TU WIEN Informatics

Many-Objective Optimization for Maximum Flexibility in Industrial Building Design

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Visual Computing

eingereicht von

Xi Wang-Sukalia, B.Sc.
Matrikelnummer 01226083

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Mag.rer.nat. Dr.techn. Hannes Kaufmann
Mitwirkung: Mag. Dr.techn. Peter Kán

Wien, 11. Mai 2022

Xi Wang-Sukalia	Hannes Kaufmann

TU WIEN Informatics

Many-Objective Optimization for Maximum Flexibility in Industrial Building Design

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Visual Computing

by

Xi Wang-Sukalia, B.Sc.
Registration Number 01226083

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Mag.rer.nat. Dr.techn. Hannes Kaufmann
Assistance: Mag. Dr.techn. Peter Kán

Vienna, 11th May, 2022

Xi Wang-Sukalia	Hannes Kaufmann

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Xi Wang-Sukalia, B.Sc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 11. Mai 2022

Xi Wang-Sukalia

Acknowledgements

First of all, I would like to thank the BIMFlexi team that I could be part of this project. Special thanks goes to my BIMFlexi colleagues Julia Reisinger, Maria Zahlbruckner and Maximilian Knoll from whom I have learned a lot about industrial building design and production planning, and who supported me during my implementation and provided me with data for the studies.

Furthermore, I would like to thank my supervisors Dr. Hannes Kaufman and Dr. Peter Kán. Despite my tight schedule, I received valuable, timely feedback, ideas for improvements and excellent proofreading.

I want to thank my friends, who motivated me and helped me to keep my mood up. Special thanks goes to Xiang, who also helped me proofreading.

Last but not least, my thanks goes to my family, especially my husband, Fabjan, who always would lend me an ear when I needed it and supported me in many different ways. Without him, I would not have been able to finish this thesis.

Kurzfassung

Industriegebäude haben oft einen kurzen Lebenszyklus aufgrund von inflexiblen Tragwerken. Die sich ständig ändernden Produktionsprozesse führen zum Abriss von Industriegebäuden, da diese sich nicht an ihre neuen Anforderungen adaptieren lassen. Die vorliegende Arbeit ist Teil des *BIMFlexi* Projektes, dessen Ziel es ist eine auf Building Information Modeling (BIM) basierende Plattform zu entwickeln, die Produktionsplanung integriert, und somit alle Interessensvertreter_innen eines Gebäudeplanungsprozesses hilft flexible and nachhaltige Industriegebäude zu entwerfen.

In der vorliegenden Arbeit wird ein multikriterielles Optimierungstool präsentiert, um Entscheidungsträger_innen in der Designphase zu unterstützen. Das Tool baut auf einem parametrischen Framework für Tragwerksgenerierung auf. Durch das Präsentieren von mehreren Tragwerkskonstruktionen mit verschiedenen Eigenschaften können Entscheidungsträger_innen eine informierte Entscheidung über unterschiedliche Kompromisse zwischen Kosten, Umweltauswirkungen und Flexibilität eines Tragwerkes treffen. Das Tool wurde auf zwei unterschiedliche Arten untersucht. Eine Benutzerstudie wurde durchgeführt, um die Benutzerfreundlichkeit und die Zweckmäßigkeit zu untersuchen. Die zweite Studie untersucht drei verschiedene evolutionäre Algorithmen, um den passenden Algorithmus für die Optimierung von Industriegebäuden zu finden.

Abstract

Industrial buildings often have a very short lifespan due to inflexible design of load bearing structures. Frequently changing production processes often lead to demolition of industrial buildings because these buildings cannot be adapted to the new requirements. This work is part of the BIMFlexi project, whose goal is to develop an integrated Building Information Modeling (BIM) based platform to connect all stakeholders in a building planning process to design flexible and sustainable industrial buildings.

In this work a many-objective optimization tool is presented to support decision makers during the design phase. The tool is built on top of a parametric framework for load bearing structure generation. By presenting multiple optimized load bearing structures with different properties decision makers can make informed decisions about trade-offs between cost, environmental impacts and flexibility of a load bearing structure. The tool has been studied in two different ways. A user study was conducted to verify its usability and usefulness. A second study compared three different evolutionary algorithms to find the best fitting algorithm for industrial building optimization.

Contents

CHAPTER 1

Introduction

1.1 Motivation and Problem Statement

Due to fast changing production processes, current industrial buildings often lack the flexibility to adapt to their new usage. This often results in demolition and new construction of the building which does not only lead to high costs but also increased resource consumption whose production has a high impact on the environment. Therefore, flexibility and expandability of buildings with respect to production systems are main goals in industrial building design.
In order to maximize flexibility it is necessary to investigate possible effects of certain design decisions at an early stage of the design process as they have a great impact on the workflow of the remaining process and the life cycle of a building. This, however, is challenging in multiple ways. Industrial building design is a lengthy and complex design process that requires specialists from a variety of fields, including architects, structural designers, energy- and production planners as well as building owners. Furthermore, each stakeholder pursues different and sometimes conflicting objectives when designing or making decisions. Desired changes from one stakeholder may cause improvement in one objective but deteriorate another objective at the same time. Yet the effects of such changes are often only detected at a later stage where redesigning may be too late or too expensive.
Another problem arises when looking at the sheer number of possibilities on how an industrial building can be designed. Due to the complexity, designing takes a lot of time and it is not possible to design hundreds of options by hand. This limits the involved people to focus on one or two solutions instead of exploring and examining the possible effects of multiple different designs. While there are tools that work well for each individual part of the design process there is currently no holistic solution available to support the whole process to find the most sustainable and flexible design option.

1.2 BIMFlexi Project

This thesis is part of the BIMFlexi Project [BIM20], which is funded by the Austrian Research Promotion Agency. The goal of *BIMFlexi* is to develop an integrated Building Information Modeling (BIM) -based platform to connect all stakeholders of the industrial building planning process. It links discipline specific tools and provides data over the entire life cycle of an industrial building. Therefore, the platform enables planning, analyzing and optimizing of flexible buildings in one tool. BIMFlexi supports multidisciplinary teams to understand different goals and analyze structural designs and their effects to make decisions that maximize the overall value.
For this reason a parametric model was designed [RKK21]. For a given input set the framework automatically calculates different flexibility values, environmental values and life cycle costs. This enables the automatic generation of many different structure variations and therefore the optimization of structures for a given scenario. Using this platform stakeholders are able to shift their focus from designing to analyzing and evaluating structures and make holistically informed decisions.

1.3 Aim of this Work

In order to enable stakeholders to make informed decisions on trade-offs between different objectives one major part of the BIMFlexi project is the integration of a many-objective optimization. Based on the aforementioned information, this master thesis tries to answer the following questions:

1. *Does an optimization tool provide a good overview of different designs and supports users in their decision making?*
   The optimization tool should provide an easier and a more efficient way to explore the different designs. Furthermore, the results should be presented in such a way that the decision maker can compare the solutions against each other and is able to select a subset of the Pareto optimal solutions. Because of the huge search space and conflicting objectives this work focuses on the implementation of Pareto-based many-objective evolutionary algorithms (MaOEA).

2. *Is there an algorithm that works better than other algorithms?*
   The "No Free Lunch" theorem tells us that all optimization algorithms perform equally well when averaged over all possible problems [WM05]. Every MaOEA performs differently on different problems and there is no single algorithm that will always perform better. This holds true for many-objective optimization problems (MaOP) in real-world applications as well. Therefore, another focus of this thesis is to find an appropriate algorithm for our problem. We will investigate the performance of multiple algorithms in terms of convergence and diversity.

1.4 Methodology and Contribution

The methodological approach consists of four steps.

- First, an extensive literature research gives a deep understanding of the challenges in many-objective optimizations. In this step we investigate the optimization possibilities within Grasshopper3D as well.
- The second step consists of implementing the optimization tool into the BIMFlexi framework, including visualizing the data. We chose three multi-objective evolutionary algorithms (MOEA) to test: the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [ZLT01], SPEA2 with the Shiftbased Density Estimator (SDE) [LYL13] and the Constrained Two-Archive Evolutionary Algorithm (CTAEA) [LCFY18].
- A user study was conducted to verify the feasibility and quality of solutions found by the optimization algorithm as well as the usefulness of an optimization tool in general.
- Finally, we compared the optimization algorithms on three different test cases to analyze the performance in terms of quality on each algorithm.

The contribution of this thesis is a new optimization tool to complement the BIMFlexi framework. The current framework already provides a substantial improvement in the area of design exploration. With the addition of the optimization tool even more variants can be explored and compared without the time consuming, manual generation of different designs. Additionally, a user study and a comparison of algorithms is done to provide information for future improvements of the current framework.

1.5 Structure of the Thesis

The remainder of this thesis is structured as follows. First, in Chapter 2 a literature review is given. We look at related works and the current developments in three areas, industrial building design, multi-objective optimization and optimization in Grasshopper. This chapter also gives an explanation on what Pareto optimality is and the difference between multi- and many-objective optimization.
The optimization tool is presented in Chapter 3. This includes the implemented algorithms, visualization of data, the integration of the tool into the existing script of the parametric model and the workflow of the whole framework including the optimization.
To validate the tool a user study was conducted with master students from architecture and civil engineering from TU Wien. Additionally, an algorithm comparison was done with three different test cases. The three test cases also cover three different problem sizes, i.e. sizes of decision space. Results of the user study and the algorithm comparison are presented in Chapter 4. The results are then discussed in Chapter 5.
Finally, Chapter 6 concludes the thesis. Findings are summarized and suggestions for future steps are given.

CHAPTER 2

Literature Review

2.1 Sustainability and Flexibility in Industrial Building Design

One major goal in industrial building design is to improve the sustainability of the built environment. Especially industrial buildings consume large amounts of materials and energy. In 2018, the transformation of raw materials into construction materials alone accounts for roughly 11% of global emissions [Age19]. Many researchers point out that decisions made at an early design stage have a greater impact on the whole project, while decisions made at a later stage are much costlier and more difficult to implement. Kovacic et.al. [KWG16] for example developed a decision support tool for different façade systems of industrial building system. The results revealed that the initial costs diverge up to 27% while after 35 years the difference in life cycle costs is only 6%. The timber façade had the highest initial costs but produced the least emissions (80% less) and would therefore have been the best choice for the time period of 35 years. The study shows the decision impacts on the whole life cycle and suggests that decision supporting tools should be implemented in the early design stages where they can guide designers towards optimal solutions.

Optimization has been a popular tool to achieve improvements in various areas of building design, including sustainability. A recent meta study shows that it has been intensively applied to enhancing building's energy efficiency, improving thermal and visual comfort, minimizing life-cycle costs, and emission [MN22]. Optimization has been frequently used for design space exploration as well to present designers different alternative designs [SAG05] and guiding them towards more optimal solutions. Further research shows that Rhino and Grasshopper are the software packages that currently dominate the field [TT17] and numerous studies use optimization tools like Octopus, Galapagos and Colibri [MN22].

Due to frequently changing production processes, industrial buildings often have a short life span, resulting in early demolition and therefore high environmental impact. To counteract this, industrial buildings must be able to adapt, i.e. be flexible, to the changing requirements. Geraedts [Ger16] investigated different flexibility indicators and developed a catalog for flexibility assessment of existing buildings. Cavalliere et. al. [CDFL19] suggested assessment metrics for flexibility and showed that the flexibility of buildings can be automatically calculated during the design process phase. As the mentioned research does not focus on industrial buildings, they are only partially applicable in industrial building design.
In industrial buildings the load-bearing structure is one major limiting factor for flexibility. Therefore, to increase flexibility the focus should be on optimizing the load-bearing structure. Several researchers suggested and also have optimized the structure in different areas of building design but consider other objectives than flexibility. Boonstra et. al. presented [BvdBHE20] two methods to generate structural system layouts. The first method uses design rules to develop a structural system layout. The second method uses an evolutionary algorithm, multi-objective mixed-integer evolution strategy [vdBYBE19], to assign structural components to a buildings spatial design's geometry. The proposed methods have been demonstrated on two objectives, minimal strain energy and minimal structural volume. Gan et. al. [GWT$^+$19] developed a hybrid algorithm to find cost optimal high-rise reinforced concrete buildings with reduced carbon footprint. In the optimization they consider both, the structural topology and individual structural element size. The topology is optimized using a genetic algorithm while a gradient-based direct search technique is used to find the optimal element size.
Multiple papers have done research on multi-objective optimization over cross disciplines as well, as optimizing only one domain may deteriorate performance in other disciplines and vice versa. Yi et. al.[YTPB21] optimized a building for skylight roof system while considering cost, structural and energy performance. In their case study on a sawtooth roof they used Rhino with Grasshopper. Grasshopper was used to control geometry and to calculate the objectives. Afterwards the objectives were coupled to MATLAB [Mat] for optimization using the nondominated sorting genetic algorithm II (NSGA-II) [DPAM02]. Hamidavi et. al. [HAPB18] propose a structural design optimization framework using a genetic algorithm to explore early structural design alternatives. They state that such frameworks would improve the collaboration between the architectural stage and the structural stage. Brown et. al. [BDOOM16] suggest to integrate architectural and structural objectives into one multi-objective design tool in the early design process to better guide the designers towards high performing solutions. The authors implemented a suite of tools within Rhino, including the NSGA-II algorithm for multi-objective optimization and tested it on a cantilevered stadium roof. Using four parameters the structures were optimized for optimal structural efficiency, rain protection, sound dispersion and portion of visible sky. Similarly, Pan et. al. [PTL$^+$19] used multi-objective optimization to balance architectural and structural performance with the goal to integrate multi-functional indoor sports arenas with long-span structure in the early design stage. In a case study of a cantilevered stadium roof multiple structural

and spectator objectives have been optimized. Rhino and Grasshopper were used for the parametric design while the optimization was carried out in MATLAB using the NSGA-II algorithm.

In industrial building design the integration of flexibility and production planning is still rare. Yet changing production requirements is one driving factor for the need of flexibility. Furthermore, there is currently no holistic approach to combine life-cycle cost, life cycle environmental impact, production planning and flexibility into one decision tool for the early design stage. Adding multi-objective optimization would further enable designers to investigate different design alternatives, listing trade-offs between conflicting objectives. Such tool would show stakeholders the impacts of their decision for the whole life cycle of the building and therefore empower them to make sustainable decisions.

2.2 Multi- and Many-objective Optimization

2.2.1 Pareto Dominance, Optimality and Front

A multi-objective optimization problem (MOP) is a problem that involves two or more potentially conflicting objectives that need to be optimized simultaneously. A many-objective optimization problem (MaOP) is a MOP that deals with more than three objectives. Formally the problem can be modeled as follows

$$\min_{s.t.\ \vec{x}\in X}(f(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x}))) \tag{2.1}$$

Without loss of generality all objective functions are to be minimized. Objectives that are to be maximized can be multiplied by -1 to convert them to an equivalent minimization problem. For final visualization the sign may be inverted again. $f(\vec{x})$ is an objective function vector that consists of m objective functions. X is the decision space and $\vec{x} = (x_1, x_2, \ldots, x_D) \in X$ is the decision vector.
Since the objectives are conflicting, optimizing one objective will often worsen other objectives. Therefore, there is usually not only one solution that can be considered optimal but rather a set of trade-off solutions. This set is called Pareto optimal solutions, termed Pareto front or Pareto set. The number of solutions that form the Pareto front can be infinite. All Pareto optimal solutions can be considered equally good if there is no subjective preference, i.e. from the decision maker.
A solution is Pareto optimal or nondominated when it is not possible to improve one objective without worsening another one. Whether a solution dominates another solution can be defined by the dominance relation as follows: Let $\vec{x_1}, \vec{x_2} \in X$ then $\vec{x_1}$ dominates $\vec{x_2}$ ($\vec{x_1} < \vec{x_2}$ in case of minimization) if and only if

- $f_i(\vec{x_1}) \leq f_i(\vec{x_2})$ for all $i \in \{1, 2, ..., m\}$ and
- $f_j(\vec{x_1}) < f_j(\vec{x_2})$ for at least one index $j \in \{1, 2, ..., m\}$.

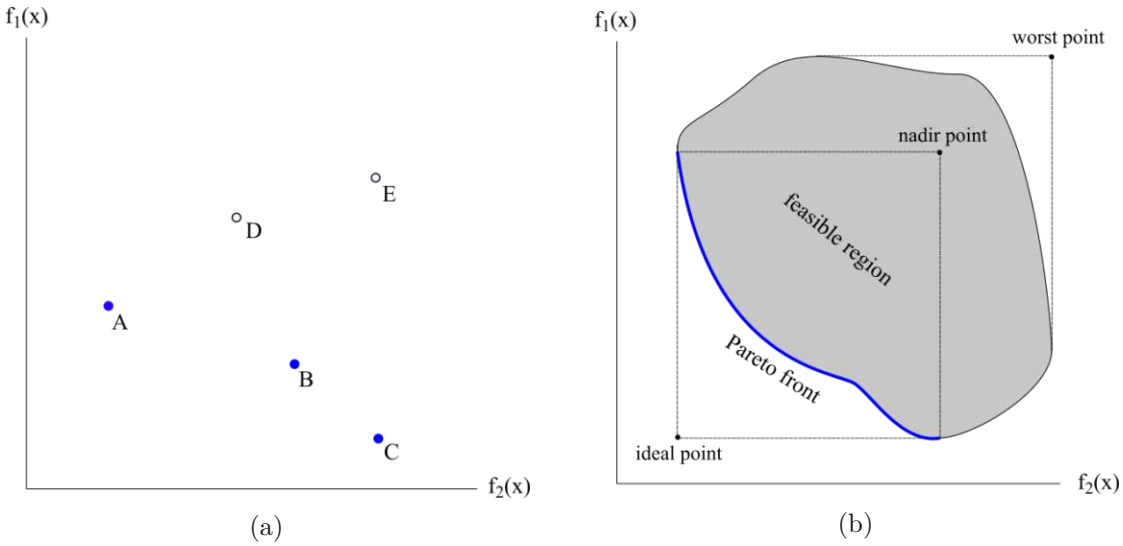


Figure 2.1: Dominance relation and points of interests. (a) The Pareto front consists of the solutions A, B and C (marked in blue). D is dominated by A. E is dominated by A, B, C and D. (b) The Pareto front is continuous (marked with blue). The ideal, nadir and worst points are shown in relation to the Pareto front.

A solution $\vec{x^*}$ is called Pareto optimal only if there is no $\vec{x} \in X$ such that $\vec{x} < \vec{x^*}$. An example of the dominance relations is given in Figure 2.1a. A, B and C are not dominated by any other solutions. Therefore, they form the Pareto front. E is dominated by all other solutions, while D is only dominated by A.

Related to the Pareto front there are further points, the ideal, nadir and worst point, that are of interest as they are often used for optimization or evaluation of the Pareto front. An example of all three points is given in Figure 2.1b
The ideal point z^*, sometimes called utopia point, is constructed by taking the minimum value of each objective. Formally, it is defined as $z^* = (z_1^*, z_2^*, ..., z_m^*)$ where $z_j^* = min_{\vec{x} \in X} f_j(\vec{x})$.
The worst point z^w is the opposite of the ideal point. It is the composite of all worst objective values. It is therefore defined as $z^w = (z_1^w, ..., z_m^w)$ where $z_j^w = max_{\vec{x} \in X} f_j(\vec{x})$.
The nadir point z^{nad} is similar to the worst point, but takes only the solutions on the Pareto front into consideration.

2.2.2 Multi-Objective Evolutionary Algorithms (MOEA)

When the search space is too large to practically examine it completely it becomes necessary to use metaheuristic algorithms. Evolutionary algorithms (EA) are population-based metaheuristics inspired by biological evolution. Using natural evolution as an analogy, each possible solution is called an individual and a set of individuals is called a population. Any individual consists of multiple inputs, called genes. All genes together

is named the genotype. The visible solution output, e.g. model, path, text, etc. of an individual is called the phenotype. EAs start with a set of initial solutions, usually randomly generated, that are then further optimized. The optimization process can be summarized into the following steps:

1. The current population is evaluated in the objective space. Each individual is assigned a scalar value, the fitness. The fitness contains information about many things, like the objective values, density, feasibility or dominance. It is used to make two solutions comparable.

2. A set of individuals is randomly sampled according to the fitness values for producing new individuals.

3. A recombination operator is applied to multiple individuals that were selected for reproduction, called parents. The recombination operator recombines different parts from the parents to produce new individuals, called offspring or children.

4. To avoid early convergence a mutation operator is applied with a certain probability to the newly generated individuals. The mutation operator changes small parts of the individual.

5. Finally, a new population, the next generation, is formed by selecting the best individuals from either the offspring set or the latest population. This way of selecting the new generation is also known as elitism.

One iteration of all the steps is called a generation. These steps are repeated until some exit conditions are met (e.g. time limit, maximum number of generations). The goal of an EA is to find or approximate the optimal solution. As we usually do not know the Pareto optimal sets for real world problems, we will only deal with the approximation of Pareto sets. In case of MOPs a good approximation cannot mathematically be exactly described as it is in the case of single objective problems. For a MOEA, however, there are two main goals that are used to guide the search for the Pareto set. The first one is to guide the search towards the Pareto front and the second one is to keep a diverse set of solutions. Keeping these two criteria in mind there are many different ways to approximate the Pareto front using MOEAs. In general, there are Pareto-based and non-Pareto-based approaches. Pareto-based approaches use Pareto dominance to calculate the fitness of an individual. Non-Pareto-based approaches use other metrics instead of Pareto dominance, like the hypervolume (volume of a n-dimensional body), to approximate the Pareto front. In Pareto-based approaches the dominance relation can be incorporated into an individual's fitness in different ways [ZLB04]. Some algorithms use the dominance count, i.e. the number of individuals one individual dominates, other use the dominance rank, i.e. the number of individuals one individual is dominated by. Another technique is to separate the population into multiple fronts of domination, and make use of the dominance depth to select individuals for the next generation.

Two well known representatives of this category are NSGA-II [DPAM02] and SPEA2 [ZLT01]. Since SPEA2 and parts of NSGA-II is implemented within our optimization tool, we will go through those two algorithms in more detail.

SPEA2 [ZLT01] is an elitist evolutionary algorithm. It keeps a separate memory, called archive, to store nondominated solutions during the whole search process in addition to the usual population. The fitness is calculated by determining the strength value $S(i)$, i.e. dominance rank, of each individual i in the population and archive, first. Based on the strength value the raw fitness $R(i)$ of an individual is defined as the sum of strength values from each individual j that dominates i. nondominated solutions will have a raw fitness of zero. As one can see, SPEA2 incorporates both dominance rank and dominance count into the fitness calculation. To keep a good diversity the algorithm further includes a density estimator. If there are more nondominated individuals than the archive can hold the algorithm will remove nondominated individuals from a more crowded region. In the original paper an adaptation of the k-nearest neighbor method was used as the density estimator. However, it can be exchanged with other density estimation techniques like SDE [LYL13], which we will discuss in the next subsection.

NSGA-II is an elitist evolutionary algorithm as well. Unlike SPEA2 it does not keep a separate archive for nondominated individuals. An important part of NSGA-II is the fast nondominated sort algorithm which divides the population into multiple fronts of dominance, i.e. dominance depth. This algorithm is also seen in other MOEAs, e.g. TAEA [LLTY14]. The first front is the current Pareto front, i.e. it contains all individuals that are nondominated. The second front contains individuals that are nondominated when removing all individuals from the first front, and so on. An example is given in Figure 2.2. This information is used to select individuals for the next generation. The pool of individuals for the next generation is filled as follows: Starting from the first front, each front is added into the pool until the next front i cannot be added completely. Then, NSGA-II uses a crowding function, similar to SPEA2, to determine which individuals in front i should survive into the next generation.

These Pareto-based algorithms have shown that they work very well when dealing with two or three objectives. Unless the objectives are highly correlated [IAON11], their performance drops drastically when the number of objectives increases.

