Chapter 9

Applications in Building Design

Acknowledgement

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

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

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

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

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

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

9.1 Combining Co-Evolution and Optimisation

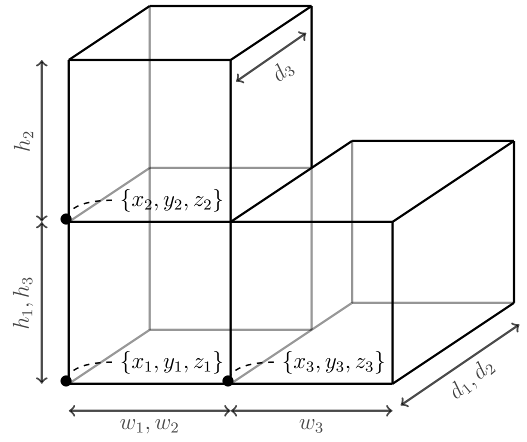
To benefit from both exploration and optimisation, a relay hybrid search using both a superstructure and a superstructure free representation is proposed. To this end, SMS-EMOA-SC (Section 6.3.2) is used with the supercube representation (Section 3.2), and co-evolutionary design (CD) simulations [52]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

9.1.1 Superstructure Free Representation

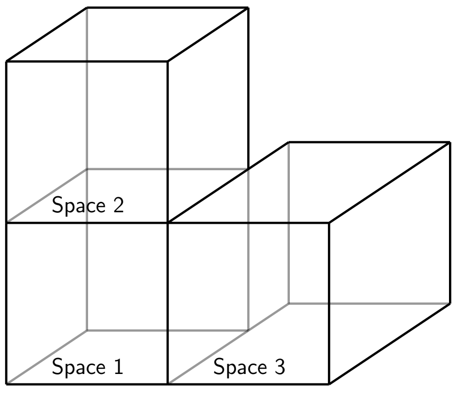
Although in Chapter 3 a superstructure representation was selected for building spatial design optimisation, a design process can also benefit from a free representation. For instance, during optimisation the number of spaces is constant in the supercube representation (Section 3.2), but this is not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

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

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



(a) Movable sizeable representation.



(b) Spatial design.

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

9.1.2 Conversion

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

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

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

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

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

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

9.1.3 Co-Evolutionary Design Simulation

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

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

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

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

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

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

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

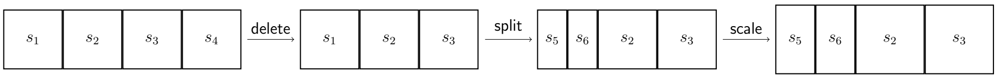


Figure 9.2: A movable sizeable design changed according to CD modification rules.

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

9.1.4 Combination

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

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

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

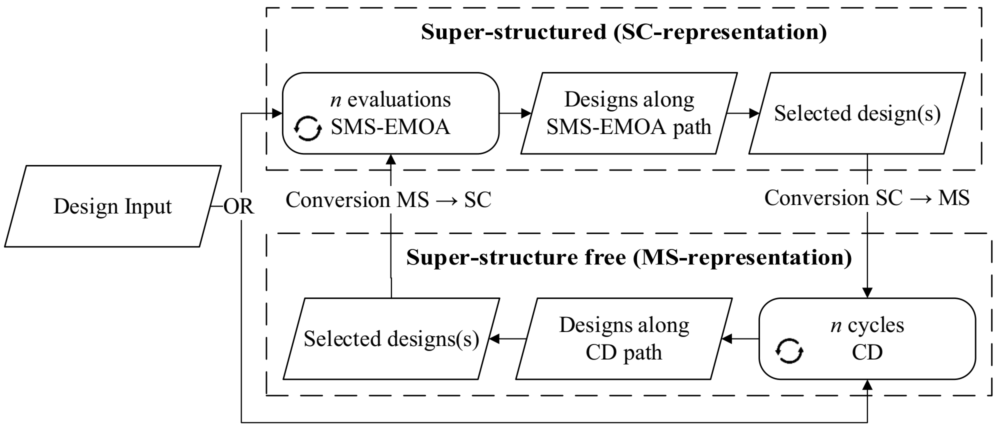


Figure 9.3: Relay hybridisation for the supercube (SC) and movable sizeable (MS) representations.

