterations

$\mathbf{I}$ integer matrix

IM initialisation mutations

IT initialisation technique

k thermal conductivity

κ maximum number of generations that an individual can stay in the population

$\mathbf{K}$ stiffness matrix of an element

L coordinates of the location of a space's origin in the movable sizeable representation

Λ Lebesgue measure

λ offspring population size

λ_k tournament size

lb lower bound

M heightmap

m number of objectives

MC continuous mutation probability

MP mutation probability

MT mutation type probability

μ parent population size

N_{cells} number of cells

N_{cont} number of continuous variables

N_d number of divisions in depth for the supercube

N_{dims} number of decision variables in a given supercube

N_h number of divisions in height for the supercube

N_{spaces} number of spaces

ν Poisson's ratio

N_w number of divisions in width for the supercube

p cumulative value of I

pen penalty value

p_{live} live load

$p_{w,p}$ wind load, pressure

$p_{w,s}$ wind load, suction

$p_{w,sh}$ wind load, shear

Q_c cooling power

Q_h heating power

$\mathbf{r}$ vector of continuous decision variables

$\boldsymbol{\rho}$ reference point

ϱ density

RP recombination probability

$\mathbb{S}$ set of solutions

s a space in the moveable sizeable representation

S_A total outside surface area

$\mathbf{s}$ a vector of spaces in the moveable sizeable representation

S_d total outside surface area of depth vectors

s_d shape depth

$sgn(\cdot)$ sign function

S_h total outside surface area of height vectors

s_h shape height

σ step size of continuous variables

ς step size of integer variables

ST step size technique

S_w total outside surface area of width vectors

s_w shape width

t thickness

τ_1 Local learning rate

τ_2 Global learning rate

T_c cooling set point temperature

T_g ground temperature

T_h heating set point temperature

$U(\cdot, \cdot)$ Uniform distribution

ub upper bound

$\mathbf{u}$ displacement vector of an element

V_0 total volume

V_c current volume

w_i size of a specific width division of the supercube

X finite set of decision vectors

x decision variable

$\mathbf{X}$ vector of concatenated decision vectors

$\mathbf{x}$ vector of decision variables

Y finite set of objective vectors

y objective value

$\mathbf{Y}$ vector of concatenated objective vectors

$\mathbf{y}$ vector of objective values

$\mathbf{z}$ vector of integer decision variables

ζ step size of nominal discrete variables