2.2.3 Many-Objective Evolutionary Algorithms (MaOEA)

As the number of objectives rises, the problem becomes increasingly harder to solve. In literature MOPs with more than three objectives are referred to as many-objective optimization problems (MaOP) to highlight their complexity. With the increasing number of objectives both diversity and convergence within the solutions become challenging. An increase of objectives means an exponential increase of solutions needed to reasonably represent the Pareto front [IS19]. To keep a good diversity within the solutions, a large number of individuals need to be generated. This is computationally very expensive. Furthermore, because of the huge Pareto front it is very likely that a generated individual

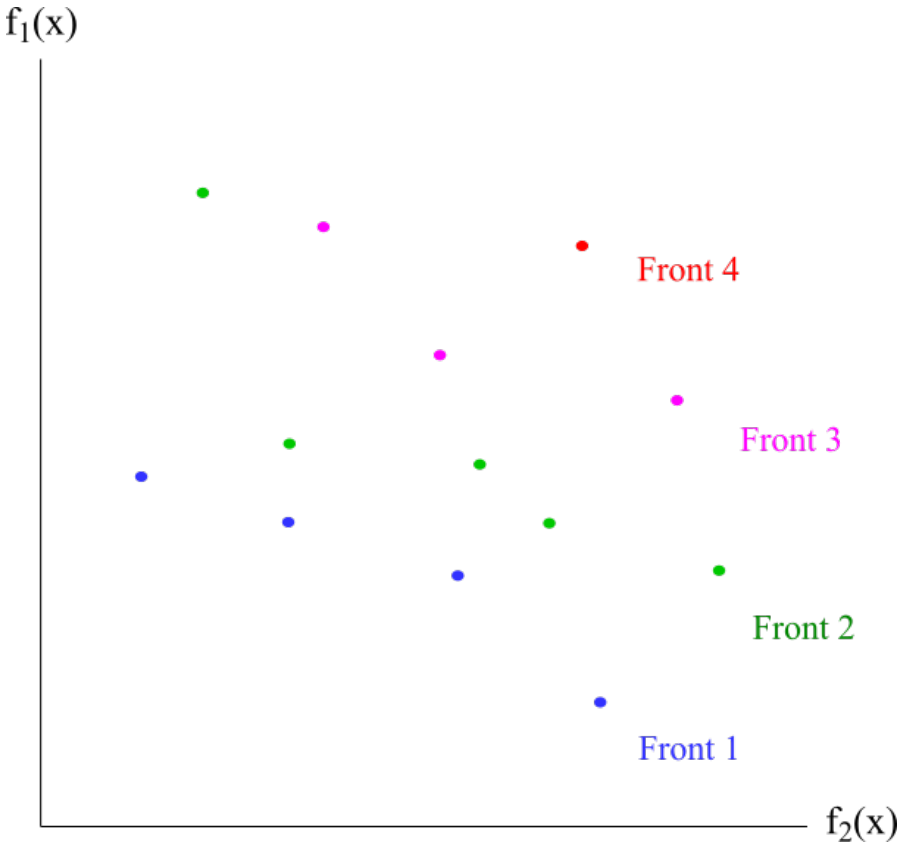


Figure 2.2: nondominated sorting.

is not dominated by any other solutions. The resulting problem is that there is not enough selection pressure towards the actual Pareto front and therefore convergence performs poorly. As conventional Pareto-based approaches fail to provide satisfactory results, new and improved algorithms have been developed that specifically target these problems.

One way to deal with MaOP is to decompose the complex problem into multiple simpler ones. This method is a non-pareto-based approach. Multiple objectives can be aggregated into a single one where each objective is multiplied by a weight. By applying a set of weights a MaOP can be decomposed into multiple sub-problems that can be optimized simultaneously. MOEA/D [ZL07] and MSOPS-II [Hug07] are representatives of this method. Other methods use reference vector-guided approaches to partition the objective space into subspaces like RVEA [CJOS16] or NSGA-III [DJ13] for example.

Another non-Pareto-based approach is to use different performance indicators as a guide in the search process as suggested in [ZK04], i.e. IBEA. A list of indicators for multi-objective optimization (MOO) have been summarized by Audet et. al. [ABC+20]. One indicator that measures both convergence and distribution is the hypervolume value. The hypervolume has been used in SMS-EMOA [BNE07] for example. Because of its

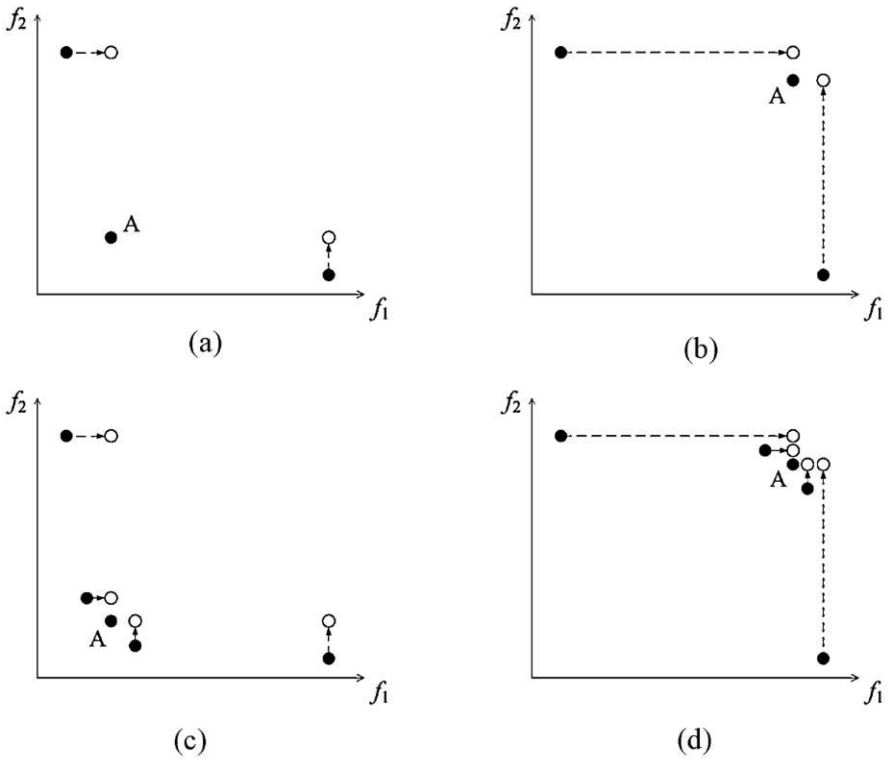


Figure 2.3: SDE for four different situations of an individual A in a minimization problem. (a) A population with good convergence and diversity. After applying the shift operator there is still a good convergence and diversity. (b-d) The shift operator puts A into a crowded region when the population has a poor convergence and/or poor diversity. [LYL13]

computational complexity HypE [BZ11] uses Monte Carlo simulation to approximate the hypervolume.

Other approaches combine the advantages of various algorithms to balance convergence and diversity. MOEA/DD [LDZK14a] for example combines dominance and decomposition-based approaches. Two_Arch2 [LLTY14] applies two archives each focusing on convergence and diversity separately.

Since the traditional Pareto dominance is insufficient for an MaOP, many researchers have suggested using a modified version of it. Hence, ϵ-dominance [LTDZ02], L-optimality [ZCLK08], fuzzy dominance [WJ07], θ-dominance [YXWY15] and many others were introduced. Further algorithms have been developed that modify the secondary criterion related to diversity only to achieve good balance between convergence and diversity. The main idea is that diversity maintenance becomes even more important when the population is mostly saturated with nondominated solutions, which is usually the case in a MaOP. When the population is filled with nondominated solutions the diversity maintenance mechanism is the only mechanism that is able to further push the population

towards the Pareto front. Therefore, at this stage a density estimator needs to take not only diversity into account but convergence as well. The shift-based density estimation (SDE) [LYL13] is an example for that. SDE can be used in any existing algorithm that incorporates some kind of density estimation, e.g. SPEA2 or NSGA-II. To estimate the density of an individual i, SDE shifts the position of other individuals in the population based on the comparison of objectives between these individuals and i. Using this method individuals with poor convergence and/or poor diversity will be automatically shifted into a more crowded region, while individuals with good convergence and diversity will stay in a less crowded region (see Figure 2.3).

2.2.4 Constrained MOP and MaOP

In practice optimization problems are often constrained. Liu et. al. [LW19] differentiate between constraints in the decision space and objective space. Objective space constraints are described through objectives while decision space constraints can be described through the decision variables. In the case of our load-bearing structure the size of the structure is constrained by the property size. This would be a decision space constraint. On the other hand structural stability is an objective constraint. Formally, constraints can be defined by equality and inequality equations.

$$\begin{aligned} &\text{minimize } f(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})) \\ &\text{subject to } g_j(\vec{x}) \leq 0, j = 1, 2, ..., q \\ &\qquad\qquad h_j(\vec{x}) = 0, j = q+1, ..., l \end{aligned} \tag{2.2}$$

$g_j(x)$ and $h_j(x)$ are the jth inequality and equality constraints, respectively. The degree of constraint violation (CV) over all constraints can be expressed as

$$CV(\vec{x}) = \sum_{i=1}^{l} CV_i(\vec{x}) \tag{2.3}$$

and $CV_i(\vec{x})$ is the degree of constraint violation on the ith constraint.

$$CV_i(\vec{x}) = \begin{cases} max(0, g_i(\vec{x}), & \text{if } i \leq q \\ max(0, |h_i(\vec{x})| & \text{otherwise} \end{cases}, i = 1, ..., l. \tag{2.4}$$

x is feasible when $CV(x) = 0$. Now, additionally to diversity and convergence an algorithm has to be able to balance the feasibility aspect as well. This raises another challenge for MaOP. In literature the handling of constraints can be divided into multiple categories [LCFY18].

In the first category feasibility has the highest priority. The search process is mainly driven by the feasibility information, i.e. feasible solutions are always ranked better than infeasible ones and therefore will have a higher priority to survive. The easiest way to

implement this kind of priority setting is to simply discard all infeasible solutions. Deb et. al [DPAM02] proposed a constrained dominance relation for constrained multi-objective optimization problems (CMOP). A solution $x1$ constrained-dominates $x2$ if

1. $x1$ is feasible and $x2$ is not feasible
2. $x1$ and $x2$ are feasible and $x1$ dominates $x2$
3. $x1$ and $x2$ are infeasible and $CV(x1) < CV(x2)$

By simply replacing the Pareto dominance relation with the constrained dominance relation, a number of MaOEA and MOEA can be adjusted to tackle CMOPs.

Another set of constraint handling methods try to balance the trade-off between convergence and feasibility. Infeasible solutions will not automatically have a lower priority to survive but are rather evolved towards feasibility. Woldesenbet et. al [WYT09] combine objective function value and the sum of constraint violation to decide between feasible and infeasible solutions. In [NGY+17] each solution is assigned a constrained nondominated rank based on its constraint violation degree and Pareto rank.

Instead of evolving infeasible solutions into feasibility over time, they can also be repaired into feasibility e.g. by using local search [HSOK07] or other metaheuristic approximation methods like simulated annealing [SRS10].

The mentioned strategies for handling constraints mainly concentrate on feasibility. However, overemphasizing feasibility may lead to early convergence. The objective space can be complex and might have disjoint feasible regions. When removing infeasible solutions too early disjoint regions closer to the Pareto front may never be explored. Li et. al. [LCFY18] therefore suggest that all three criteria, diversity, convergence and feasibility should be equally important and should be balanced simultaneously during the search process.

In recent years algorithms have been suggested that keep feasible and infeasible solutions separated. A set of infeasible solutions are maintained to keep a balance between feasibility, diversity and convergence. Yi et. al. [YYHW20] propose to solve a MaOP in two steps. In the first step constraints are ignored and the main goal is to optimize the objectives only. nondominated feasible solutions are stored at this stage. In the second step the nondominated feasible solutions are further evolved and more emphasis is given towards feasibility. This framework works with any existing MaOEA.
CTAEA [LCFY18], which we use as one of our algorithms, has two archives, called the convergence-oriented archive (CA) and diversity-oriented archive (DA), in addition to the normal population. The first archive, CA, keeps a nondominated, feasible and diverse set of solutions. If there are more nondominated and feasible solutions than the archive can hold, then the fast nondominated sorting approach with a density estimator using reference vectors, is applied to take the best individuals. At the end, CA will hold the feasible Pareto front. The goal of the second archive, DA, is to explore the search space

regardless of feasibility to prevent getting the algorithm stuck at a local optima. DA holds the unconstrained Pareto front. One important step in CTAEA is the selection, which will balance convergence, diversity and feasibility. During the selection process both archives are combined into one set $H_m = \text{CA} \cup \text{DA}$. The selection of the first parent is dependent on the proportion of nondominated solutions of CA and DA in H_m. If CA has a higher proportion of nondominated solutions in H_m, it means that CA has a better convergence than DA. Therefore, the first parent is chosen from CA. Otherwise, the first parent will be chosen from DA. The second parent is dependent on the proportion of nondominated individuals in CA. The more nondominated solutions the higher the possibility the second parent comes from CA.

As one can see there are many different MaOEAs including different ways of handling constraints. In the next section we investigate the optimization possibilities within Grasshopper3D.

2.3 Optimization in Grasshopper3D

Rhinoceros3D [Rhi] is a 3D computer graphics and computer aided design (CAD) software. It is a modelling environment often used by architects, civil engineers and industrial designers for rapid prototyping. Rhino itself is a non-parametric modeller, but includes the parametric modelling and visual programming tool Grasshopper3D [Gra] since Rhino 6. Before that Grasshopper3D was available as plugin. It enables people with basically no programming knowledge in the sense of writing code, to develop their own functionalities and to generate complex procedural geometry.

A Grasshopper script consists of multiple functional components arranged on a canvas. The component's complexity range from simple mathematical operations (e.g. add, subtract, etc.) to complex structural analysis like in Karamba3D. Components are connected to each other via wires that represent data flow (see Figure 2.4). Data flow within Grasshopper can be represented as a directed acyclic graph (DAG), which comes with some restrictions. Simple loops are not possible and components are processed sequentially. Furthermore, in Grasshopper any action in one component will trigger an update downstream regardless of whether there was a change or not. On the other hand Grasshopper makes extensive use of lists and trees, which makes data flow much easier.

2.3.1 Integration of Optimization Plugins in Grasshopper

Grasshopper comes with an integrated optimization tool called Galapagos which is the base of most existing optimization tools for Grasshopper. It was created by David Rutton, the author of Grasshopper. It supports a Genetic algorithm (GA) and Simulated Annealing (SA) for single objective optimization. Because of the DAG representation, Galapagos must be a component outside of the DAG otherwise it would trigger an endless loop. Therefore, its connection works differently. Galapagos is connected using only outgoing wires (see Figure 2.5). On the one side all genes are connected and on the other

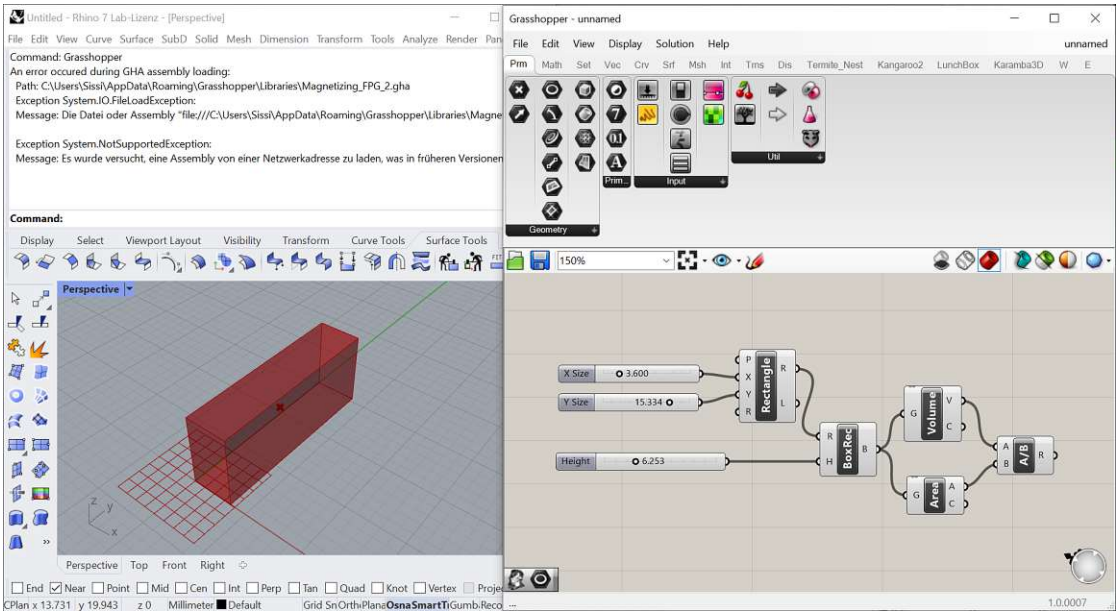


Figure 2.4: Simple example of a Grasshopper script (right) that calculates the surface-area-to-volume-ratio. Data flow is from left to right. Rhino (left) visualizes the geometry that is generated through the script.

side one objective value is connected. Genes are represented by sliders. Each combination of slider value is a different solution to the problem. Galapagos does not offer the ability to connect constraints. Any constraints need to be reformulated into an objective and incorporated into the fitness function.

Upon double clicking the component, an UI window opens up where the actual optimization can be started. The UI offers different settings for the optimization and lets the user choose between the GA and SA solvers. During the optimization process Galapagos changes the slider values, issues a recalculation in Grasshopper and retrieves the connected objective value. Changes during optimization are visualized in the UI window. These steps are repeated until an exit condition is met.

Galapagos can be used for MOO when using aggregation methods like weighted sum or reformulating the objectives and adding dependencies. Nonetheless it was not designed for MOO. However, it shows the basic steps to implement an optimization tool. Many other optimization tools extend on these ideas. Further optimization tools can be installed as plugins from the food4Rhino site [foo]. Most of them have their focus on single objective optimization, only few support MOO. In the following we will look at some of the MOO plugins.

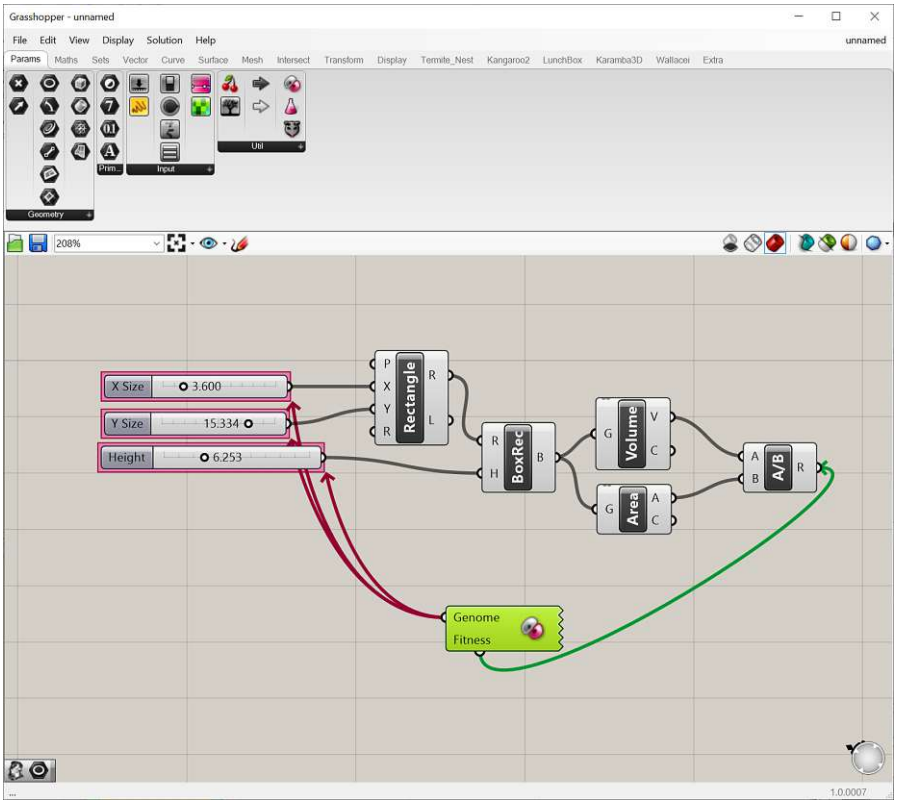


Figure 2.5: Wiring of Galapagos in Grasshopper.

2.3.2 MOO Plugins for Grasshopper

Octopus

One of the first publicly available plugins for MOO is Octopus [Vie13]. Its goal was to offer a flexible design tool for exploration and optimization for architectural engineering. Much as Galapagos, Octopus has outgoing wires with the difference that more than one objective value can be connected. It offers two different EAs, SPEA2 and HypE. Furthermore, Octopus allows adding one constraint, by wiring a boolean component to the objectives input, were true means that a solution is feasible.

The GUI visualizes the optimization process in multiple different ways (see Figure 2.6). In the main viewport solutions are shown as cubes, or meshes if a mesh was connected to Octopus. The cube has five properties, x-, y-, z- position, color and size, each representing one objective. Since there are only five properties, MOP with more than five objectives cannot be fully visualized with this method. However, it is a visualization that is rather intuitive and easy to learn.
Another visualization found in Octopus is the parallel coordinate plot. Parallel coordinate plots are a common way to visualize high-dimensional datasets. No information is lost using this method. The interpretation of this plot is quite complicated. In Octopus it

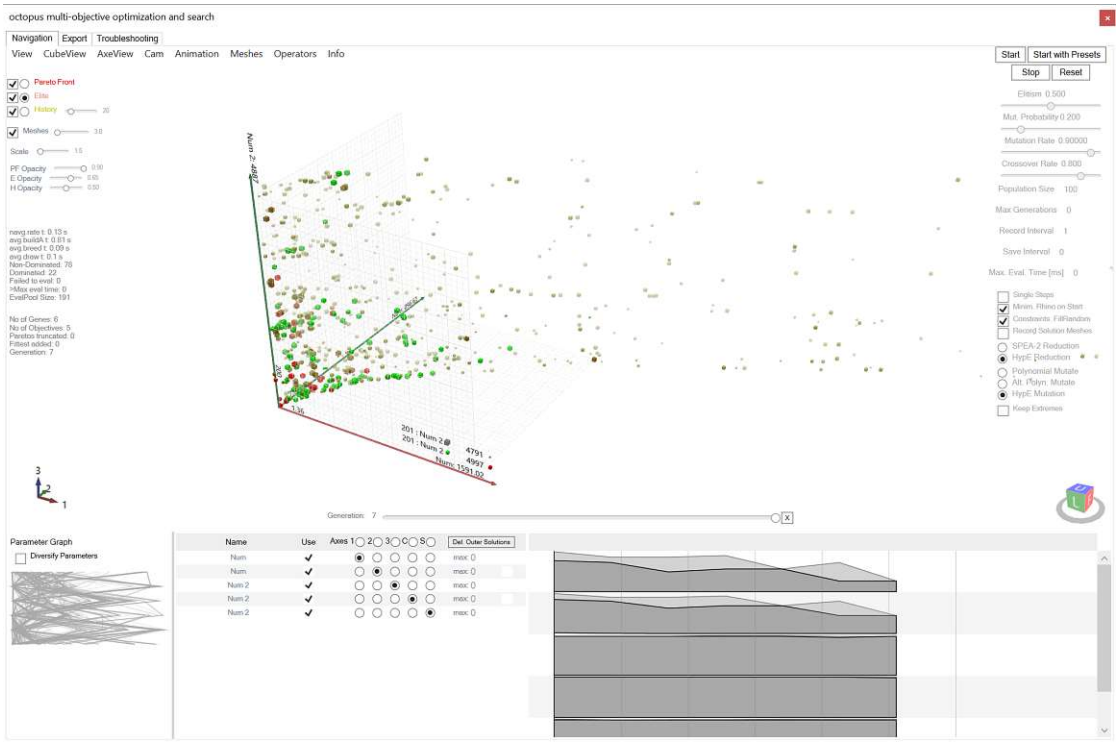


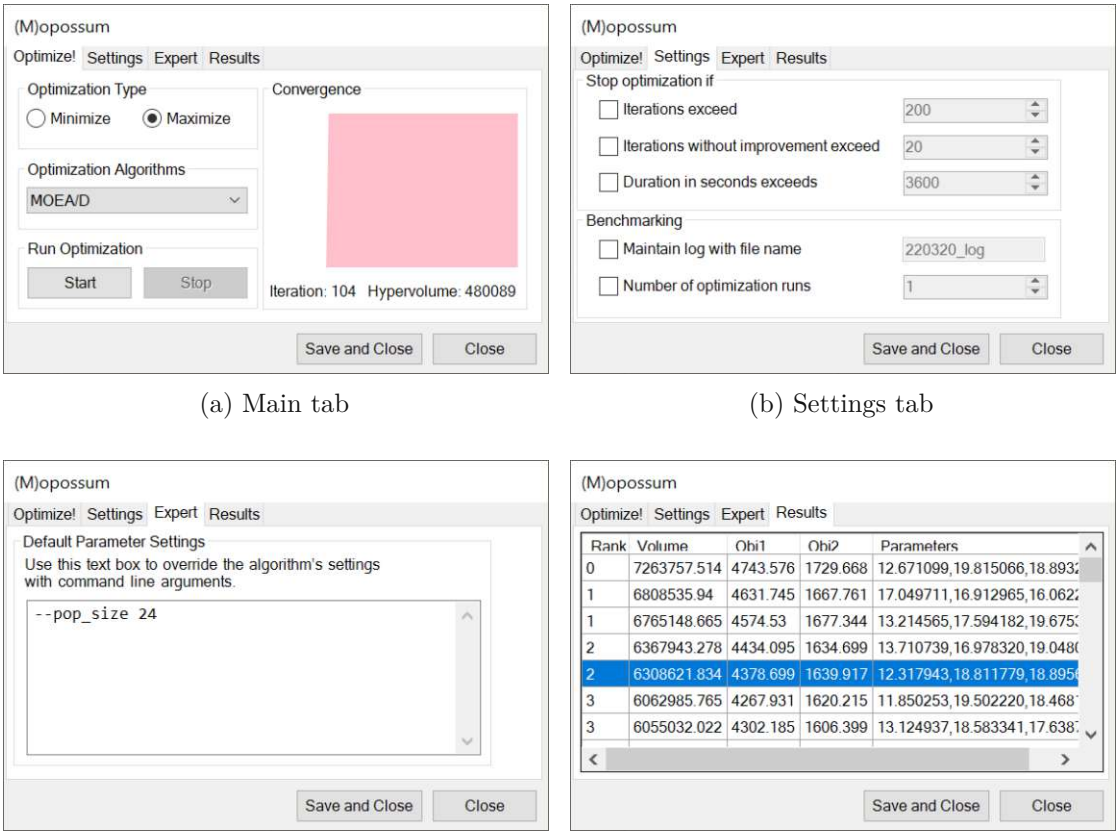
Figure 2.6: Graphical user interface of Octopus.

is a visualization that complements the cube visualization by highlighting the line of selected solutions in the main viewport.
Finally, a third plot shows the convergence of solutions on each objective. The elite solution is compared against the upper- and lower-bounds of the Pareto front. The development of each objective is plotted separately.
After optimization the numerical results can be exported as text.