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

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

9.1.5 Case Study

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

Objectives

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

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

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

Settings

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

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

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

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

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

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

9.1.6 Results

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

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

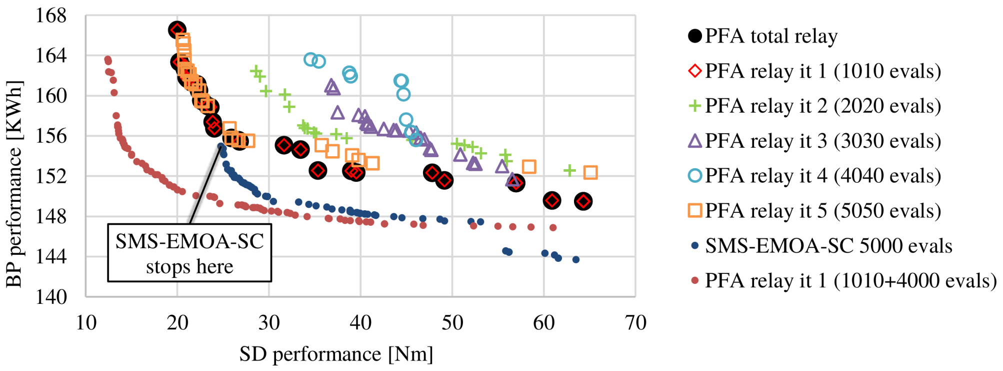


Figure 9.4: Pareto front approximations for each iteration of the relay method, a single run of SMS-EMOA-SC, and an extended run based on the first iteration of the relay.

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

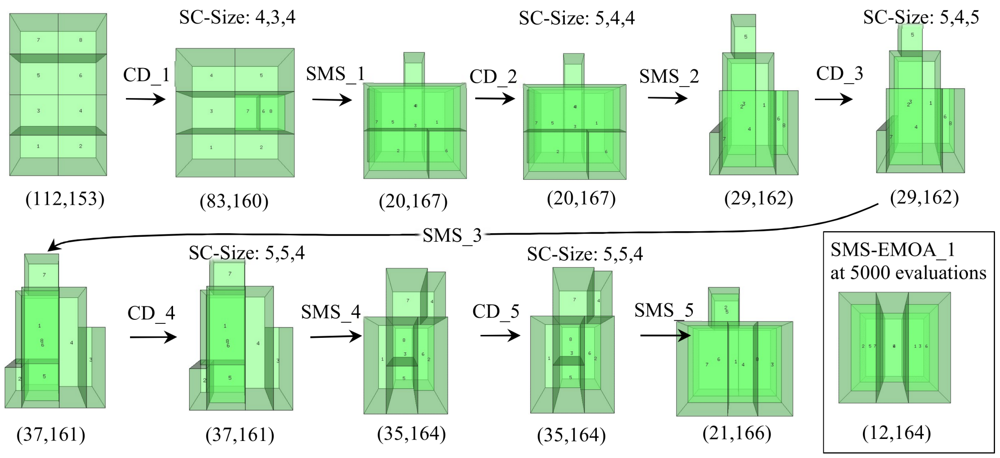


Figure 9.5: The path of the relay method, i.e. selected designs (knee points) after each optimisation run. Here SC stands for supercube.

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

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

9.2 Spatial Design Optimisation in Practice

Acknowledgement

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

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

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

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

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

9.2.1 Integrating BouwConnect BIM and Optimisation

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

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

9.2.2 Case Study

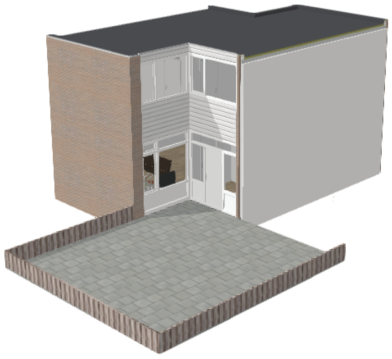
In this case study, a building design from BouwConnect BIM is modified based on optimisation results. To do so, the following procedure is considered. An existing design is selected, the design is then transferred to SMS-EMOA-SC (Section 6.3.2), and optimised. Based on the optimisation results, the design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

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

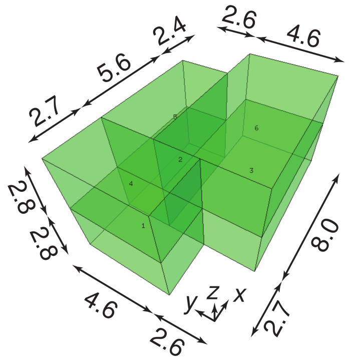
Although some parts of the conversion process from BouwConnect to the optimiser were automated, others could not. Particularly, BouwConnect defines building elements, whereas the supercube representation considers spaces. As a result, spaces had to be defined manually based on the building elements,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

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

[1]`www.bouwconnect.nl`



(a) BIM model in BouwConnect.