Opossum

Opossum [Opo] is a plugin that originally started out as a model-based optimizer that targeted computationally intensive simulations, e.g. day-lighting or building energy. Since its publication there has been updates adding multiple MOO algorithms including MOEA/D and NSGA-II. It further supports other MOO algorithms such as RBFMOpt [RBF], MACO (Ant Colony) and NSPSO (Particle Swarm). To the best of our knowledge, Opossum does not directly handle constraints. Constraints therefore need to be reformulated into objectives.

Opossum has a simpler GUI design consisting of four tabs (see Figure 2.7). In the first tab users can select an algorithm and set the optimization type, i.e. whether the current problem is a minimization or maximization problem. Furthermore, there is a visualization of the hypervolume. The hypervolume shows the convergence of the search.



(a) Main tab

(b) Settings tab

(c) Expert tab

(d) Results tab

Figure 2.7: Graphical user interface of Opossum.

If the hypervolume value does not change much then the search is slowly converging and it may be stopped earlier. The second tab holds settings for the stopping criteria of the optimization process, like the maximal number of iterations without improvements. In the 'Expert' tab many more settings can be set through command line arguments, e.g. population size, maximum number of iterations. The final tab holds all results from the optimization process in a form of a table. Each line is one solution. Upon double clicking one line the corresponding parameters are loaded into Grasshopper again. A direct export is not supported, however, copy&paste into any text or spreadsheet program works.

Wallacei

Wallacei [Wal] is a MOO plugin with a goal of providing the user with one user interface in which the algorithm, analysis of simulations and selection of optimal solutions can be done. At the time of writing, Wallacei only supported the NSGA-II algorithm. However, it provides a wide range of different visualization methods, including parallel coordinate plot, objective space visualization using cubes (like in Octopus), standard deviation

graphs, spider diagrams and trendlines, to help designers in analyzing the results (see Figure 2.8. Furthermore, when a mesh, i.e. phenotype, from the parametric model is linked to the optimization component, it is possible to display all resulting meshes from the simulation runs side by side in Rhino. This enables users to directly compare the resulting designs rather than objective values.

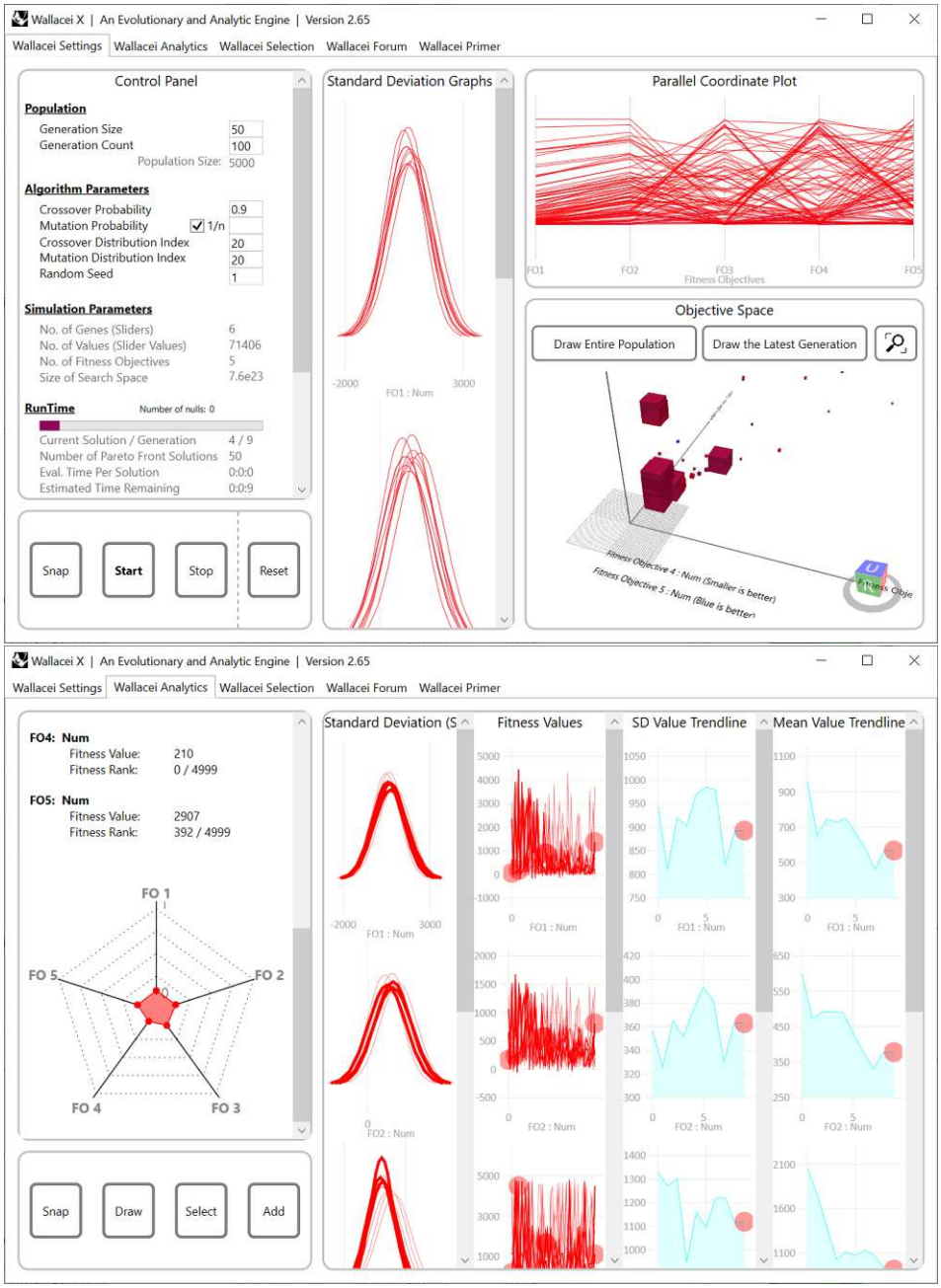


Figure 2.8: Graphical user interface of Wallacei.

CHAPTER 3

Optimization in Industrial Building

The BIMFlexi framework consists of two parametric models that are interconnected [RWSK+22]: a model for automated generation of optimized production layouts and a model for structural layout generation including the calculation of objectives. Each of the models are implemented within a Grasshopper3D for Rhino script. An Excel file is connecting the data flow between the two scripts. The optimization tool developed in this thesis is build on top of the structural layout generation script. Figure 3.1 shows the connections between the three parts. We outline the important parts of the two models which are preceding steps to the structural optimization and describe the optimization part in more detail as this is the main focus of the thesis.

3.1 Production Layout Generation and Optimization

In this section the important parts of the production layout framework are briefly described. A more detailed explanation can be found in the work of Reisinger et. al. [RZK+22] and Zahlbruckner [Zah21].

The production layout framework automatically generates multiple production layouts and optimizes them using an evolutionary algorithm. An example of such layout is shown in Figure 3.2. It takes multiple inputs, which are listed in Table 3.1. The production cubes describe the geometrical properties of the production layout while the lean-factor matrix and the transport intensity matrix describe the relational properties, where each cell of the matrix represents a relation between two production cubes. The lean-factor matrix is discrete and contains information on the neighborhood of a production cube. Four states are possible: absolutely necessary, important and core, unimportant/indifferent, and undesirable. As the name suggests, "absolutely necessary" means that two cubes

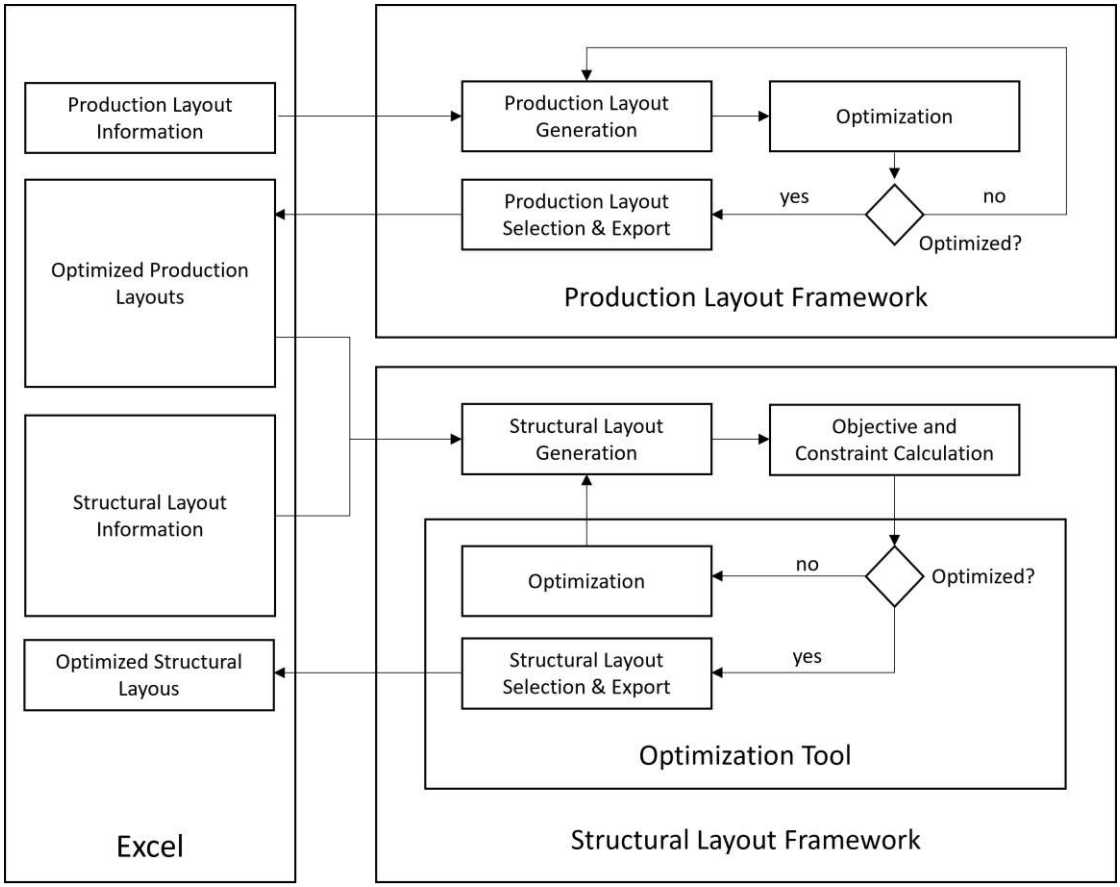


Figure 3.1: Data flow and workflow between the three parts, production layout model, structural layout model and the optimization tool, of the BIMFlexi framework

should be next to each other, "important and core" indicate that two cubes should be at least close to each other, e.g. 0 or 1 cubes away, "unimportant/indifferent" means that the placing of cubes does not matter and "undesired" means that two cubes should not touch each other. The transport intensity describes the material flow between rooms. For each pair of cubes the transport intensity is 0 or greater.

Constraints and objectives for production layout generation are listed in Table 3.2 and Table 3.3. The evolutionary algorithm uses a weighted sum approach to optimize all objectives simultaneously. For the optimization all objectives are reformulated into minimization problems. Feasible layouts are preferred over infeasible ones by the algorithm and fitter layouts, i.e. smaller fitness values, are ranked higher. The resulting production layouts are therefore ranked by the number of constraint violation in ascending order first and then by ascending fitness values. The best four ranked layouts are exported into an Excel file in order for the structural layout framework to read in. Users can also manually manipulate the production layouts in the Excel file. Within the structural

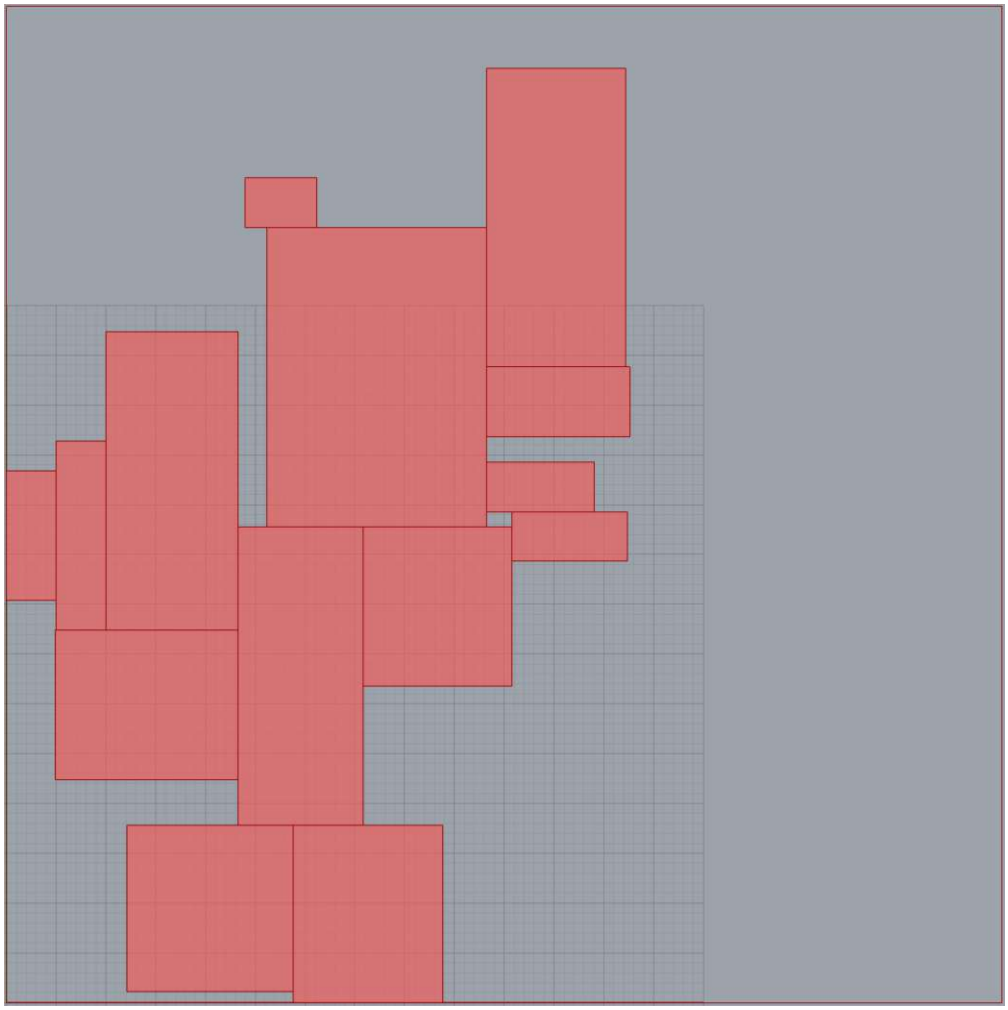


Figure 3.2: Production layout generated by the production layout generation and optimization framework.

Input	**Description**
Property size	Size of the property in x and y dimension or in m^2
Production cubes	A list of rooms with minimum x and y dimension as well as minimum area.
Lean-factor matrix	A matrix of room neighborhood dependencies.
Transport intensity matrix	A matrix of transport intensities between each pair of rooms.

Table 3.1: List of decision variables

Constraints	**Description**
Cohesive layout	All production cubes should be connected, i.e. at least 1/3 of the shorter side should be touching another production cube.
Building area boundary	The production layout should be contained by the property.
Adhere to lean-factor *absolutely necessary*	Neighborhood dependencies marked as absolutely necessary should be touching each other.
Adhere to lean-factor *undesirable*	Neighborhood dependencies marked as undesirable should not touch each other.
Adhere to minimum dimensions and area	Dimensions and area of each production cube should be at least as great as the minimum dimensions and area given by the input.

Table 3.2: Constraints for production layout generation

Objectives	**Description**
Free building area	The area covered by the production layout should be minimized in order to have enough space for future expansion.
Dense layout	Maximize the layout density in order not to waste area between cubes.
Production cube ratio	The difference of the x-y ratio of cubes between the minimum dimension and the generated production cube should be minimal
Lean-factor neighborhood *important & core*	Maximize the number of cubes that adhere to the lean-factor neighborhood *important & core*
Minimize transport length	Minimize the distance between rooms with transport intensity higher than 0.

Table 3.3: Objectives for the optimization of production layouts

layout framework the production layouts are then used as one of the flexibility objectives where the objective is to maximize the number of different production layouts that one structural layout can support.

3.2 Structural Layout Generation and Objective Calculation

The script for the structural parametric model is written in Grasshopper3D for Rhino. For a set of inputs it generates structural layouts and calculates the objectives needed for the optimization. The script roughly consists of three main parts.

1. Preparation of input parameters: The script loads all necessary inputs from an excel file. The file includes information on e.g. production layouts, property size, loads, preferences of stakeholders. The input will be processed into multiple parameters for the model, some of which are also linked as decision variables to the many-objective optimization (MaOO).

2. Model generation and static analysis: After the parameters are set, the script generates a model and does static analysis on it. It further calculates all necessary values to check whether some constraints have been violated, e.g. stability.

3. Assessment: Finally, the framework calculates all objectives based on the generated model.

The most important parts of the framework for the MaOO to work are the decision variables, constraints and objectives. In total there are nine decision variables, seven constraints and eleven objectives. All variables are listed and briefly described in Table 3.4, 3.5 and 3.6. A more detailed explanation of the parametric model and of all the variables can be found in [Kno21] and [RKK21].

The decision space variables are discrete and are represented through a slider and a list. As in most optimization plugins, our optimization tool changes the slider, whose value acts as an index to the list, to generate a new solution.

Since the generation and evaluation of one structure with its objectives and constraints is computationally expensive, especially for larger structures, it is important not to waste computation time. Therefore, one modification has been made on the decision variables regarding decision space constraints, i.e. constraints that can be evaluated prior to optimization. We consider the constraints c_7 and c_8, which are limiting the building size. The given decision variables can be grouped into geometrical parameters and structural parameters. Geometrical parameters include the first four decision variables, which are the primary and secondary grid size and number of fields, while the remaining ones are structural parameters. The size of a building structure is mainly controlled by the

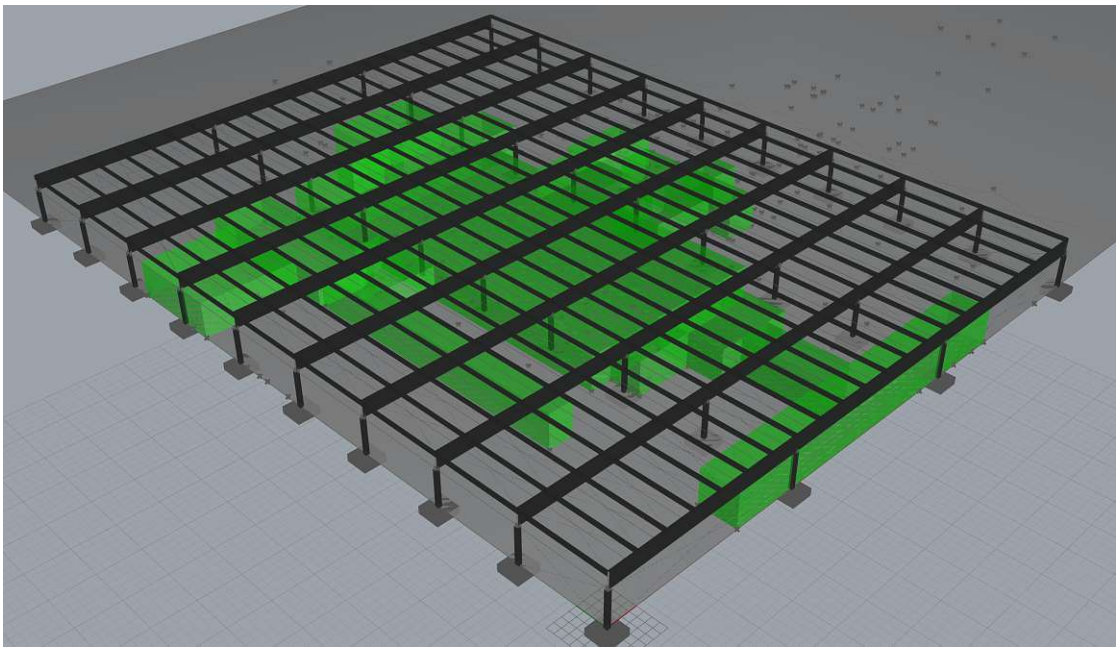


Figure 3.3: Parametric model. The load bearing structure is generated around one of the possible production layouts (colored in green).

	Decision variables	**Description**
d_1	Primary Axis Grid	Distance between two columns in meters (integer) in x direction.
d_2	Primary Axis Fields	Number of columns in x direction
d_3	Secondary Axis Grid	Distance between two columns in meters (integer) in y direction
d_4	Secondary Axis Fields	Number of columns in y direction
d_5	Primary Structure Type	Construction form and material of the primary load bearing structure
d_6	Secondary Structure Type	Construction form and material of the secondary load bearing structure
d_7	Column Type	Construction material of the columns
d_8	Bracing Type	Bracing system for walls and roof
d_9	Retrofitting	Retrofitting load for future retrofitting ability of the system.

Table 3.4: List of decision variables

	Constraints	**Description**
c_1	Maximum stress utilization	The utilization must be ≤ 1.
c_2	Maximum structural displacement	The displacement must be smaller than a predefined threshold.
c_3	Stability	The building must be stable.
c_4	Maximum secondary axis grid	Secondary axis grid must be smaller than the tertiary span width of the roof construction.
c_5	Maximum building height	The building height must be smaller than the maximum allowed height.
c_6	Feasible production layouts	Additional feasibility check of production layouts in case of manual manipulation.
c_7	Minimum building dimensions	The building must be large enough to contain at least one production layout.
c_8	Maximum building dimensions	The building must be smaller than the property.

Table 3.5: List of constraints

geometrical parameters. To avoid structures that are too small or too big all combinations of the four parameters have been filtered for feasible sizes and sorted by area.

Another decision space constraint is the user's preference. In case the user has any preferences concerning materials or specific geometrical properties prior to optimization, the user may limit those parameters through an excel file. This will further reduce the size of the search space. Any parameter that does not have more than one option due to user preference will be removed from the optimization automatically.

Constraints are represented through boolean components and are all treated as hard constraints, i.e. they are not allowed to be violated under any circumstances. If any of the constraints are violated the solution is infeasible.

Objectives are represented through double precision floating point number components. All objective values are greater than 0. Some objective values are already normalized such that the values are always ranging from 0 to 1, while other objectives do not have any bounds, e.g. costs. For the MaOEA all objectives need to be minimized. Therefore, any objective that are to be maximized are multiplied by -1. Further discussion on the objective calculation can be found in the paper from Reisinger et. al. [RWSK+22].

	Objectives	Description
o_1	Life Cycle Cost (LCC)	Minimize cost over 25, 50 or 80 years
o_2	Global Warming Potential (GWG)	Minimize Life-cycle environmental impact
o_3	Acidification Potential (AP)	Minimize acidification potential
o_4	Primary Energy Intensity (PEI)	Minimize primary energy use
o_5	PEI non-renewable (PEInt)	Minimize non-renewable primary energy use
o_6	Recycling Rate	Maximize recycling potential, e.g. use separable materials
o_7	Retrofittability	Maximize load-bearing capacity for future loads.
o_8	Expandability	Maximize expandability of the production area
o_9	Flexibility in space	Maximize building height reserve
o_{10}	Flexibility in floorplan	Minimize area that is occupied by columns, i.e. reduce the number of columns.
o_{11}	Production layouts	Maximize the number of different production layouts one structural layout can support.

Table 3.6: List of objectives

3.3 Many-Objective Optimization Tool

3.3.1 Algorithms

The optimization tool includes four algorithms: SPEA2, SPEA2+SDE, CTAEA and a simple enumeration algorithm to evaluate all possible combinations. The latter was added to have the possibility to evaluate the complete decision space if the decision space is small enough due to various constraints, e.g. property sizes, user preferences, etc. SPEA2 and the SDE adapted version both use the constraint dominance relation to deal with constraints. In CTAEA the density estimation uses unique weight vectors [LDZK14b]. For the generation of the weight vectors the number of divisions H was set to the same number as the objectives, which is 11. The archive size for each algorithm was set to the same size as the population size. Simulated binary crossover [DA$^+$95] and polynomial mutation [DG$^+$96] is used as the crossover and mutation operator. The distribution index, which is used to control the distribution of the offspring solution, were set to 2 for both, crossover and mutation. The low index will generate solutions that

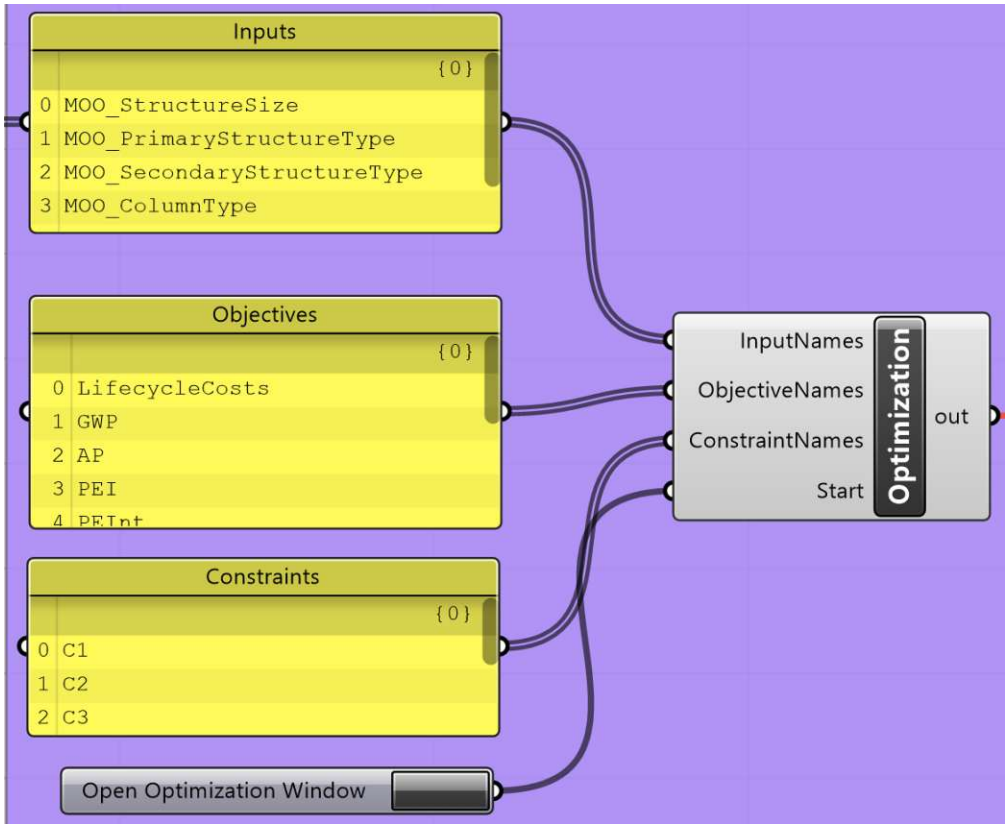


Figure 3.4: C# optimization component. Decision variables, objectives and constraints are linked via names

are further spread from the parents in order to have a higher diversity. On the other hand, a higher number indicates a lower distribution. The generated solution will have parameters closer to the parents.

The choice of algorithms covers multiple scenarios and lets us compare different characteristics of the algorithms at the same time. SPEA2 and SPEA2+SDE are in general the same algorithm but with different density estimators. In theory, for MaOO SPEA2+SDE should perform better if the objectives are not highly correlated. On the other hand SPEA2 may be used for lower number of objectives, i.e. the decision maker may not be interested in all objectives in some cases. CTAEA handles constraints differently than the other two algorithms. If the objective space is highly constrained CTAEA may perform better than SPEA2 and SPEA2+SDE. This may be the case for very large grid sizes.

3.3.2 Linking to the Framework

The MaOO component is a C# script in Grasshopper (Figure 3.4) and is in the same Grasshopper script as the parametric model for the structural layout. Because of the DAG the optimization components need to be linked externally. Most of the optimization

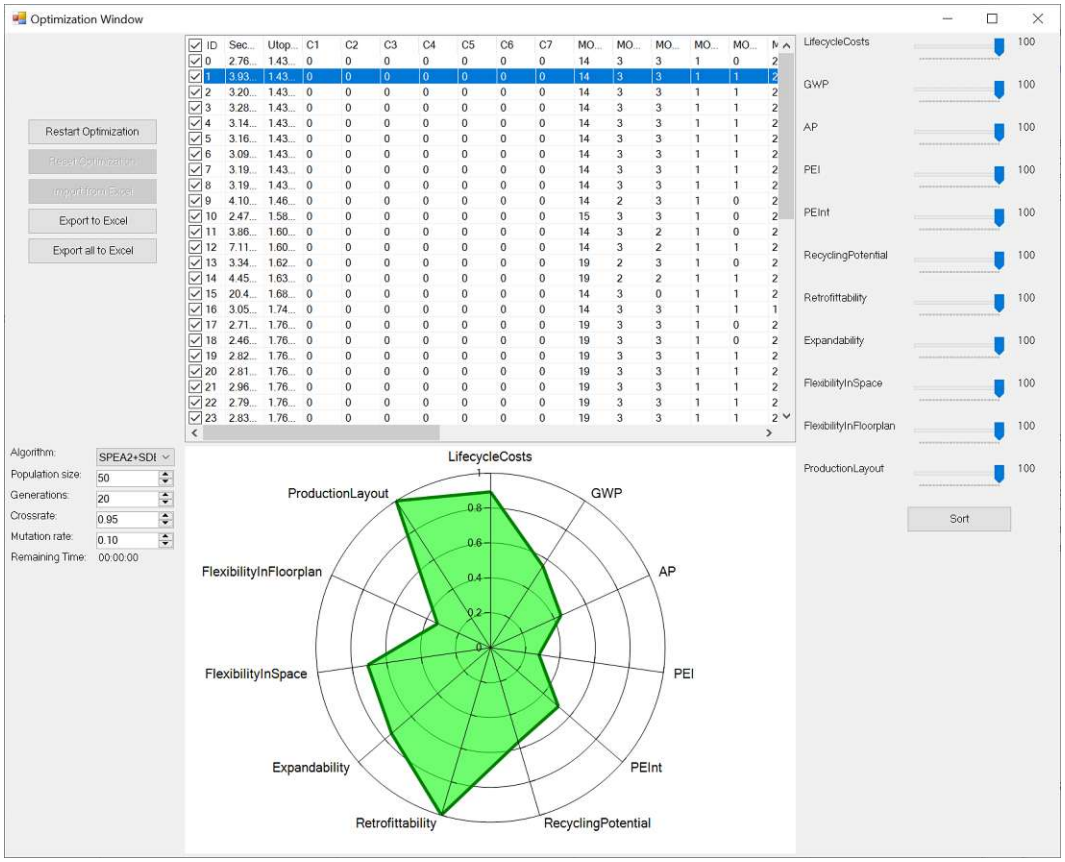


Figure 3.5: Optimization window of the tool.

plugins offer the ability to connect the sliders and objectives via a custom written wire. For this thesis the wires are replaced by the component names. Adding or removing decision variables, objectives and/or constraints can therefore be done by simply adding or removing the names. This setup also allows the automatic addition or removal of parameters, e.g. in case stakeholders are not interested in them. The MaOO component itself has four inputs: The names of the decision variable sliders, objective and constraint components as well as a button to open the optimization window (see Figure 3.5). Upon opening the optimization window the script reads in the names of the components and tries to find the corresponding components in the Grasshopper document. To generate a solution the algorithms change the sliders and issue a recalculation in Grasshopper. After recalculation, values in the objective and constraint components can be read and used for further steps of the chosen algorithm.

3.3.3 User Interface and Solution Visualization

Within the optimization window one of the four algorithms can be selected for the optimization. Except for the complete enumeration method, the number of generations,

population size, crossover and mutation rate can be set. Per default they are set at 50, 20, 0.95 and 0.10 respectively.

The optimization can be stopped at any time. When the optimization is stopped, the current Pareto front will be listed. Each line contains the input parameters for the framework, objective values and constraint violations. The actual model of the structure is not saved. To view the model the solution needs to be calculated again such that Grasshopper visualizes the model. As optimization may take very long, depending on the structure, the MaOO component gives an estimated finishing time as well. Furthermore, one single solution or multiple solutions can be selected from the list to be exported into an Excel file. For analysis purposes, all evaluated solutions from one search are currently saved and can all be exported to Excel as well.

Per default the list is sorted by the normalized euclidean distance to the ideal point. For normalization the nadir and ideal points were used. As we are dealing with trade-off solutions, weights have been added for post sorting. Since the sorting happens after optimization, the decision maker can sort the Pareto front with different weights on the objective to target specific trade-off solutions without the need to restart the optimization process. Solutions from previous runs can also be imported for further analysis.

Additionally to the list representation, solutions are visualized using a radar diagram. Since we do not know the objective values' range, the radar diagram shows the relative performance of one solution compared to the approximated Pareto front. More specifically, we use the nadir point and the ideal point to normalize the objective values into the range from 0 to 1, where 0 means worst and 1 best performing objective in the current Pareto set. For example the radar diagram in Figure 3.5 shows that the solution supports all provided production layouts and performs well in terms of lifecycle costs and retrofittability, however, is only doing moderate in terms of GWG, AP, PEI, PEInt and Recycling Potential compared to all the other solutions in the Pareto set.

3.3.4 Implementation Details

The whole optimization tool is written in C#. Figure 3.6 shows a class diagram of the most relevant parts of the tool.

- OptimizationForm: This class inherits from *System.Windows.Forms*. It displays data collected from optimization and contains event handlers to handle user inputs. It is also the class holding all information.
- GrasshopperCommunicator: As the name suggests this class handles communications with the Grasshopper3D script for calculating the structural layout. Data are read in and written out through an instance of this class.
- OptimizationAlgorithms: All algorithms derive from this abstract class. It is an interface for the OptimizationForm to start the optimization process and retrieve results from the optimization.

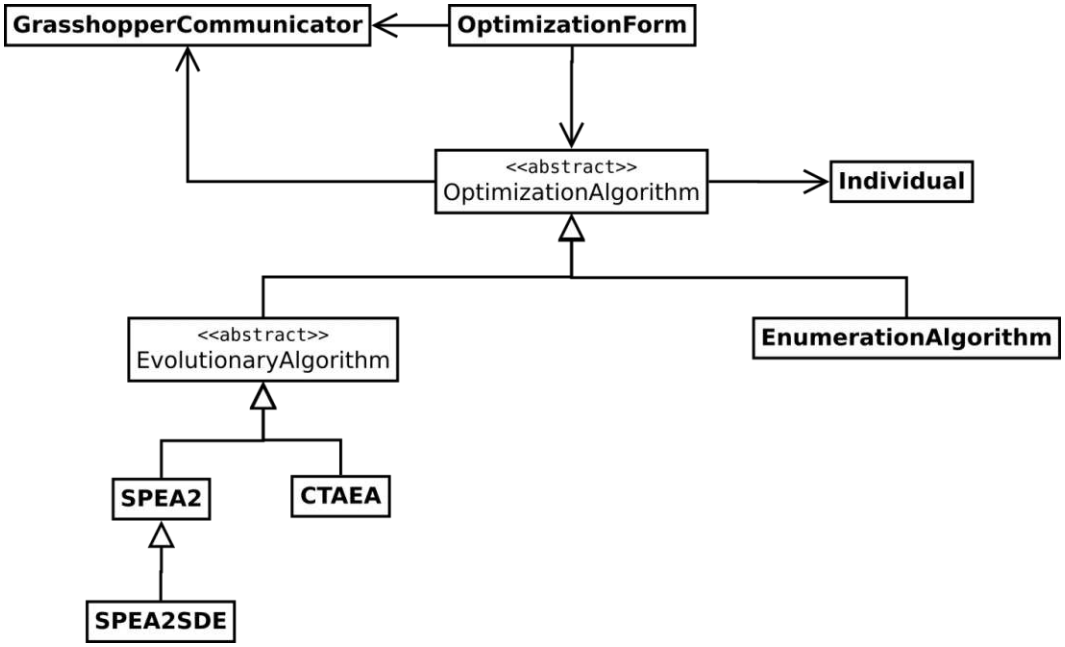


Figure 3.6: Class diagram of the optimization tool highlighting the most important parts and associations.

- EvolutionaryAlgorithm: This abstract class inherits from OptimizationAlgorithm and is the base class for all evolutionary algorithms. It contains variables that are only relevant for evolutionary algorithms, e.g. population size, number of generations.
- EnumerationAlgorithm, SPEA2, SPEA2SDE, CTAEA: These classes implement the four different optimization algorithms. The EnumerationAlgorithm class will enumerate over all possible combinations of the structure, while the other three classes implement the SPEA2, SPEA2+SDE and CTAEA algorithms respectively.
- Individual: An instances of this class represents one solution. It contains the inputs for the structural parametric model, objective values and constraint violations. The other three classes operate on instances of this class.

When a user opens the optimization window the C# script will create an instance of OptimizationForm which takes the Grasshopper document and the names of the decision variable, constraint and objective components linked to the C# component. The Grasshopper document is the script in which the C# component is contained. It therefore has access to all components that are within the structural layout script. To prevent Grasshopper from reinstantiating the OptimizationForm class when input updates to the C# component occur, i.e. releasing the "Open Optimization Window" button, the OptimizationForm object is saved in a static variable. This object holds the whole state of the optimization tool.

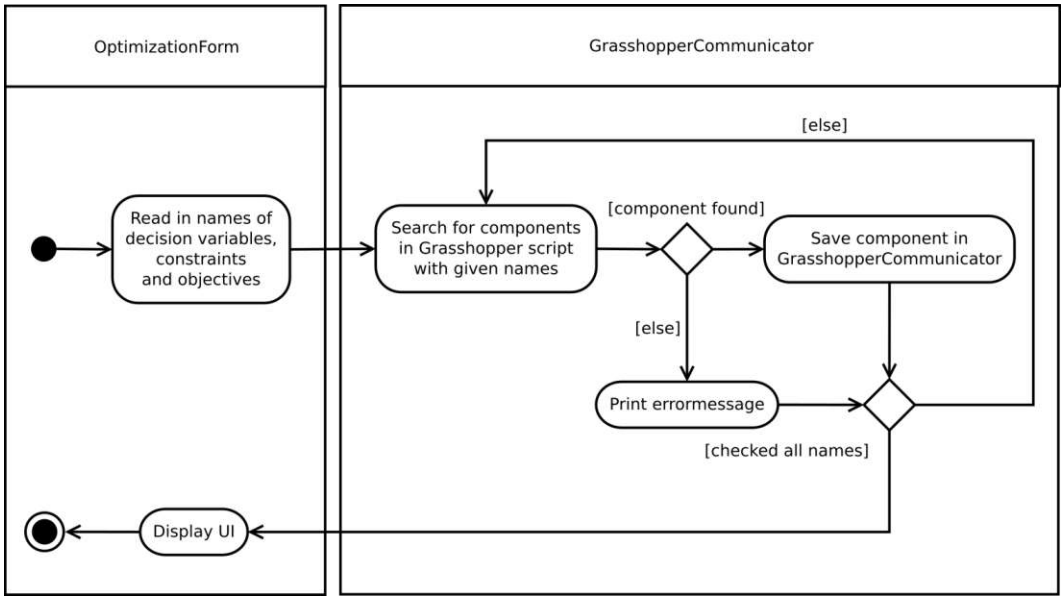


Figure 3.7: Activities when opening the optimization window.

The OptimizationForm proceeds to instantiate a GrasshopperCommunicator object which will take the component names. The GrasshopperCommunicator searches in the Grasshopper document for the components with the given names and will save the reference to those components for later use. After that, the OptimizationForm shows the user interface. The whole process is visualized in an activity diagram in Figure 3.7.

The optimization process is depicted in Figure 3.8. When the user starts the optimization, the OptimiztionForm will read in the input from the text fields. It creates an instance of the correct OptimizationAlgorithm with the given inputs. It also receives the GrasshopperCommunicator instance for solution calculation. The algorithm randomly generates an initial population and proceeds to optimize the population. Details about the specific algorithm can be found in the original papers [ZLT01], [LYL13], [LCFY18]. The optimization stops when the number of generations is reached or when the user stops the optimization. In both cases the latest solution set, i.e. Pareto front, is taken and its ideal and nadir point is calculated as well as the distances of the solutions to the ideal point. The OptimizationForm then takes the solution set and sorts it by distance to ideal point and finally displays the results to the user.

In order to get the objective values and the constraints of one structural layout the optimization tool needs to initiate the solution calculation from the Grasshopper script. This process is shown in Figure 3.9. The OptimizationAlgorithm first generates the inputs for the individual. For the initial population the input values are random. For the remaining generations the inputs are obtained by crossover and mutation. The GrasshopperCommunicator takes the individual and sets the sliders of the Grasshopper document to the inputs of the individual and initiates a recalculation of the Grasshopper script.

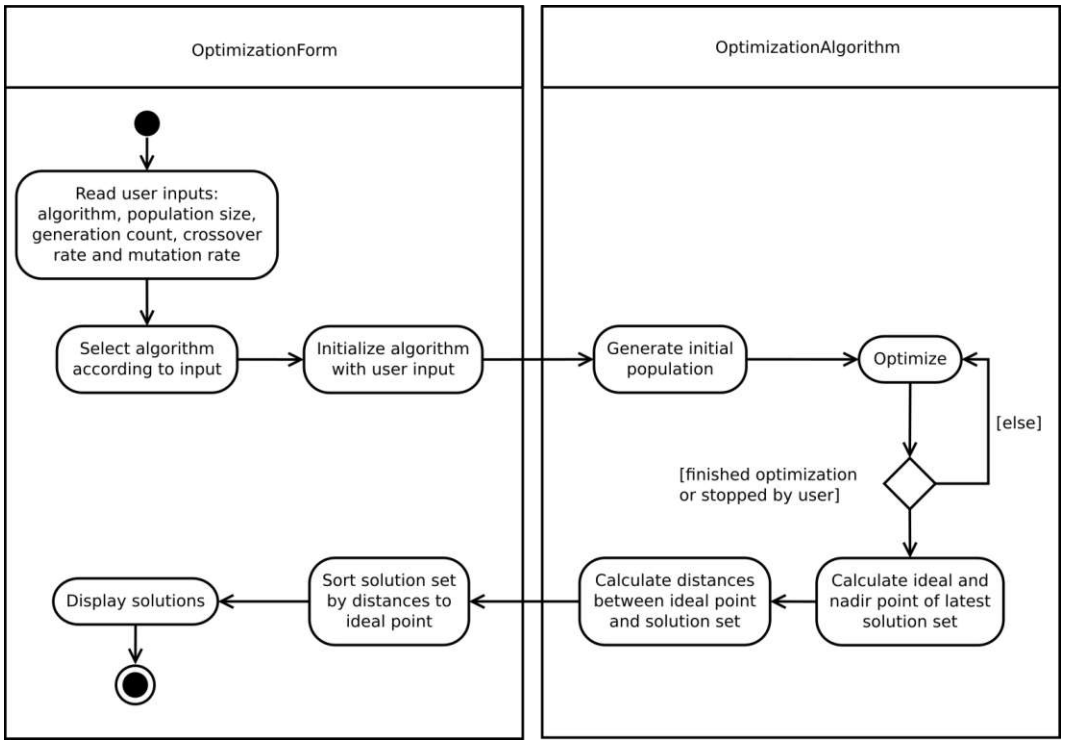


Figure 3.8: Activities when starting an optimization.

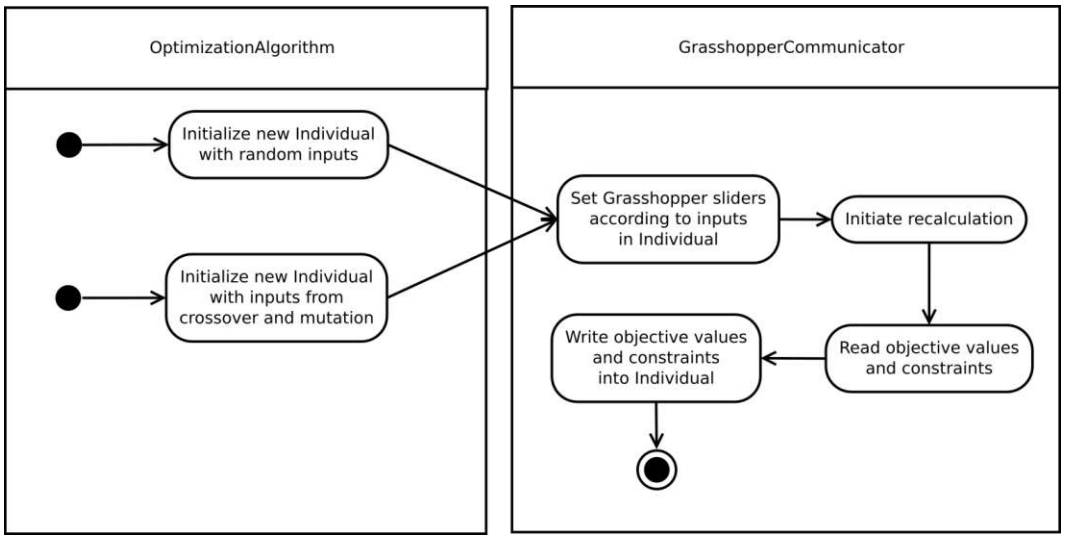


Figure 3.9: Activities when calculating a structural layout

When the recalculation is finished it reads the values from the objective and constraint components and saves the results back into the individual. The OptimizationAlgorithm then proceeds with the optimization process.