(b) Spatial layout with dimensions in metres.

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

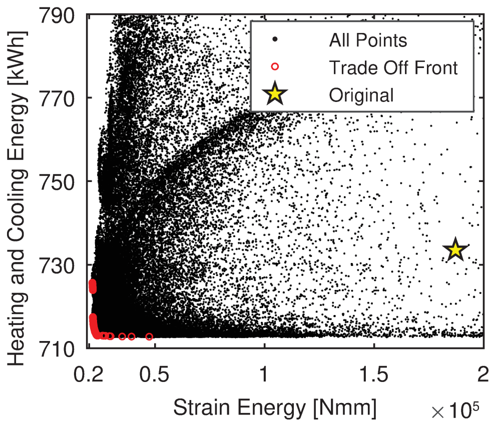
in width, depth, and height. Note that although the supercube settings are based on the original design, SMS-EMOA-SC starts from designs that are randomly initialised within those supercube restrictions. Finally, SMS-EMOA-SC is executed 10 times, for 10 000 evaluations.

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

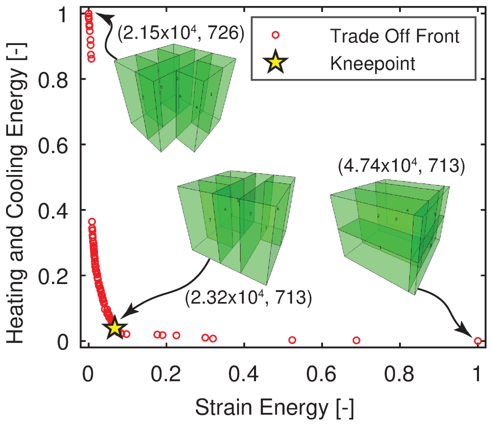
9.2.3 Results

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

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



(a) Scatter plot without outliers (< 5 %).



(b) Normalised Pareto front approximation.

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

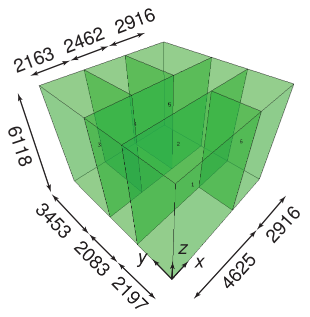


Figure 9.8: The knee point design found by SMS-EMOA-SC, including dimensions in millimetres.

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

A comparison is made between the original and the optimised design based on various characteristics generated by BouwConnect and SMS-EMOA-SC, as shown in Table 9.1. Most notable is that, although the volume of the design is maintained, the floor area has been reduced by 50 %. On the other hand, SMS-EMOA-SC was very effective in reducing the objectives it considered, strain energy, and heating and cooling energy. This also resulted in a reduction of the total annual energy use[2]. The energy index,[2] on the other hand, takes into account the floor area in its computation. Since the floor area became smaller, the energy index be came worse (higher). Specifically, the index changed from $32.82/118 = 0.28\,\mathrm{MW\,h\,m^{-2}}$ to $25.45/59 = 0.43\,\mathrm{MW\,h\,m^{-2}}$. The differences between the values for the heating and cooling energy and the total annual energy use can also be explained. Heating and cooling energy is computed over just six days (details in Section 6.4.1), while the total energy use considers a full year and takes ventilation, hot water, and lighting into account on top of heating and cooling.

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

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

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

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

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

[2] According to Dutch regulation NEN-7120

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

9.3 Structural Design Optimisation

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

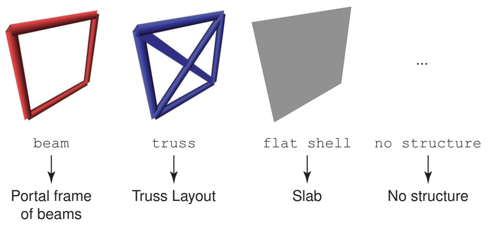
Here a grammar is introduced that aims to generate high quality structural designs in an efficient manner. This contrasts with the handcrafted, and simplistic grammar used earlier in Section 6.4.1. With a grammar that is capable of quickly generating structural grammars it is possible to support architects,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

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

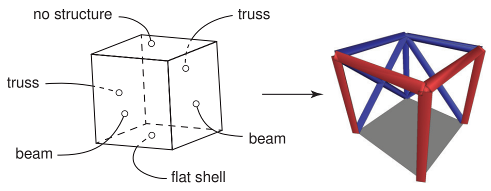
9.3.1 Structural Design

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

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



(a) Possible structural types.