CHAPTER 4

Evaluation and Results

The evaluation of the optimization tool consists of two parts, each trying to answer one of the research questions stated in the Introduction (Chapter 1). Both parts were conducted independently and will be presented separately in this chapter. The first part investigates the usability of the tool and users' acceptance of the automatically generated structural layouts. In this part the use of the optimization tool is compared against the manual use and results are obtained through questionnaires and notes on selected or preferred structures from users. In the second part we will look at the algorithmic side of the tool. The three implemented algorithms SPEA2, SPEA2+SDE and CTAEA are compared in terms of convergence and diversity of the obtained Pareto fronts. The evaluation is conducted with three different test cases provided by the civil engineers from the BIMFlexi project. The goal of this part of the evaluation is to find out which algorithm, if any, works best for our use case. Finally, time analysis on the optimization process is given as well.

4.1 User Study

A user study was conducted to compare the manual use of the parametric optimization and decision support tool (method M) to the automated, i.e. no manual adjustment of parameters, optimization tool (method A) [ZRWS+22] [RWSK+22]. The goal was to investigate whether users would prefer the generated designs from the automated method over the designs that were generated manually.

In total 36 students from TU Wien have volunteered to take part in the study. Twenty-three participants had a background in architecture and thirteen in engineering (eight civil engineers, three spatial planners and one mechanical engineer). Qualitative and quantitative feedback was collected using questionnaires (Appendix A: 6). There were thirty-three closed and three open questions. The closed questions were formulated as a 5-point Likert scale questions. With exception to one question, all questions were

positively formulated where 5 means the most positive response. The rating of the inverted question was adjusted for the evaluation. Users have been asked about their satisfaction with the presented technology, using the technology acceptance model (TAM) by Davis (1989), and their satisfaction on people-process-level for both methods. More precisely the questionnaire contains questions about six different sub-categories: ease of use, usefulness, visualization, process satisfaction, outcome satisfaction and collaboration satisfaction. Additionally, data have been collected on the selected or preferred structural layouts. The students had to fill out a layout sheet that contained information on the selected parameters for the structural layout and choose one final layout from either method M or method A.

4.1.1 Experimental Setup

The user study was conducted at an office at TU Wien. All participants had a tutorial on how to use the framework and the optimization tool prior to the study. Since the optimization takes a while it was run beforehand using the default settings (algorithm: CTAEA, population: 50, generation: 20, crossover: 0.95, mutation: 0.1) and results were imported for the study. The participants tested the tools in groups of two. All groups tested both methods. Half of the groups started with method A and the other half with method M to make the results more comparable in case there is a learning effect between the two methods. The same production layouts were used for both methods.

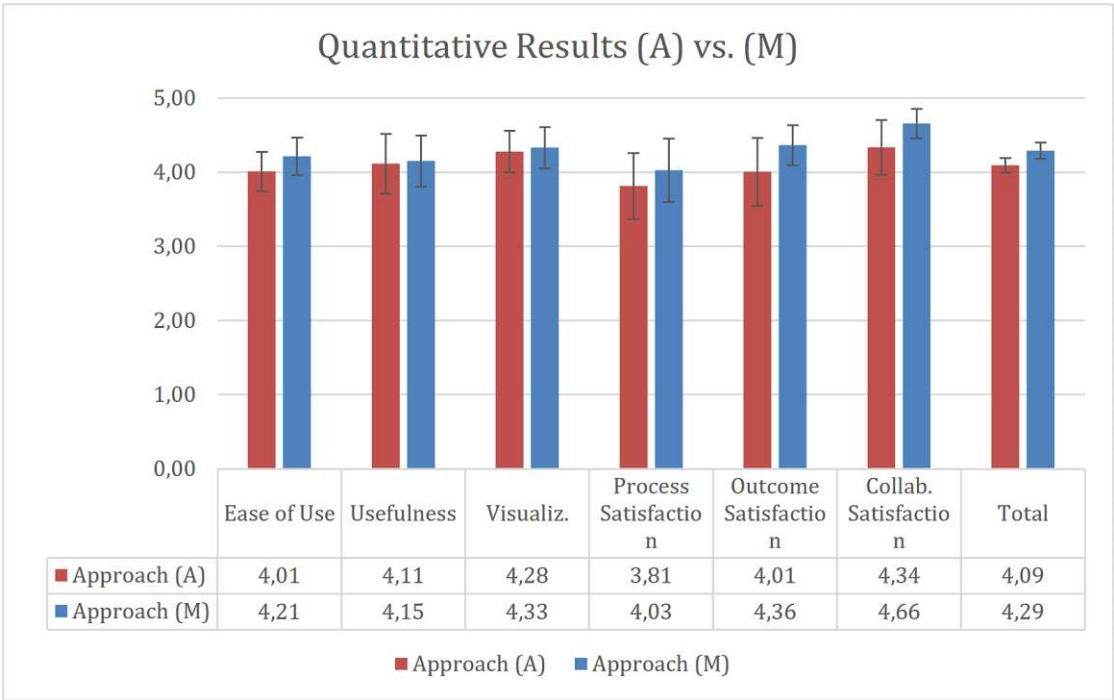
The participants were given the assignment to design a structural building around already generated production layouts. Before starting the exercise the groups were instructed to take different roles during the discussion of selections, e.g. architect, civil engineer, owner, to represent different stakeholders. Furthermore, each group received a layout sheet where they could write down the parameters and objective values of their layout choices for method A and the generated layouts for method M.

The groups were asked to choose or generate at least three layouts for each method. Then, for each method they had to decide on one layout from the multiple they generated. In case of method M participants had to discuss different building variants they were interested in and decide on parameters first before manually setting them. Participants testing method A would look and click through the variants that the optimization showed. For both methods all parameters of the generated structural layouts were noted.

There was around 30 minutes time to do the testing and 15 minutes to fill out a questionnaire for each method. At the end the group had to decide between the one layout from method A and the one layout from method M.

4.1.2 Results

Results presented in this section are based on 72 questionnaire responses of participants, 36 for each method, and 18 layout sheets (one from each group). Figure 4.1 shows the mean score in every sub-category for each method. The evaluation reveals that the users were overall satisfied with both methods. Except for the process satisfaction with method



	Ease of Use	Usefulness	Visualiz.	Process Satisfaction	Outcome Satisfaction	Collab. Satisfaction	Total
Approach (A)	4,01	4,11	4,28	3,81	4,01	4,34	4,09
Approach (M)	4,21	4,15	4,33	4,03	4,36	4,66	4,29

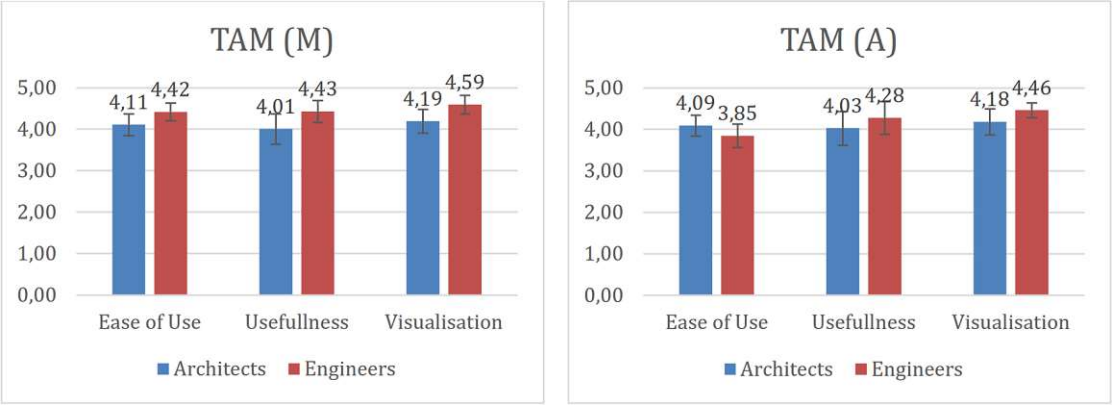
Figure 4.1: Evaluation of the manual (M) and automatic (A) method. From [ZRWS+22].

A (avg. 3.81), each category had an average rating higher than four. For both methods collaboration satisfaction was rated the highest (method A: 4.34, method M: 4.66).

Comparing the manual and the automated method, results show that the manual approach was rated higher in every sub-category. While the differences in technology satisfaction was rather small (differences: ease of use: 0.2, usefulness: 0.04, visualization: 0.05), participants were less satisfied on the people-process-level (differences: process: 0.22, outcome: 0.25, collaboration: 0.32).

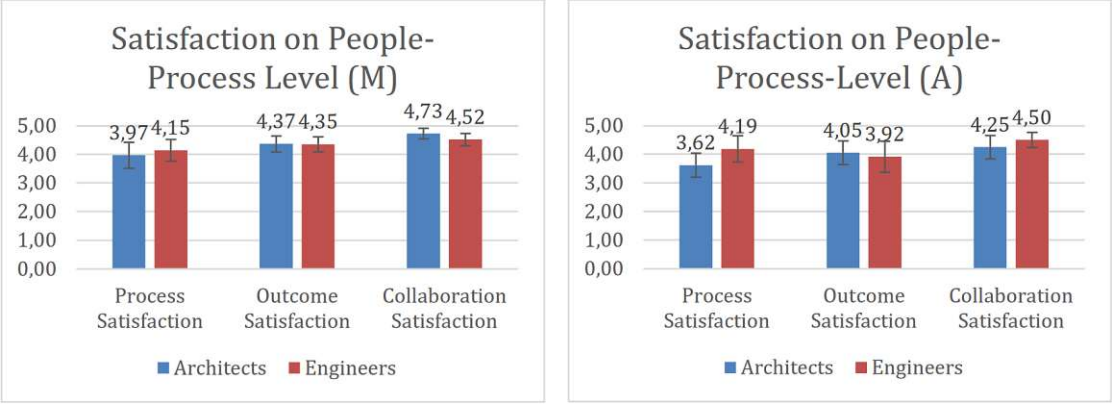
Further evaluation was done on different disciplines. Figure 4.2 shows the satisfaction with technology for each method, separated by discipline. Overall, engineers were more satisfied with the technology of both methods than architects. One exception to this is the ease of use for the automated version, were architects gave a higher score (architects: 4.09, engineers: 3.85). Architects seem to not have a preference for either method. The difference in scores between both methods are marginal (differences are less than 0.05 for all subcategories). On the other hand, engineers were more satisfied with the manual approach (differences: ease of use: 0.57, usefulness: 0.15, visualization: 0.13).

Comparison on people-process-level for the two disciplines is shown in Figure 4.3. Engineers rated the process and collaboration satisfaction for both methods fairly similar (differences 0.04 and 0.02 respectively) while there is a clear preference of the manual method by the architects (differences 0.35 and 0.47). Both groups were more satisfied with the outcome from the manual approach.



(a) manual (b) automated

Figure 4.2: Results for technology satisfaction of the manual (a) and automatic (b) method separated by discipline. From [ZRWS+22].



(a) manual (b) automated

Figure 4.3: Results for satisfaction on people-process-level of the manual (a) and automatic (b) method separated by discipline. From [ZRWS+22].

Concerning the choice of the final structural layout, ten groups selected a layout from method M and eight groups selected a layout from method A. Though there seems to be a slight preference for method M, the one-sample chi-square test shows that the difference is not statistically significant (chi-squared=0.22 and p=0.64). Five groups out of nine that started out with method M selected an automatically generated layout as their final choice. Whereas only three groups from the nine groups starting with method A chose an automatically generated layout. Results are summarized in Table 4.1

The structural layouts were further compared by their objectives and results are listed in Table 4.2. From the eighteen layout sheets, fourteen were trade-off choices, i.e. there was no layout that was better than the other layout in all objectives. The choice of the final

	Chose method M	**Chose method A**
Started with method M	4	5
Started with method A	6	3
Total	10	8

Table 4.1: Final structural layout choices

Best layout	**Chose method M**	**Chose method A**	**Total**
Trade-Off	9	5	14
Layout M	1	1	2
Layout A	0	2	2

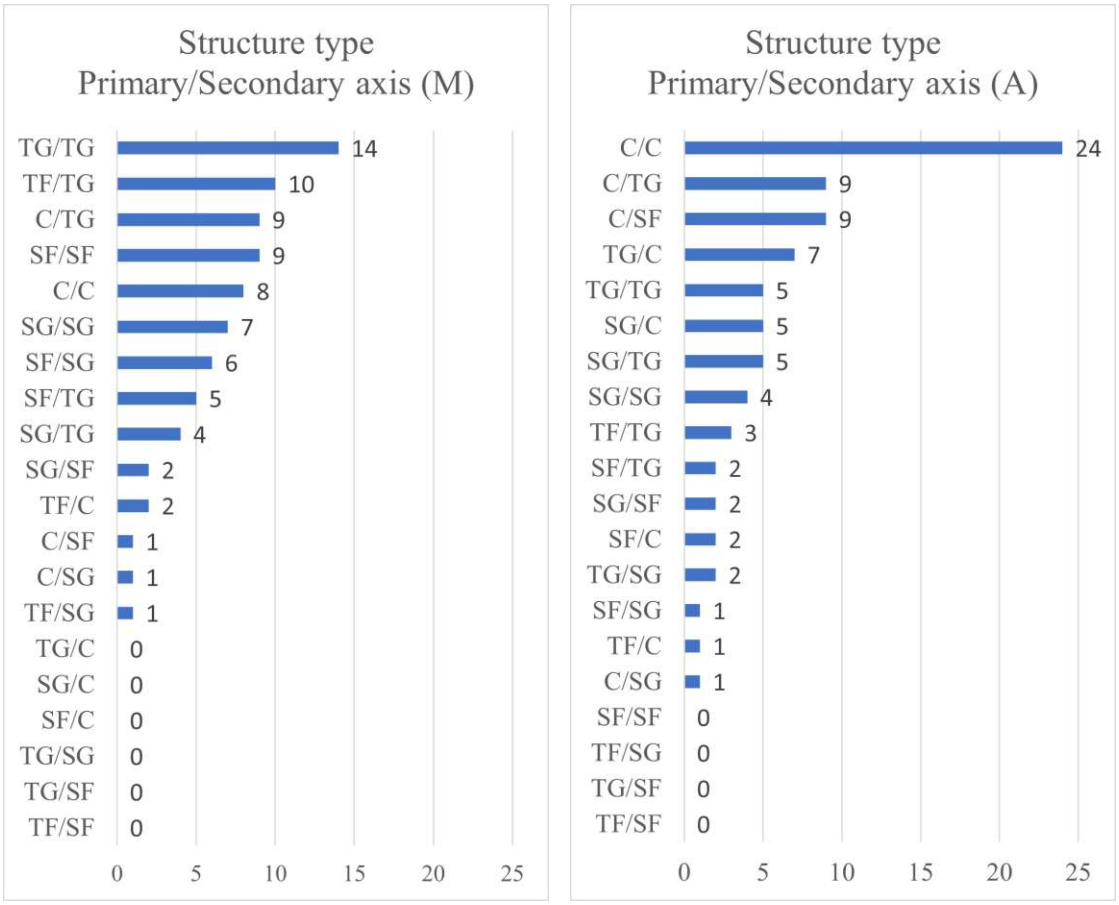
Table 4.2: Final structural layout choices considering objective values.

layout was therefore a preference question, e.g. the participants preferred lower cost over more flexibility. Two groups had an automatically generated layout that was better than their favorite manual layout. In both cases they chose the layout from the optimization tool. On the other hand, two groups had a manually generated layout that was better than the final one from method A. One of the group chose the manual one, while the other group selected the layout from the optimization tool.

A clear difference can be seen in the choice of the structure types. Figure 4.4 lists the combinations of primary and secondary structure type of all layouts that were recorded by the participants and Table 4.3 shows the number of layouts that had in both directions the same material. When participants manually generated their layouts there is a clear preference for timber girders (TG) for both directions, while T-precast concrete beams (C) were by far the most selected one from method A. Another observation is that during manual generation participants very often seem to select the same material for both directions, e.g. timber/timber or steel/steel. This is not the case using the optimization tool.

Material	**Method M**	**Method A**
Timber	28	8
Steel	25	3
Concrete	8	24

Table 4.3: Number of structural layouts that had the same material for primary and secondary structure type



(a) manual (b) automated

Figure 4.4: Structure type selection for manually generated layouts (a) and automatically generated layouts (b). TG=Timber Girder, TF=Timber Framework, C=Concrete T-Beam, SF=Steel Framework, SG=Steel Girder

Finally, in the questionnaire the participants were asked for potential improvements of the tool. One third of the participants stated that the calculations of the objectives were not transparent for both methods. For method M users found that it would be helpful if it was possible to store multiple variants during the search to better compare them. Participants wished for a more refined user interface with a detailed description of the MOO outputs. Related to this the students asked for a better visualization or indication of the variant's material, as they understand names or colors while the tool would only show numbers.

Concluding, the results of this study reveal that the optimization tool was well accepted and users were satisfied with its use. However, users preferred the manual method slightly more. Therefore, there is much more room for improvements for the optimization tool.

4.2 Algorithm Comparison

The three implemented algorithms, SPEA2, SPEA2+SDE (ongoing referred to as SDE) and CTAEA have been compared in their performance regarding convergence and diversity using the hypervolume (HV) [ZT98] and inverted generational distance plus (IGD+) metric [IMTN15]. The hypervolume refers to the volume in n-dimensional space (n > 3). To calculate the hypervolume of a Pareto front a reference point $r = (r_1, r_2, ..., r_m)$, were m is the number of objectives, is needed to span a hypervolume between each solution in the Pareto front and r. The union of these individual hypervolumes is the hypervolume of the Pareto front. r is a parameter that needs to be manually set. In our evaluation we set $r = (1.1, 1.1, ..., 1.1)$ to better reflect the diversity aspect of the normalized Pareto front. The closer a solution set is to the approximated Pareto front and the more wide spread the solutions in the set are the higher the hypervolume is. A higher hypervolume value is therefore a better result. On the other hand the IGD+ metric measures the distance from a set to the approximated Pareto front. A lower value is therefore desirable. Both metrics were calculated using the pymoo library [BD20]. Furthermore, the number of feasible and unique solutions in each solution set, i.e. Pareto front obtained by each algorithm, is compared between each algorithm and approximated Pareto front.

4.2.1 Experimental Setup

Three different example facilities with different search space sizes have been used to test the algorithms. The search space size is mainly dependent on property size and

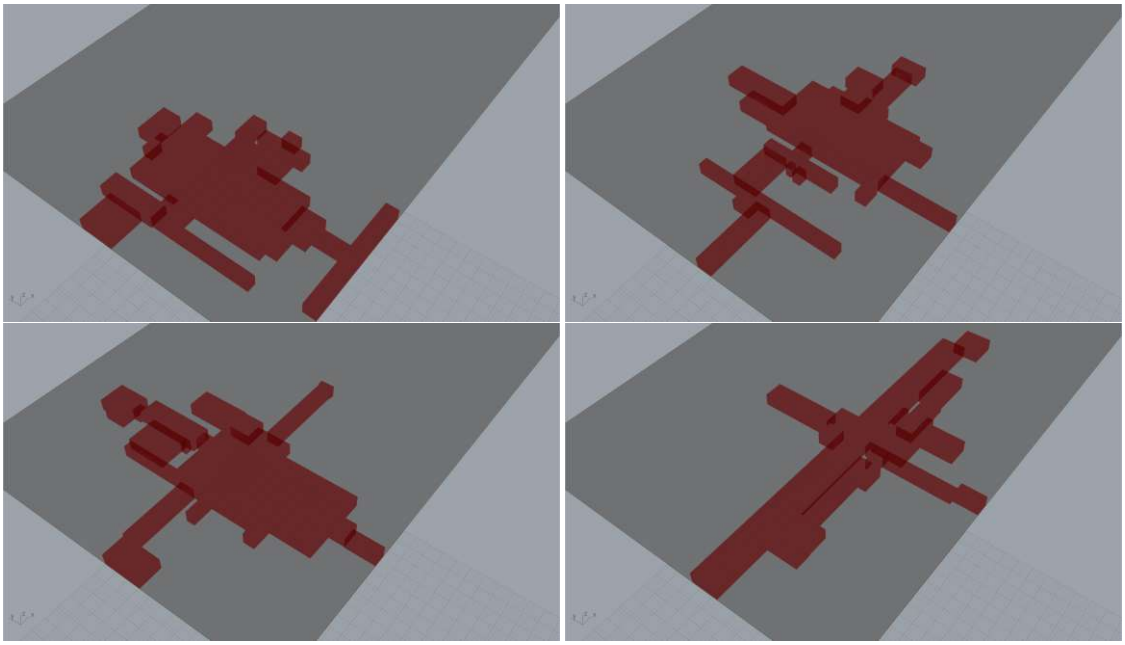


Figure 4.5: Production layouts for test case 1

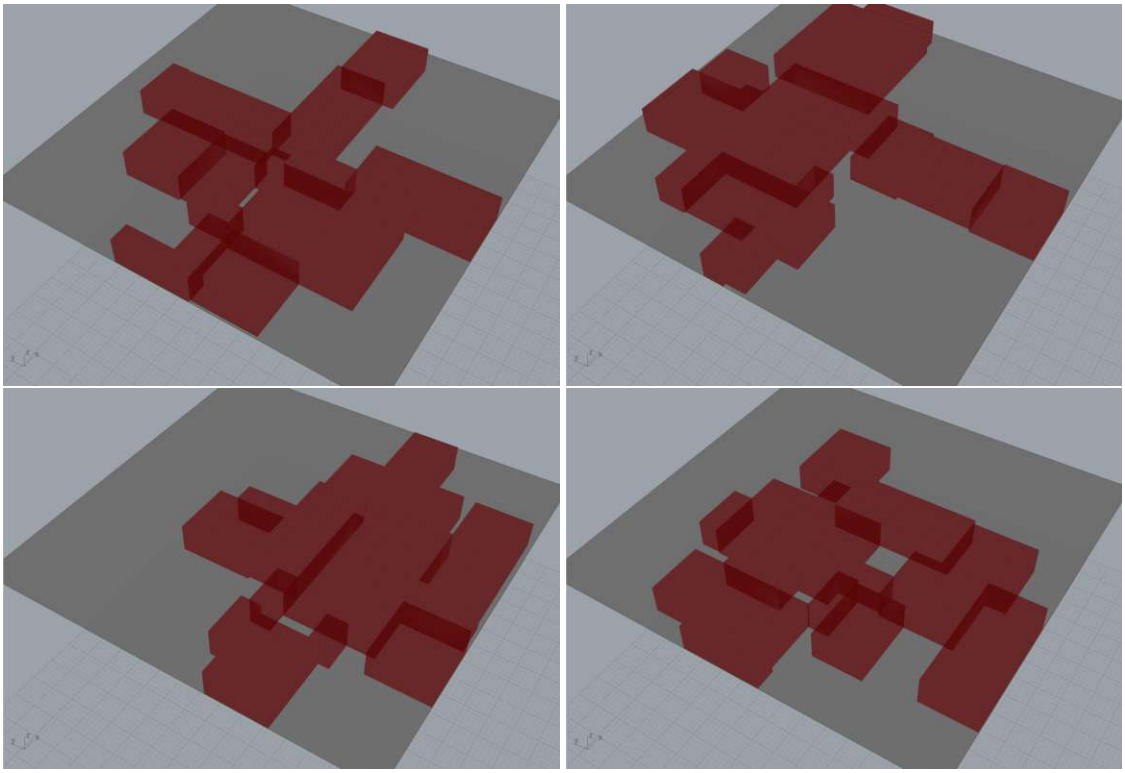


Figure 4.6: Production layouts for test case 2

production layout sizes. Each test case includes four different production layouts that were generated automatically in a preceding step [Zah21].

Test case 1 is a food and hygiene facility. The property size is 37500 m^2. The four production layouts have an area of around 3200m^2, 3110m^2, 3500m^2 and 3380m^2 respectively (see Figure 4.5). For test case 1 there are around 128 000 possible input combinations.

The second test case is a metal processing facility with a property size of 10000m^2. Three production layouts have an area of around 3250m^2 and one has an area of 3800m^2 (see Figure 4.6). This test case has around 20000 possible input combinations.

The last test case is a food and hygiene facility again with 7125m^2. The production layouts have an area of around 3160m^2, 3280m^2, 3330m^2 and 3100m^2 (see Figure 4.7). Test case 3 has around 6000 possible parameter combinations.

For each test case and algorithm the same optimization parameters were used: population size = 50, number of generations = 20, crossover rate = 0.95 and mutation rate = 0.10. The input values for the parametric model like structure type or axis grid sizes were not restricted through all the test runs. Table 4.4 lists the decision variable inputs that were considered.

Five runs for each algorithm and test case were done. During each run the resulting

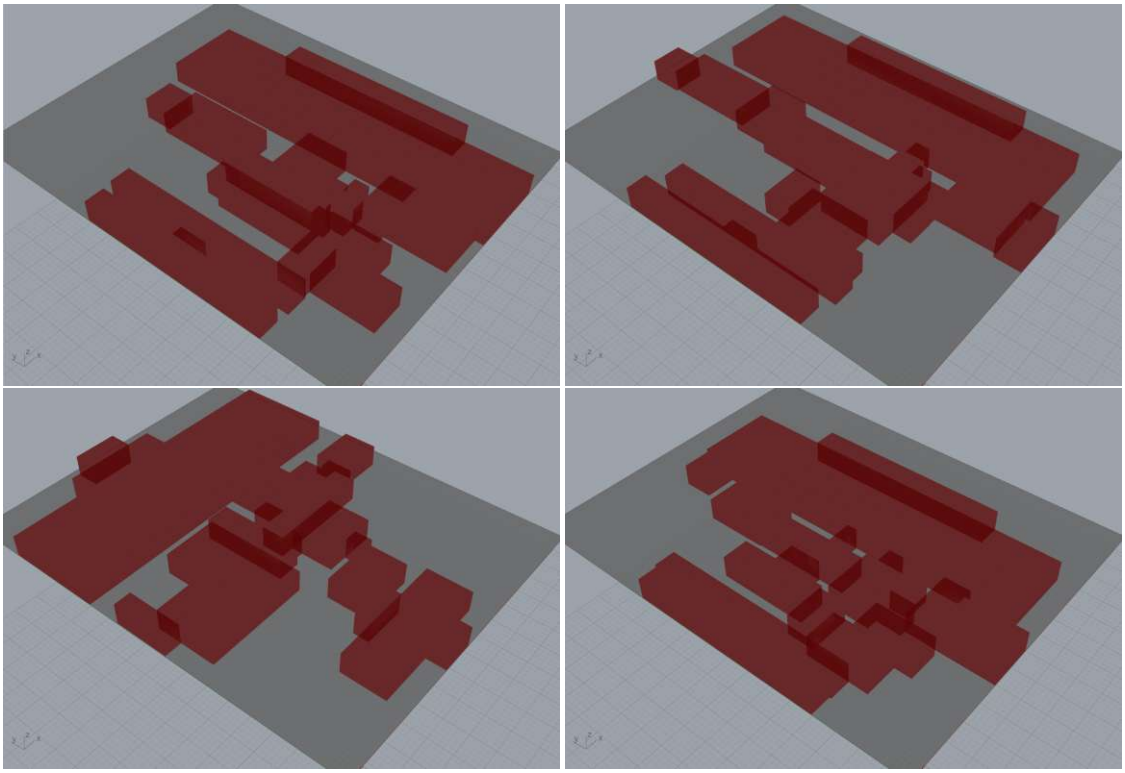


Figure 4.7: Production layouts for test case 3

Pareto front containing 50 solutions has been saved. For evaluation purposes we also logged all 1050 solutions that were generated during the search (1000 solutions due to the population size and the number of generations, and the additional 50 to initialize the archives).

To calculate the IGD+ metrics an approximated Pareto front is needed. Usually this is calculated by joining the Pareto front from each run. However, since more solutions have been logged during search we approximate the Pareto front using all solutions that were calculated during all runs for each test case. This yields a better approximation as it is less restricted by the algorithms. For better readability, going forward we will use the term "approximated Pareto front" to refer to the Pareto front obtained from all algorithms and runs and use "solution set" to refer to the Pareto front that was obtained by a single algorithm. Ideal and nadir point are calculated from the approximated Pareto front. The objectives are then normalized using both ideal and nadir points such that all objectives are in the range of 0 and 1, where 0 is better than 1. For the HV the objectives are normalized as well and a reference point of $r = (1.1, 1.1, ..., 1.1)$ is used. Since this value is slightly higher than the worst normalized value, i.e. 1, the HV value will better reflect the diversity of a solution set.

Decision Variables	Possible inputs
Primary Axis Grid	[8, 12, 16, 20, 28]
Primary Axis Fields	1 - 15
Secondary Axis Grid	[6, 12, 18]
Secondary Axis Fields	1 - 20
Primary Structure Type	[Steel Framework, Steel Profile, T-Beam Precast Concrete, Glued Laminated Timber Beam, Timber Framework]
Secondary Structure Type	[Steel Framework, Steel Profile, Beam Precast Concrete, Timber Beam]
Column Type	[Precast Concrete-Quadratic, Steel-HEM]
Bracing Type	[Clamped, Joint XY - 2X-2Y]
Retrofitting	[0, 1, 2]

Table 4.4: Decision variables and ranges

4.2.2 Results

In this section the results for each test case are presented separately. First, for each test case the search space properties are evaluated to have a better insight to why the results are as they are. This includes information about how many unique and feasible solutions have been found during all runs for the specific test case as well as the size of the approximated Pareto front. Afterwards the HV and IGD+ are shown, containing the mean value and standard deviation from five runs. As mentioned, the higher the HV value the better. On the contrary, a lower IGD+ value is desirable. For reference, the hypervolume of the approximated Pareto front (labeled as PF) is always included. The IGD+ value of the approximated Pareto front is always zero. Furthermore, solution sets are compared against the approximated Pareto front and between each algorithm. For each test case four structural layouts are presented - the cheapest, the most environmentally friendly, the most flexible or the most balanced structure, i.e. minimum distance to ideal point. The structures have been selected according to the weighted sorting, i.e. for the most flexible structure all flexibility objectives were set to 100% while all other objectives were set to 0. The best ranked structure after sorting was selected. Finally, time measurements are reported for all three test cases combined.

Test Case 1

Combining all calculated solutions from each run and each algorithm a total of 15750 solutions have been evaluated for test case 1. The results for the search space analysis is

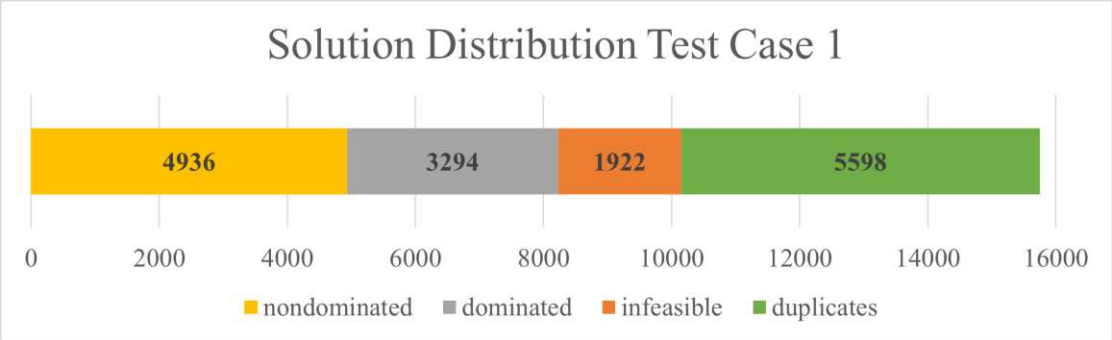
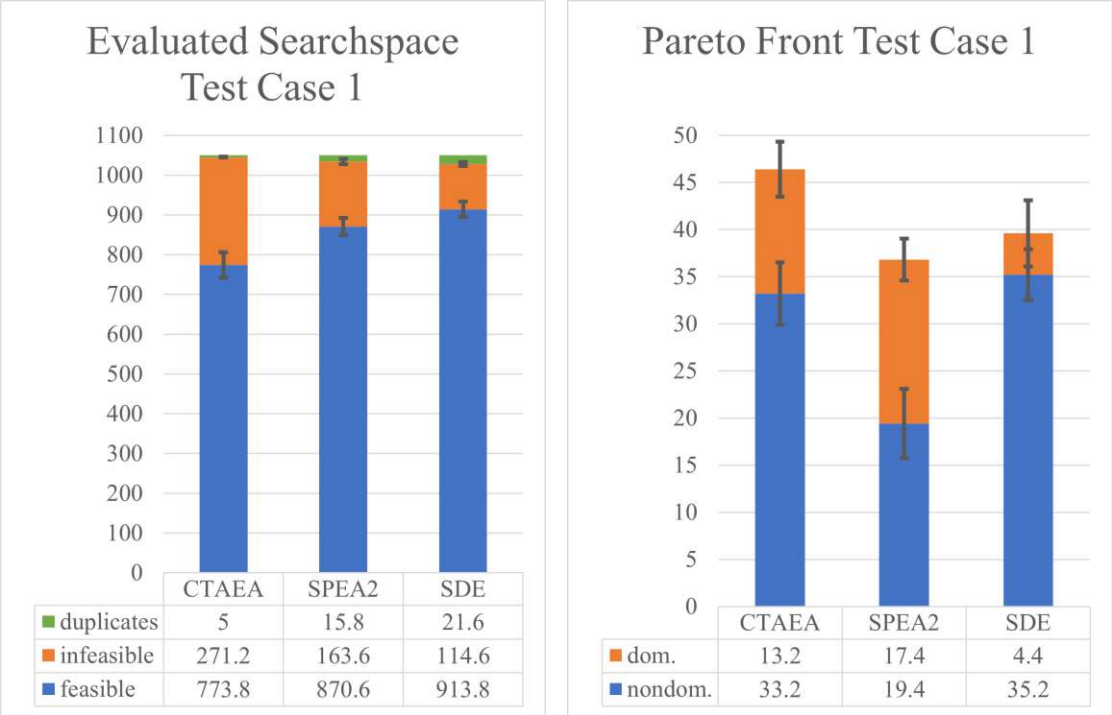


Figure 4.8: Solution distribution of test case 1. Solutions combined from five runs.



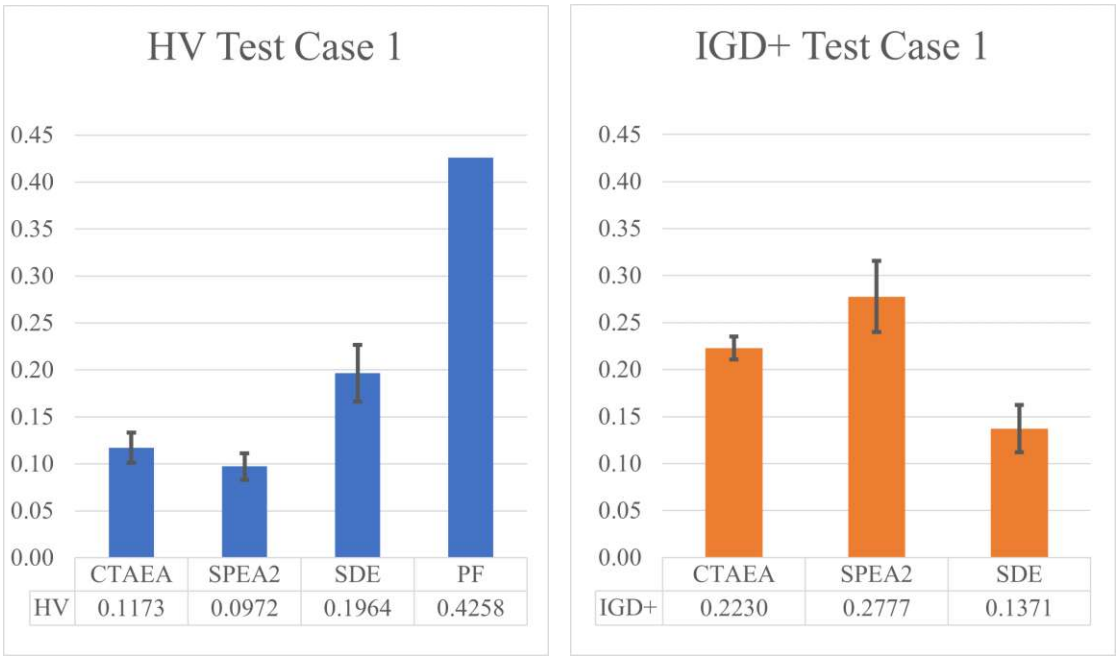
(a) feasibility of evaluated search space

(b) size of Pareto front

Figure 4.9: Search space analysis of test case 1. Average number and standard deviation from five runs for each algorithm.

visualized in Figure 4.8. Out of the 15750 solutions 10152 are unique solutions. 1922 solutions were constrained by objective space constraints and 8230 solutions were feasible. Therefore, around 18.93% of the search space is infeasible. The approximated Pareto front of test case 1 consists of 4936 solutions and makes around 48.62% of the evaluated search space. According to this, almost every second solution is a nondominated solution.

Further breaking down the search space by each algorithm Figure 4.9a shows that relatively

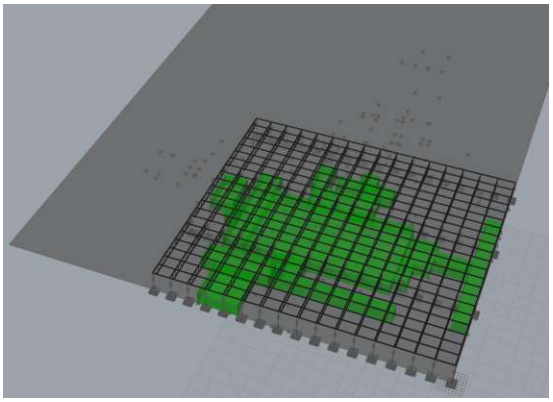


(a) HV (b) IGD+

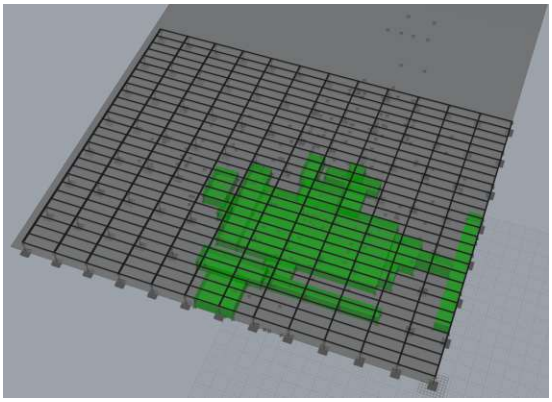
Figure 4.10: HV and IGD+ of the solution sets from test case 1

less duplicates have been evaluated meaning that most duplicates from Figure 4.8 were either across multiple runs within the same algorithm or across multiple algorithms. SDE had the highest number of duplicate evaluation (avg. 22), which is around 2% of the search space. On the other hand SDE found the highest number of feasible solutions (avg. 914), while CTAEA has the highest infeasible number of solutions (avg. 271). SPEA2 lies in between SDE and CTAEA regarding feasible and infeasible solutions (avg. 163 and 871 respectively). However, looking at Figure 4.9b SPEA2 has the smallest unique and feasible solution set (avg. 37). CTAEA has the largest overall solution set (avg. 46) while SDE has the largest nondominated solution set (avg. 35). With only 4 dominated solutions on average, SDE's solution set has the highest percentage of nondominated solutions.

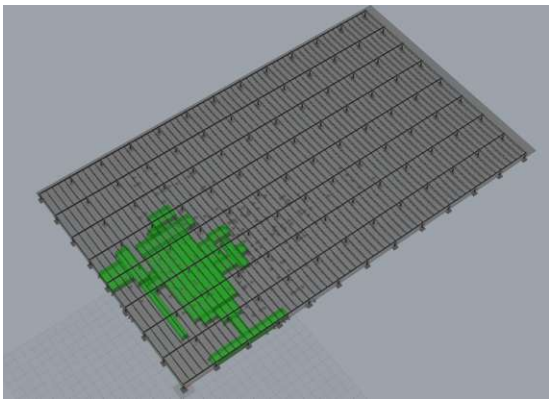
The HV and IGD+ values for each algorithm are shown in Figure 4.10a and 4.10b. It can be derived that SDE performs the best, while SPEA2 is ranked last in terms of both values. The approximated Pareto front has a HV of 0.43. The HV value for CTAEA and SPEA2 are around 0.12 and 0.10, while SDE has an avg. HV value of 0.20. SDE, therefore, performs around 100% better then CTAEA and SPEA2. When looking at the IGD+ value, again, SDE performs better. SDE has a value around 0.14, while CTAEA and SPEA2 have a value around 0.22 and 0.28 respectively, which is around 57% and 100% worse than SDE. All in all, for test case 1 SDE performs best, followed by CTAEA and then by SPEA2.



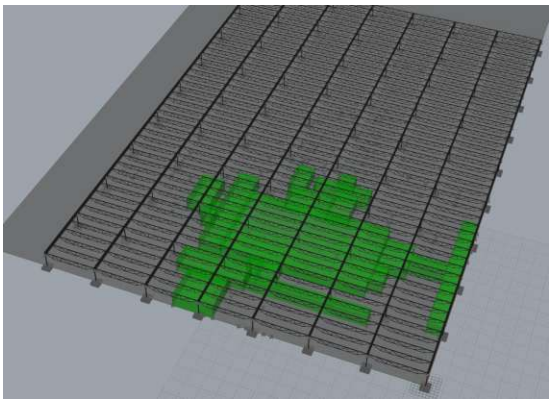
(a) Cheapest structure. Grid: 28m x 6m, x: Timber Framework, y: Timber Beam



(b) Most environmentally friendly structure. Grid: 16m x 12m, x: T-Beam Precast Concrete, y: Beam Precast Concrete



(c) Most flexible structure. Grid: 20m x 18m, x: Steel Profile, y: Steel Framework



(d) Most balanced structure. Grid: 20m x 18m, x: T-Beam Precast Concrete, y: Steel Framework

Figure 4.11: Four different structural layouts for test case 1, each focusing on different objectives: (a) cheapest, (b) most environmentally friendly, (c) most flexible, (d) most balanced.

Figure 4.11 shows four structures sorted by different objectives. The cheapest structure has a wide primary and narrow secondary grid axis (28m x 6m). The whole structure size is kept to a minimum. It only fits one production layout. Timber framework and timber beam is used as the roof construction type. The most environmentally friendly structure is larger in size and has an axis grid of 16m x 12m. The roof construction type is beam precast concrete in both directions. The most flexible structure spans most of the property. It is the largest structure of all four. Steel profile and framework is used for the roof construction. The most balanced structure, i.e. all objectives weighted equally, is a structure with a grid size of 20m x 18m. It spans a large portion of the property and uses precast concrete and steel framework as the roof construction type.

Test Case 2

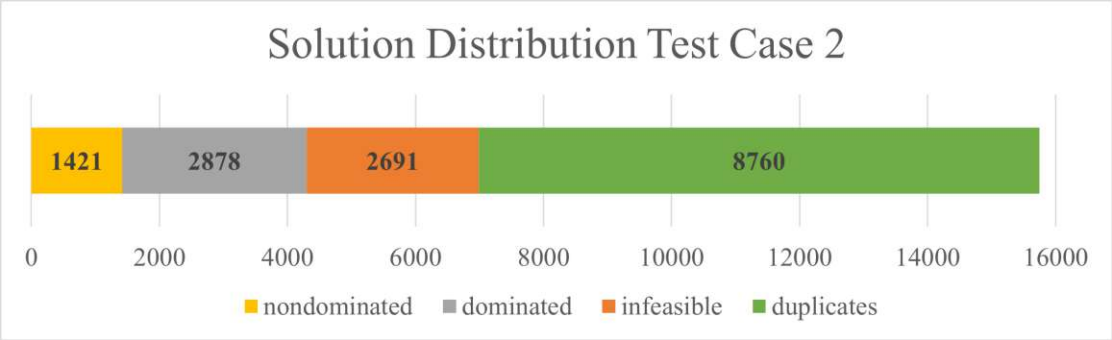
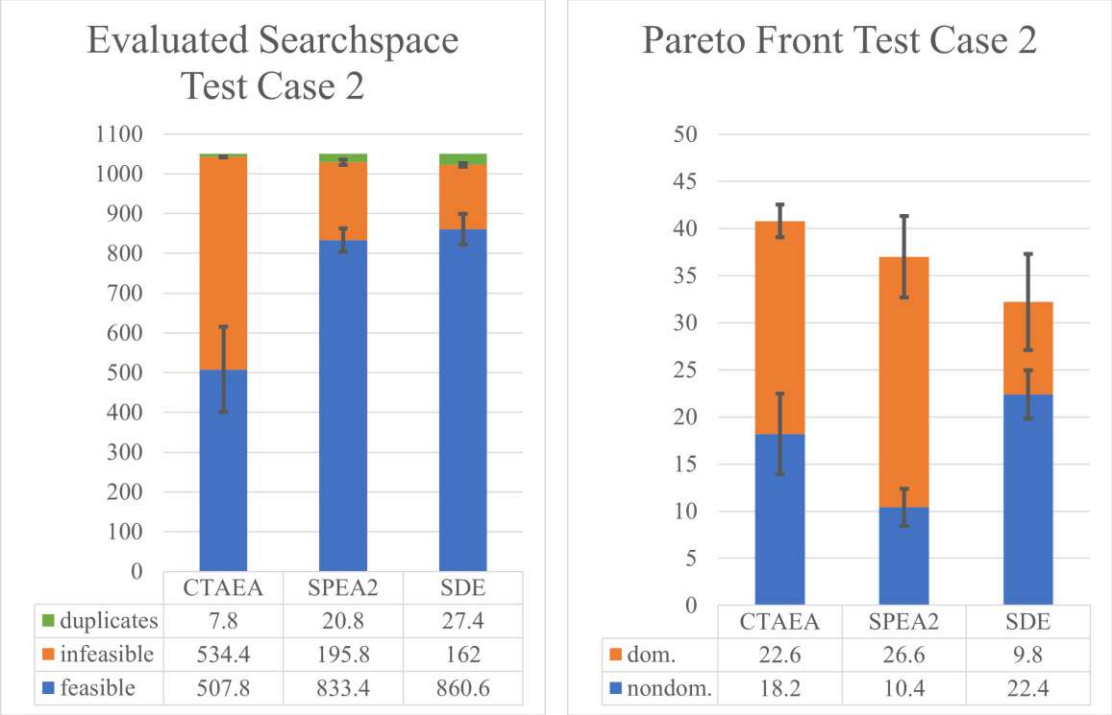


Figure 4.12: Solution distribution of test case 2. Solutions combined from five runs.



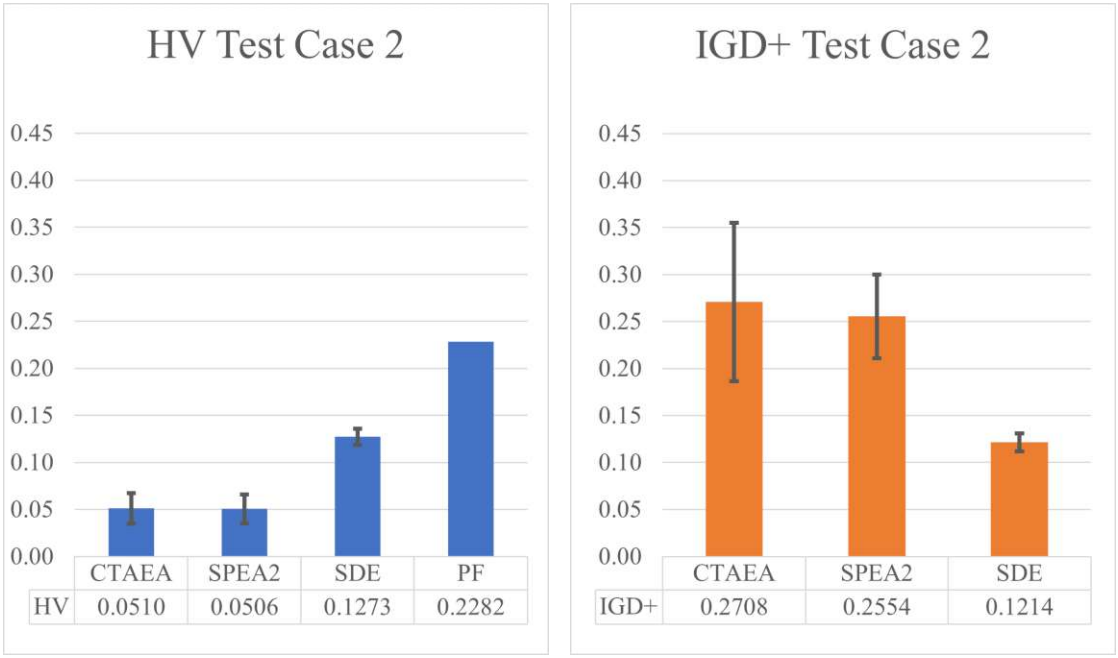
(a) feasibility of evaluated search space

(b) size of Pareto front

Figure 4.13: Search space analysis of test case 2. Average number and standard deviation from five runs for each algorithm.

Search space analysis for test case 2 is depicted in Figure 4.12. Since test case 2 has a smaller search space than test case 1, the number of unique solutions obtained is smaller, which is 6990. Therefore, all runs combined have evaluated around 35% of the complete search space. Around 38.5% of the obtained solutions are infeasible. This is a much higher percentage than in test case 1. Out of the 4299 feasible solutions, 1421 are

nondominated and form the approximated Pareto front for test case 2. The approximated Pareto front makes around 20.33% of the search space.

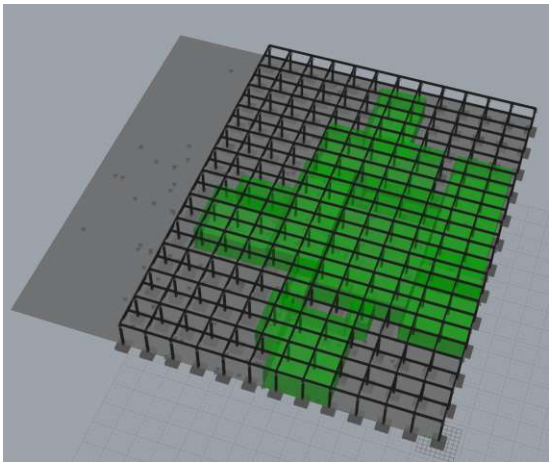


(a) HV (b) IGD+

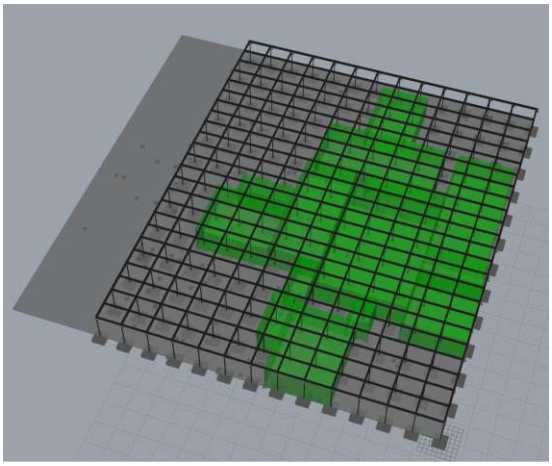
Figure 4.14: HV and IGD+ of the solution sets from test case 2

Breaking down the search space by algorithm Figure 4.13a reveals that CTAEA has a very high contribution of infeasible solutions with high variance. Out of 1050 solutions, more than 50% where infeasible. Both SPEA2 and SDE covered around the same ratio of feasible and infeasible solutions, around 80% feasible and 20% infeasible solutions. Figure 4.13b shows that the solution set sizes on average with 36 solutions are smaller than in test case 1. The ratio of nondominated solutions is very low for SPEA2, less than 30%. The number of nondominated solution obtained by CTAEA is less than 50%, which is more than SPEA2 even though it found less feasible solutions. SDE has on average the smallest solution set but the highest number of nondominated solutions.

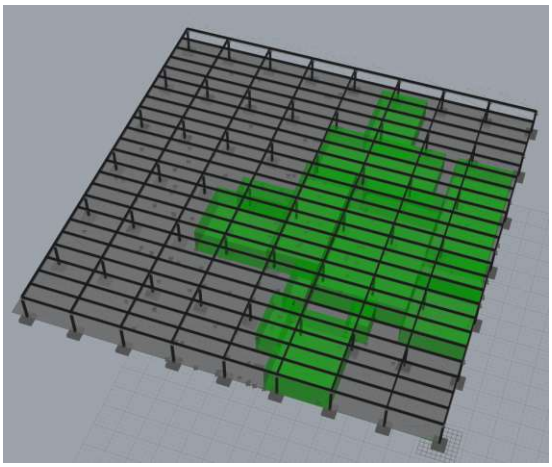
Out of the three algorithms SDE performs the best again. Figure 4.14a and 4.14b show that SDE had the highest HV (avg. 0.13) and the lowest IGD+ value (avg. 0.12) compared to the other two algorithms. SDE's HV value is more than 100% better than that of CTAEA or SPEA2 (both avg. 0.05) and its HV covers more than 50% of the HV of the approximated Pareto front (avg. 0.23). Regaring IGD+, CTAEA (avg. 0.27) and SPEA2 (0.26) have a value that is more than 100% higher than SDE's. Furthermore, the variance on SDE is small as well (0.01), while the variance of CTAEA on IGD+ with around 0.08 is quite large. On this test case CTAEA and SPEA2 perform equally well and SDE outperforms both.



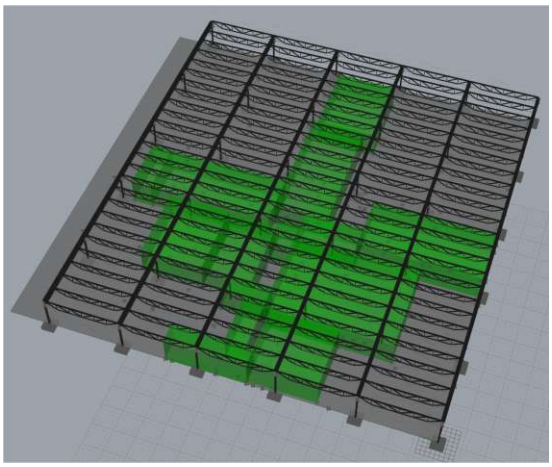
(a) Cheapest structure. Grid: 8m x 6m, x: Timber Framework, y: Timber Beam



(b) Most environmentally friendly structure. Grid: 8m x 6m, x: T-Beam Precast Concrete, y: Beam Precast Concrete



(c) Most flexible structure. Grid: 16m x 12m, x: Steel Profile, y: Steel Profile



(d) Most balanced structure. Grid: 20m x 18m, x: T-Beam Precast Concrete, y: Steel Framework

Figure 4.15: Four different structural layouts for test case 2, each focusing on different objectives: (a) cheapest, (b) most environmentally friendly, (c) most flexible, (d) most balanced.

Figure 4.15 shows the cheapest, most environmentally friendly, most flexible and most balanced structural layouts for test case 2. The roof structure type for all three structures are the same as in test case 1. Only the grid sizes vary. The first two structures have the most narrow grid, while the flexible and balanced structures have a more wider grid. Not surprisingly, the cheapest has the smallest structure while the most flexible structure is the largest.

Test Case 3

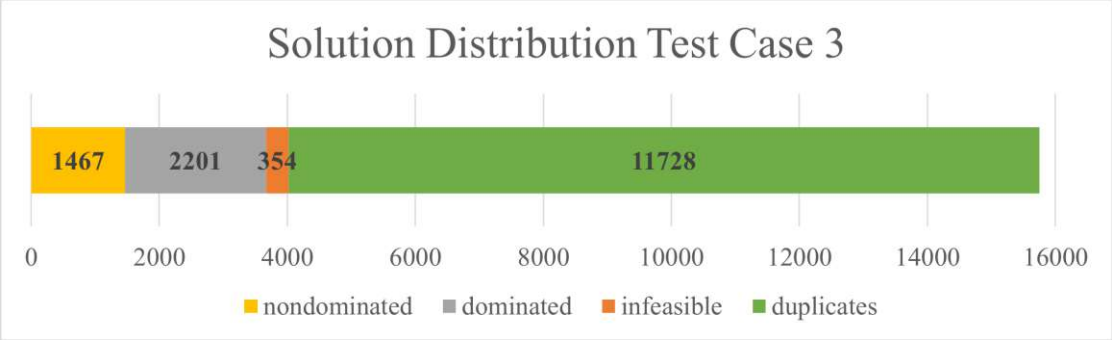
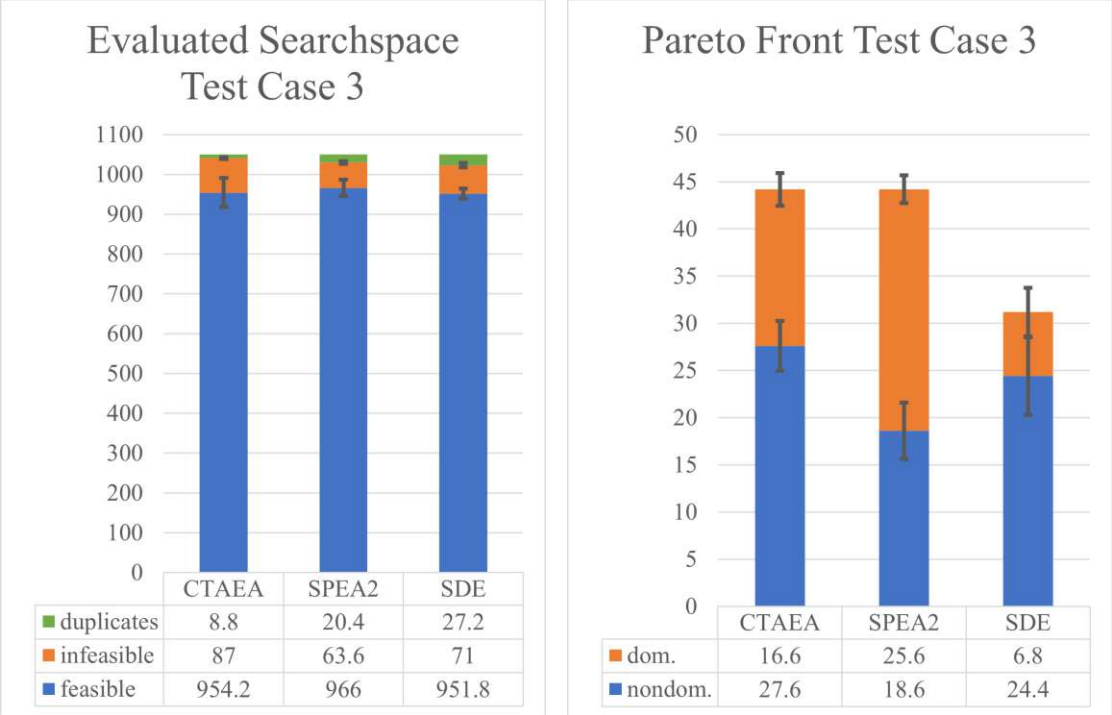


Figure 4.16: Solution distribution of test case 3. Solutions combined from five runs.



(a) feasibility of evaluated searchspace

(b) size of Pareto front

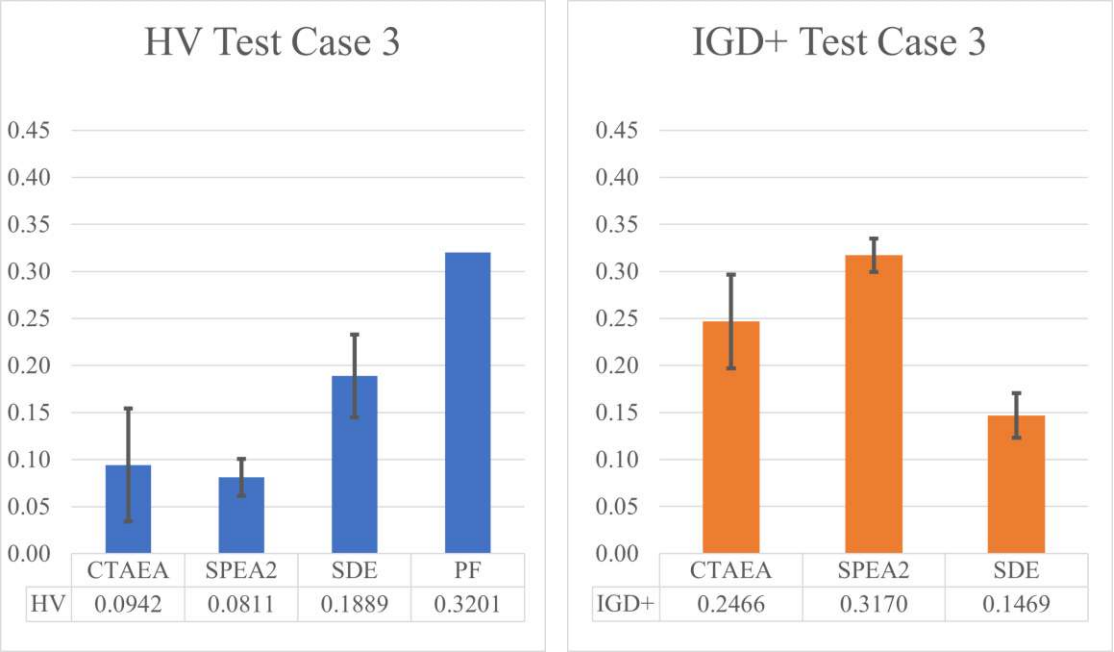
Figure 4.17: Search space analysis of test case 3. Average number and standard deviation from five runs for each algorithm.

Test case 3 has the smallest search space out of all three test cases. Even though the number of calculated solutions highly surpasses the size of the search space (around 6000 possible combinations), only 4022 unique solutions were evaluated and 11728 evaluations were duplicates (see Figure 4.16). 354 solutions were infeasible, which is about 9% of the evaluated search space. The approximated Pareto front encompasses 1467 solutions

which makes around 40% of the feasible solutions.

Figure 4.17a shows that all three algorithms covered around the same size of feasible and infeasible solutions. All three found between 950 and 970 feasible solutions. CTAEA found slightly more infeasible solutions (avg. 87) compared to SPEA2 and SDE (avg. 63.6 and 71). On the other hand SPEA2 and SDE found slightly more duplicates (avg. 20.4 and 27.2) compared to CTAEA (avg. 8.8). In terms of solution sets (see Figure 4.17b) SDE has the smallest overall set (avg. 31.2). SPEA2 and CTAEA have a larger set but include more solutions that are dominated.

Even though SDE has the smallest solution set on average Figure 4.18 shows that its solution set performs better than that of CTAEA or SPEA2. The HV obtained by SDE (avg. 0.19) is again on average around 100% better than that of CTAEA (avg. 0.09) and SPEA2 (avg. 0.08) and covers more than 50% of the HV from the approximated Pareto front (avg. 0.32). Similarly, the IGD+ value from SDE solution sets (avg. 0.15) is smaller compared to CTAEA (avg. 0.25) and SPEA2 (avg. 0.32). Overall, SDE performs best for test case 3, followed by CTAEA and SPEA2.

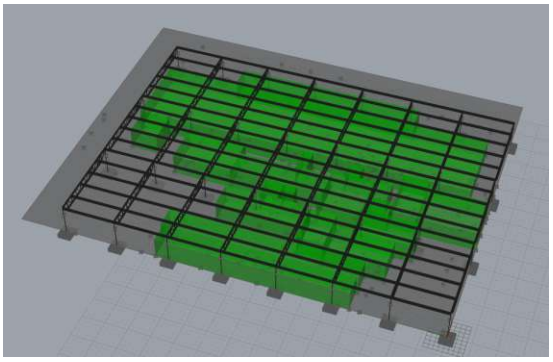


(a) HV (b) IGD+

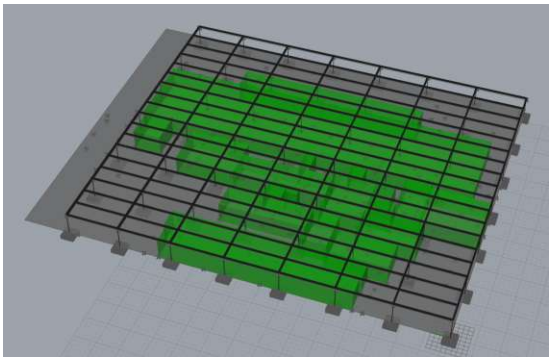
Figure 4.18: HV and IGD+ of the solution sets from test case 3

The results of this study suggest that SDE is the best performing algorithm by far out of the three algorithms with the given parameters for the optimization. CTAEA did slightly better than SPEA2. In conclusion this study suggests that SDE should be the preferred optimization algorithm for many-objective optimization of industrial building structures.

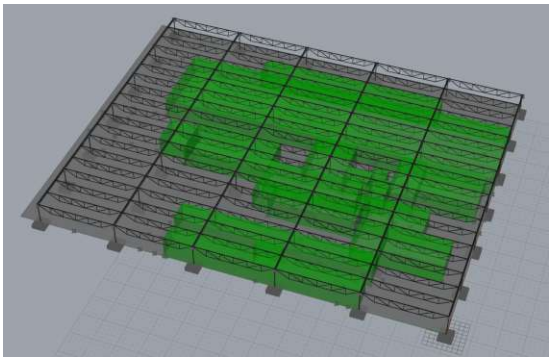
Figure 4.19 shows four resulting structural layouts for test case 3. The most flexible



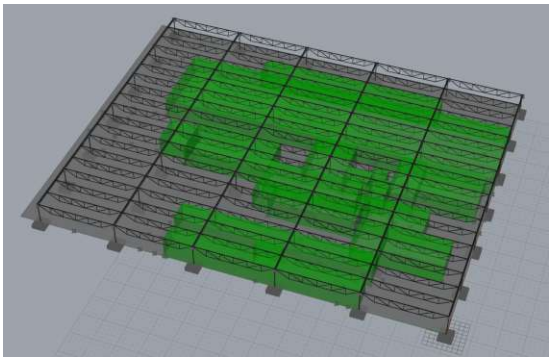
(a) Cheapest structure. Grid: 20m x 12m, x: Timber Framework, y: Timber Beam



(b) Most environmentally friendly structure. Grid: 12m x 12m, x: T-Beam Precast Concrete, y: Beam Precast Concrete



(c) Most flexible structure. Grid: 12m x 18m, x: Steel Profile, y: Steel Framework



(d) Most balanced structure. Grid: 12m x 18m, x: Steel Profile, y: Steel Framework

Figure 4.19: Four different structural layouts for test case 3, each focusing on different objectives: (a) cheapest, (b) most environmentally friendly, (c) most flexible, (d) most balanced.

building is also the most balanced one. With exception to the balanced structure, the first three structural layouts have the same roof construction type and only the grid sizes vary.

Time Measurements

For each test case and run the time was measured as well. The runs were done on a Windows 10 Pro machine with an AMD Ryzen 7 1700 Eight-Core Processor and 32 GB RAM. Figure 4.20 summarizes the average measured time in hours for each algorithm and test case over 5 runs. However, the displayed time is the time spend computing the structures and the objectives and therefore does not contain the time consumed by the algorithms as the amount of time consumed by the algorithm is very small compared to the structure generation. We confirmed that the computation time of the algorithm is negligible in comparison to computation time for structures and objectives by doing one

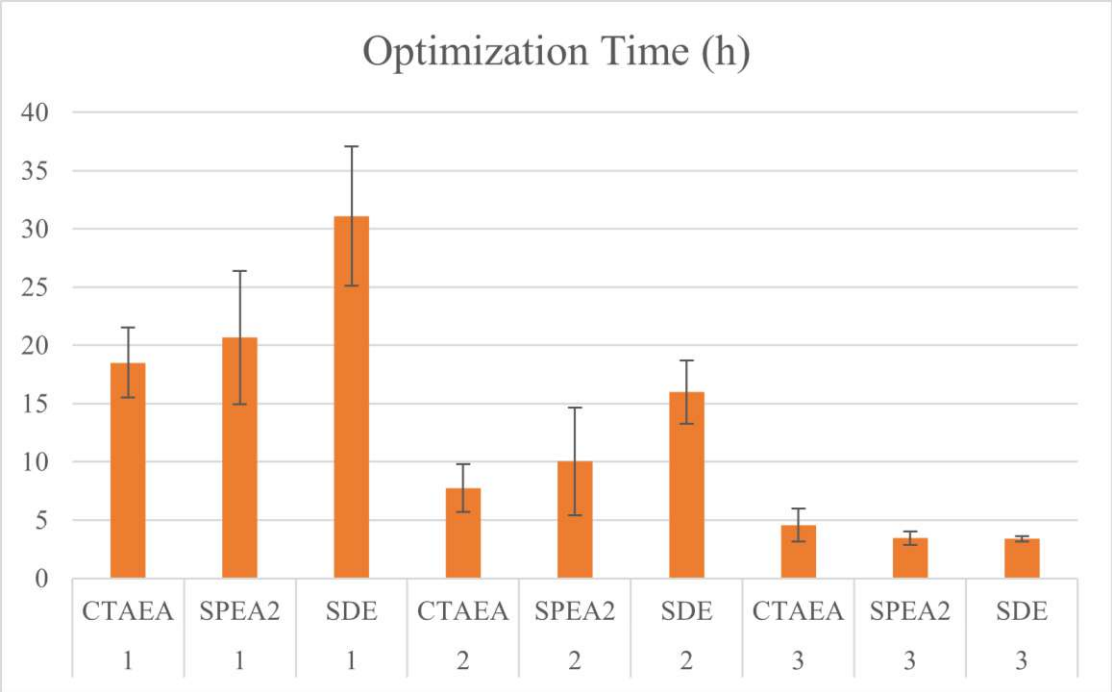


Figure 4.20: Average optimization time over 5 runs in hours (y-axis) for each test case and algorithm (x-axis).

run for each algorithm for test case 2 and 3. Two different time spans were measured. First, the time for each structure generation, and secondly, the total time of the whole optimization. The time difference between the sum of all generated structures and the total time is the time that was consumed by the algorithm for optimization. In both cases, SDE and SPEA2 take around 25 seconds, and CTAEA takes around 3 minutes and 25 seconds confirming that the most time is spend on structure generation.

Comparing all three test cases, test case 1 took the longest to optimize. Test case 1 has the largest property and less compact production layouts. As a result larger structures are generated. We hypothesize that larger structures need more computation time and therefore lead to longer optimization time. Throughout all runs for test case 1 the fastest optimization run was finished after 11 hours while the longest took 37 hours. On average CTAEA finished the earliest, followed by SPEA2. Optimizations with SDE took the longest (avg. 31 hours). However, since these times do not include algorithm computation time, it only can be concluded that SDE generated layouts that took longer to compute. Optimizations on test case 2 were a little bit faster. The fastest run took around 3.5 hours. The longest run took around 18 hours. CTAEA took the least time to optimize, followed by SPEA2 and SDE. Finally, test case 3 took the least time to optimize for all algorithms (between 2.5 and 4.5 hours). On average structural layouts generated by CTAEA took the longest to compute on test case 3. Compared to test case 2 and 1, there is only a small deviation in time on test case 3 for all algorithms.

CHAPTER 5

Discussion and Possible Improvements

5.1 Limitations

Both, user and algorithm study, were conducted independently and did not influence the setup or results of the other study. Especially, the user study may have profited by the outcome of the algorithm study, since the choice of the algorithm have a high impact on the final solution set that was presented to the users. Furthermore, both studies were limited by computation time of the generation of structural elements and calculation of objectives. The results of the measured times suggest that larger structures take longer to compute. Additionally, we found that the structure type influences the time as well. Structures using Steel Frameworks take longer to generate. The generation of a structure and the calculation of the objectives takes a few seconds to multiple minutes depending on the structure type and size. Hence, the whole optimization process may take several hours to days. Users were not able to run the optimization themselves, which may have given us further insights of the usability and user preferences. The comparison of the algorithms was limited to only one parameter set. Moreover, for a larger search space a bigger population size may be more beneficial, but would also drastically increase the computation time. It also has been noticed that during different runs the CPU utilization would vary greatly. Sometimes CPU utilization was at 100% over the complete run, at other times it utilized only 20% for one run, resulting in a longer optimization run. Nonetheless, through both studies important points have been discovered.

5.2 User Study

The first part of the evaluation tries to answer the first question of the thesis, whether the optimization tool provides a good overview of different designs and whether it supports

users in their decision making. The results indicate that the tool was able to assist the participants in early planning phases and the users found the tool useful. On average the tool scored 4.09 out of 5. Furthermore, users had to select a final layout from either method A or M. Sixteen out of eighteen cases the automatically generated layouts did not produce worse solutions. The tool, therefore, can compete with the manual generation in terms of objective optimization.

However, overall users were more pleased with the manual method than the automated method, i.e. the optimization tool. In the category technology satisfaction engineers were less satisfied with the optimization tool while architects rated both methods almost equally. This reveals that some needs that only concerns engineers were not covered by the tool. The answers to the qualitative questions indicate that information need to be presented in a more understandable way, e.g. names instead of indices. This can easily be added to the list representation. Additionally, the columns should be rearranged or shortened. The most interesting results to the users are the objective values and the corresponding input values. Therefore, these columns should be put in front. Constrained solutions can either be filtered and completely removed or expressed with a single column using an error code or a string with the violated constraints.

During the study participants would also ask why the radar diagram sometimes shows zero for some objectives, as zero is intuitively understood as no cost, no flexibility, etc. instead of the worst performing objective in the set. The radar diagram is a good visualization for a fast comparison between a few different solutions. Used correctly, the volume and shape can give information about the underlying data. The problem is that the scale is understood as values rather than a rating. A fast and easy way is to inform users about what the scale means. Adding a short text to describe the values, i.e. "0 - worst" and "1 - best", can be a first step to reduce confusion. Additionally, an informative text can be added to describe in detail how these values are obtained.

Architects were less satisfied in all subcategories on people-process level while engineers only rated the outcome satisfaction noticeably lower. A possible reason for this might be the lack of possibilities to control the search space. Due to time constraints, participants were not able to select their preferred material or combination of materials. Furthermore, during the study they were also not allowed to manually fine-tune the automatically generated layout to include their preferences. In real world application there would not be such constraints. Ideally, users would use the optimization tool to find good solutions and further refine the solutions according to their preferences, as designing often involves subjective elements. Moreover, the parametric model concentrates a lot on the technical aspects and gives less attention to the visualization of look and feel of such structures.

Generally, an interesting but not surprising discovery is that the aesthetics of the buildings should be considered in some way as well. There has been a clear preference in choosing the same material for both primary and secondary structure types. Users would more likely choose timber and steel when using method M, which looks more appealing than concrete. The combinations timber/timber and steel/steel were recorded very often. The optimization tool, however shows the opposite - many concrete/concrete combinations.

It also has been noted that the first best ranked layouts from the optimization, according to the distance to the ideal point, had concrete/concrete as their material, which would explain the high numbers for method A. Consideration of aesthetics can be added in at least two ways, either before or after optimization. Prior to the optimization, it can be incorporated as one of the objectives for the optimization. Since this is a rather subjective objective, users can give weightings to different combinations, i.e. through the excel file. Another way is to filter for different material combinations after optimization. Currently, searching for different combinations is hard, as the focus is more on the objectives. Both ways have advantages and disadvantages. As one of the objectives, aesthetics would guide the search during optimization. Results would have desired combinations but may also interfere with other objectives, which may be more important. Filtering after optimization would result in better solutions, i.e. better objective values in all other categories, but may not include the desired combination at all.

5.3 Algorithm Comparison

The second part of the evaluation answered the second question of the thesis. First, the search spaces of all three test cases have been evaluated to get insight of the different properties and challenges of each test case. Test case 1 had a very large search space with a high number of nondominated solutions. A challenge in such search spaces is to push the search towards the Pareto front and sample the Pareto front efficiently. Test case 2 had a high number of infeasible solutions and a smaller Pareto front. One assumption was that in a more constrained search space the algorithms would be stuck in a local optimum and that CTAEA would do better in such situations as it was designed to tackle this difficulties. However, the results show that this was not the case. Test case 3 has the smallest search space and therefore, approximating the true Pareto front should be easier compared to the other two test cases.

Throughout all test cases it has been revealed that SDE is the one algorithm that clearly performed better than the other two algorithms. SDE always had the highest HV and the lowest IGD+ values. In most cases it performed 100% better than the other two algorithms in both metrics and covers around half the HV of the approximated Pareto front. Since the performance of SDE was consistent through all test cases it can be concluded that the search was not random and that it was guided by the evolutionary algorithm. In test case 1 where many solutions are nondominated SDE managed to keep a good diversity. Compared to CTAEA, SDE had a smaller solution set but around the same number of nondominated solutions that are in the approximated Pareto front. Yet, SDE had a much better HV and IGD+ value.

Not surprising, SPEA2 did not perform well at all. As stated in many researches SPEA2 deals well with two to three objectives, but does not handle well that many objectives. This study confirms the problem with selection pressure using non MaOEA in a real world problem. Through all test cases SPEA2 had the lowest HV and highest IGD+ value. From the solution set it can also be seen that even though SPEA2 has a greater

number of solutions, most of them are dominated compared to the approximated Pareto front, meaning that SPEA2 failed to push the search toward the Pareto front. This may also indicate that the objectives are not strongly correlating with each other.

Surprisingly, CTAEA did not perform well. It barely did better than SPEA2. Especially in test case 2, where large parts of the search space are infeasible it did very poorly. One hypothesis is that since CTAEA balances both feasible and infeasible solutions there was not enough selection pressure towards the feasible Pareto front. It was more guided by infeasible nondominated solutions than feasible ones. The evaluated search space by CTAEA shows that half of the generated solutions were infeasible in order to keep the balance. However, test case 2 might not be constrained enough or disjoined enough in order to see the advantages of CTAEA. In test case 3 CTAEA found many feasible solutions, yet was still not as good as SDE, indicating that the selection of solutions in general might not work well in our test cases.

The study showed that with the default parameters in the optimization tool, SDE performs the best. Further study need to be conducted to see if SDE would outperform the other two algorithms as well with other parameter settings, i.e. smaller or larger population and generation size. Given the great differences between the algorithms, the results suggest that this might be the case.

CHAPTER 6

Conclusion and Future Work

This thesis is part of the BIMFlexi project, whose aim is to develop an integrated BIM-based platform to connect all stakeholders in a building planning process to design flexible and sustainable industrial buildings. In order to help stakeholders in their decision making this thesis' goal was to develop an optimization tool to give an overview over trade-offs between multiple designs. The optimization tool has been evaluated in two studies and both research questions have been answered:

1. *Does an optimization tool provide a good overview of different designs and supports users in their decision making?*
   The user study indicates that users found the optimization tool useful and that it did support them in their decision making. However, at the current stage, users would slightly prefer the manual parameter modification over the results of the optimization.

2. *Is there an algorithm that works better than other algorithms?*
   Comparison of the algorithms revealed that SPEA2+SDE is the one algorithm that performed the best in our test cases with industrial buildings. Hypervolume and Inverted Generalized Distance Plus was used to compare the algorithms. In all three test cases SPEA2+SDE performed around 100% better than the other two algorithms using the default settings for the optimization.

While both studies show that the optimization tool complements the parametric framework, more research and developments need to be done. From the findings of the user study, visualization of data needs improvements. A follow up study could investigate other types of visualization, e.g. different diagrams or scales, to find out which visualization would benefit users the most. Moreover, feedback on the optimization process itself is missing, i.e. working with the excel file and running the optimization only with preferred

settings, as the optimization results were only imported for the user study. Additionally, the optimization should be run using the SPEA2+SDE algorithm to validate the results not only from an algorithmic perspective, but also from a user perspective.

Further research needs to be done on the search space. A greater variety of test cases is needed to investigate the behaviours of the algorithms. The algorithm study only compared three algorithms using one parameter settings. To gain more insight a more comprehensive study with a more diverse set of parameter settings is necessary.

One of the limiting factors was the computation time of the whole optimization process, which is a result between the time needed for generating a structural layout and the number of structural layouts needed for optimization. In order to reduce the whole computational time, either the generation of structural layouts need a faster approximation or a different optimization algorithm should be applied that needs less evaluations for the same or better outcome.

In the next research step of the BIMFlexi project, the current framework will be coupled to a Virtual Reality (VR) application. The results from the optimization will be loaded into the application and users can click through different layouts, while being able to walk inside the virtually generated buildings. It enables an immersive design space exploration experience and will further enhance the interaction and collaboration between stakeholders.

List of Figures

List of Tables

Bibliography

[ABC+20] Charles Audet, Jean Bigeon, Dominique Cartier, Sébastien Le Digabel, and Ludovic Salomon. Performance indicators in multiobjective optimization. *European journal of operational research*, 2020.

[Age19] International Energy Agency. Global status report for buildings and construction 2019. IEA Paris, France, 2019.

[BD20] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[BDOOM16] Nathan Brown, JIF De Oliveira, J Ochsendorf, and Caitlin Mueller. Early-stage integration of architectural and structural performance in a parametric multi-objective design tool. In *International conference on structures and architecture*, 2016.

[BIM20] BIM-based digital Platform for Flexible Design and Optimization of Industrial Buildings for Industry 4.0. `https://www.industriebau.tuwien.ac.at/forschung/forschungsprojekte/bimflexi/`, 2020. [Online; accessed 10-November-2021].

[BNE07] Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[BvdBHE20] Sjonnie Boonstra, Koen van der Blom, Herm Hofmeyer, and Michael TM Emmerich. Conceptual structural system layouts via design response grammars and evolutionary algorithms. *Automation in Construction*, 116:103009, 2020.

[BZ11] Johannes Bader and Eckart Zitzler. Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary computation*, 19(1):45–76, 2011.

[CDFL19] Carmine Cavalliere, Guido Raffaele Dell'Osso, Fausto Favia, and Marco Lovicario. Bim-based assessment metrics for the functional flexibility of building designs. *Automation in Construction*, 107:102925, 2019.

[CJOS16] Ran Cheng, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):773–791, 2016.

[DA+95] Kalyanmoy Deb, Ram Bhushan Agrawal, et al. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

[DG+96] Kalyanmoy Deb, Mayank Goyal, et al. A combined genetic adaptive search (geneas) for engineering design. *Computer Science and informatics*, 26:30–45, 1996.

[DJ13] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601, 2013.

[DPAM02] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[foo] food4rhino. `https://www.food4rhino.com/de`. [Online; accessed 16-November-2021].

[Ger16] Rob Geraedts. Flex 4.0, a practical instrument to assess the adaptive capacity of buildings. *Energy Procedia*, 96:568–579, 2016.

[Gra] Grasshopper3d. `https://www.grasshopper3d.com/`. [Online; accessed 15-November-2021].

[GWT+19] Vincent JL Gan, Chun Lok Wong, Kam Tim Tse, Jack CP Cheng, Irene MC Lo, and Chun Man Chan. Parametric modelling and evolutionary optimization for cost-optimal and low-carbon design of high-rise reinforced concrete buildings. *Advanced Engineering Informatics*, 42:100962, 2019.

[HAPB18] Tofigh Hamidavi, Sepehr Abrishami, Pasquale Ponterosso, and David Begg. Optimisation of structural design by integrating genetic algorithms in the building information modelling environment. *International Journal of Civil, Environmental, Structural, Construction and Architectural Engineering*, 12(9):888–893, 2018.

[HSOK07] Ken Harada, Jun Sakuma, Isao Ono, and Shigenobu Kobayashi. Constraint-handling method for multi-objective function optimization: Pareto descent repair operator. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 156–170. Springer, 2007.

[Hug07] Evan J Hughes. Msops-ii: A general-purpose many-objective optimiser. In *2007 IEEE Congress on Evolutionary Computation*, pages 3944–3951. IEEE, 2007.

[IAON11] Hisao Ishibuchi, Naoya Akedo, Hiroyuki Ohyanagi, and Yusuke Nojima. Behavior of emo algorithms on many-objective optimization problems with correlated objectives. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 1465–1472. IEEE, 2011.

[IMTN15] Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. Modified distance calculation in generational distance and inverted generational distance. In *International conference on evolutionary multi-criterion optimization*, pages 110–125. Springer, 2015.

[IS19] Hisao Ishibuchi and Hiroyuki Sato. Evolutionary many-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 614–661, 2019.

[Kno21] Maximilian Knoll. Parametric modeling of flexible structures for the industry 4.0. Master's thesis, TU Wien, 2021.

[KWG16] Iva Kovacic, Linus Waltenbereger, and Georgios Gourlis. Tool for life cycle analysis of facade-systems for industrial buildings. *Journal of Cleaner Production*, 130:260–272, 2016.

[LCFY18] Ke Li, Renzhi Chen, Guangtao Fu, and Xin Yao. Two-archive evolutionary algorithm for constrained multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 23(2):303–315, 2018.

[LDZK14a] Ke Li, Kalyanmoy Deb, Qingfu Zhang, and Sam Kwong. An evolutionary many-objective optimization algorithm based on dominance and decomposition. *IEEE Transactions on Evolutionary Computation*, 19(5):694–716, 2014.

[LDZK14b] Ke Li, Kalyanmoy Deb, Qingfu Zhang, and Sam Kwong. An evolutionary many-objective optimization algorithm based on dominance and decomposition. *IEEE transactions on evolutionary computation*, 19(5):694–716, 2014.

[LLTY14] Bingdong Li, Jinlong Li, Ke Tang, and Xin Yao. An improved two archive algorithm for many-objective optimization. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2869–2876. IEEE, 2014.

[LTDZ02] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3):263–282, 2002.

[LW19] Zhi-Zhong Liu and Yong Wang. Handling constrained multiobjective optimization problems with constraints in both the decision and objective spaces. *IEEE Transactions on Evolutionary Computation*, 23(5):870–884, 2019.

[LYL13] Miqing Li, Shengxiang Yang, and Xiaohui Liu. Shift-based density estimation for pareto-based algorithms in many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 18(3):348–365, 2013.

[Mat] Matlab. `https://www.mathworks.com`. [Online; accessed 08-March-2022].

[MN22] Mattia Manni and Andrea Nicolini. Multi-objective optimization models to design a responsive built environment: A synthetic review. *Energies*, 15(2):486, 2022.

[NGY+17] Weikang Ning, Baolong Guo, Yunyi Yan, Xianxiang Wu, Jinfu Wu, and Dan Zhao. Constrained multi-objective optimization using constrained non-dominated sorting combined with an improved hybrid multi-objective evolutionary algorithm. *Engineering Optimization*, 49(10):1645–1664, 2017.

[Opo] Opossum. `https://www.food4rhino.com/en/app/opossum-optimization-solver-surrogate-models`. [Online; accessed 23-December-2021].

[PTL+19] Wang Pan, Michela Turrin, Christian Louter, Sevil Sariyildiz, and Yimin Sun. Integrating multi-functional space and long-span structure in the early design stage of indoor sports arenas by using parametric modelling and multi-objective optimization. *Journal of Building Engineering*, 22:464–485, 2019.

[RBF] Rbfmopt. `https://github.com/bicep/RBFMopt-cli`. [Online; accessed 26-December-2021].

[Rhi] Rhinoceros 3d. `https://www.rhino3d.com/`. [Online; accessed 15-November-2021].

[RKK21] Julia Reisinger, Maximilian Knoll, and Iva Kovacic. Design space exploration for flexibility assessment and decision making support in integrated industrial building design. *Optimization and Engineering*, pages 1–33, 2021.

[RWSK+22] Julia Reisinger, Xi Wang-Sukalia, Peter Kán, Iva Kovacic, and Hannes Kaufmann. Framework for integrated multi-objective optimization of production and industrial building design. In *Proceedings of the 2022 European Converence on Computing in Construction*, 2022.

[RZK+22] Julia Reisinger, Maria Antonia Zahlbruckner, Iva Kovacic, Peter Kán, Xi Wang-Sukalia, and Hannes Kaufmann. Integrated multi-objective evolutionary optimization of production layout scenarios for parametric structural design of flexible industrial buildings. *Journal of Building Engineering*, 46:103766, 2022.

[SAG05] Kristina Shea, Robert Aish, and Marina Gourtovaia. Towards integrated performance-driven generative design tools. *Automation in Construction*, 14(2):253–264, 2005.

[SRS10] Hemant Kumar Singh, Tapabrata Ray, and Warren Smith. C-psa: Constrained pareto simulated annealing for constrained multi-objective optimization. *Information Sciences*, 180(13):2499–2513, 2010.

[TT17] Eleftheria Touloupaki and Theodoros Theodosiou. Performance simulation integrated in parametric 3d modeling as a method for early stage design optimization—a review. *Energies*, 10(5):637, 2017.

[vdBYBE19] Koen van der Blom, Kaifeng Yang, Thomas Bäck, and Michael Emmerich. Towards multi-objective mixed integer evolution strategies. In *AIP Conference Proceedings*, volume 2070, page 020046. AIP Publishing LLC, 2019.

[Vie13] Robert Vierlinger. Multi objective design interface. Master's thesis, TU Wien, 2013.

[Wal] Wallacei. `https://www.food4rhino.com/en/app/wallacei`. [Online; accessed 23-December-2021].

[WJ07] Gaoping Wang and Huawei Jiang. Fuzzy-dominance and its application in evolutionary many objective optimization. In *2007 International Conference on Computational Intelligence and Security Workshops (CISW 2007)*, pages 195–198. IEEE, 2007.

[WM05] David H Wolpert and William G Macready. Coevolutionary free lunches. *IEEE Transactions on evolutionary computation*, 9(6):721–735, 2005.

[WYT09] Yonas Gebre Woldesenbet, Gary G Yen, and Biruk G Tessema. Constraint handling in multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 13(3):514–525, 2009.

[YTPB21] Yun Kyu Yi, Amal Tariq, Jongpil Park, and Dua Barakat. Multi-objective optimization (moo) of a skylight roof system for structure integrity, daylight, and material cost. *Journal of Building Engineering*, 34:102056, 2021.

[YXWY15] Yuan Yuan, Hua Xu, Bo Wang, and Xin Yao. A new dominance relation-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(1):16–37, 2015.

[YYHW20] Xiang Yi, Xiaowei Yang, Han Huang, and Jiahai Wang. Handling constrained multi-objective optimization by ignoring constraints and using two evolutionary frameworks. 2020.

[Zah21] Maria Zahlbruckner. lntegrating production layout planning into structural design for flexible lndustrial buildings. Master's thesis, TU Wien, 2021.

[ZCLK08] Xiufen Zou, Yu Chen, Minzhong Liu, and Lishan Kang. A new evolutionary algorithm for solving many-objective optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(5):1402–1412, 2008.

[ZK04] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *International conference on parallel problem solving from nature*, pages 832–842. Springer, 2004.

[ZL07] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[ZLB04] Eckart Zitzler, Marco Laumanns, and Stefan Bleuler. A tutorial on evolutionary multiobjective optimization. *Metaheuristics for multiobjective optimisation*, pages 3–37, 2004.

[ZLT01] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001.

[ZRWS+22] Maria Zahlbruckner, Julia Reisinger, Xi Wang-Sukalia, Peter Kán, Maximilian Knoll, Iva Kovacic, and Hannes Kaufmann. Evaluation of parametric multi-objective optimization and decision support tool for flexible industrial building design. In *Proceedings of the 2022 European Converence on Computing in Construction*, 2022.

[ZT98] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, pages 292–301. Springer, 1998.

Appendix A: Questionnaire

Questions from the questionnaire. The user study was done in German. A 5-point Likert-scale was used where 5 is the most positive and 1 the most negative response. One question (U1) is an inverted question, i.e. 5 is the most negative and 1 the most positive response. For the graph in the results section the values of this question are inverted to make consistent visualization.

Satisfaction with Process - Prozesszufriedenheit [SP]

- SP1 Der frühzeitige Einsatz des Tools unterstützt meine Leistung im Planungsprozess auf zufriedenstellende Weise.
- SP2 Das Tool hilft mir bei der Herangehensweise an die Designfindung und Optimierungsaufgabe.
- SP3 Das Tool unterstützt mich bei der effizienten Durchführung meiner Aufgabe zur Designfindung in frühen Entwurfsphasen.
- SP4 Das Tool ermöglicht es mir, meine Rolle und Position bei der Gestaltung angemessen umzusetzen (Designerpräferenzen).

Satisfaction with Outcome - Ergebniszufriedenheit [SO]

- SO1 Die gelieferten Ergebnisse des Tools als Entscheidungshilfe sind für meine Aufgaben im Projekt und für die weitere Ausarbeitung zufriedenstellend.
- SO2 Die Ziele, die ich mir für die Designoptimierung gesetzt habe, konnten mit dem Tool untersucht und erfüllt werden.
- SO3 Die Ergebnisse entsprechen meinen ursprünglichen Erwartungen an ein Instrument zur frühen Entscheidungsfindung.
- SO4 Ich bin mit den in der Gruppe erarbeiteten und erzielten Gestaltungsergebnissen zufrieden.

Satisfaction with Collaboration – Kooperationszufridenheit [SC]

- SC1 Durch den Einsatz des Tools konnten die notwendigen Informationen über die Gestaltung und die Entscheidungsfindung rechtzeitig an die Teammitglieder weitergegeben werden.
- SC2 Alle Teammitglieder konnten dank des Tools zufriedenstellend zusammenarbeiten.
- SC3 Das Tool unterstützte die effektive und effiziente Kommunikation zwischen den Teammitgliedern.
- SC4 Durch den Einsatz des Tools waren die Teammitglieder in der Lage, gut zusammenzuarbeiten und sich gegenseitig zu unterstützen.

Ease of Use – Einfachheit der Nutzung [U]

- U1 Wenn ich das Werkzeug benutzte, brauchte ich oft das Handbuch oder andere Unterstützung.
- U2 Ich finde es einfach, das Werkzeug dazu zu bringen, das zu tun, was ich will.
- U3 Die Schnittstelle war intuitiv und selbsterklärend.
- U4 Es war einfach, zwischen verschiedenen Varianten zu wechseln.
- U5 Die Funktionalität des Tools ist leicht zu merken oder zu verstehen
- U6 Es war einfach "Variantenstudien" durchzuführen

Usefulness - Nutzerfreundlichkeit [UF]

- UF1 Ohne dieses Werkzeug wären meine Aufgaben nur schwer zu bewältigen.
- UF2 Durch den Einsatz des Tools erhöht sich meine Leistung bei der Erfüllung der Architektur/Strukturkonzeption.
- UF3 Das Tool steigert meine Produktivität.
- UF4 Die Verwendung des Tools erhöht meine Effizienz/Zeit bei der Erledigung von Aufgaben zur Designfindung.
- UF5 Das Tool erleichtert mir die Erfüllung meiner Aufgaben bei der frühen Entscheidungsfindung
- UF6 Insgesamt halte ich das Instrument als eine frühe Entscheidungshilfe für nützlich

Visualisation and Decision-Making - Visualisierung und Entscheidungsfindung [VD]

- VD1 Das Instrument war bei der interdisziplinären Entscheidungsfindung hilfreich.
- VD2 Die Unterschiede der strukturellen Eigenschaften zwischen den verschiedenen Varianten (Strukturtyp, Strukturmaterialien, Layout) waren leicht zu verstehen und zu untersuchen.
- VD3 Ich habe die räumliche Anordnung des Gebäudes gut verstanden.
- VD4 Ich habe die Struktur des Gebäudes gut verstanden.
- VD5 Die angegebenen Parameter (Variablen) sind ausreichend und zufriedenstellend, um vernünftige Entwurfsergebnisse zu erzielen.
- VD6 Die Visualisierung des Gebäude- und Produktionssystems unterstützte die Entscheidungsfindung im Team.