(b) Generated structure based on types.

Figure 9.9: Structural types and their assignment to rectangles.

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

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

9.3.2 Design Response Grammar

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

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

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

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

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

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

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

Algorithm 7 Iterative replacement of substitute components

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

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

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

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

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

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

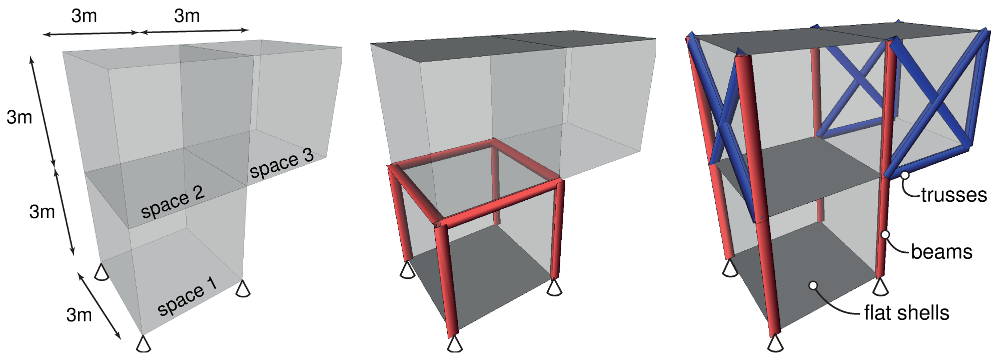
Finally, on line 21 the algorithm checks whether a rectangle still has the substitute type, in which case it is assigned the no structure type instead. This can happen if the noise threshold (η_{noise}) was not exceeded, or if none of the response thresholds $(\eta_{shear}, \eta_{norm}, \eta_{bend})$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

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

9.3.3 Case Study

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

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



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

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

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

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

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

Table 9.2: Settings used for each structural type.

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

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

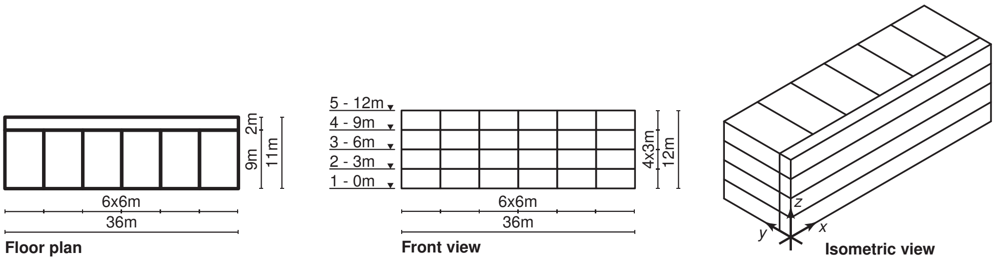


Figure 9.11: Spatial design of a low rise building with four floors, six spaces per floor, and a horizontal walkway on each floor.

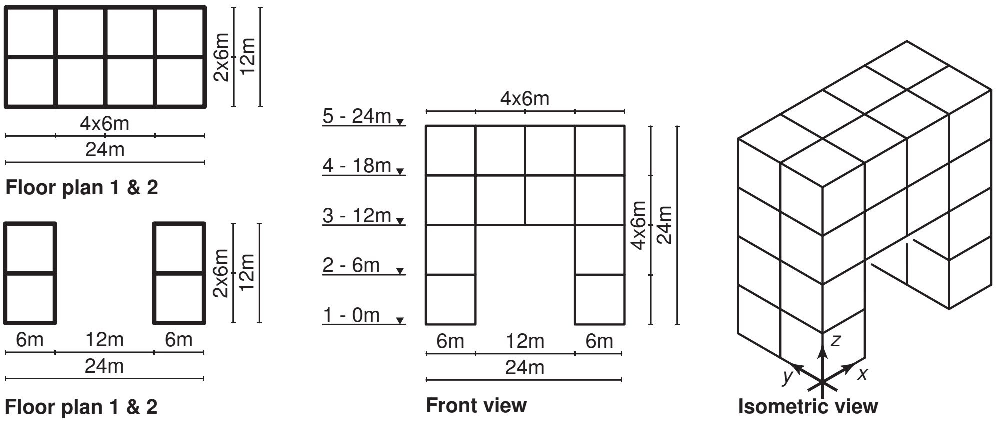


Figure 9.12: Spatial design of a portal building with four floors, 24 spaces, and a passage through its centre.

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

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

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

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

Table 9.3: Design response grammar parameter ranges.

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

9.3.4 Results

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

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

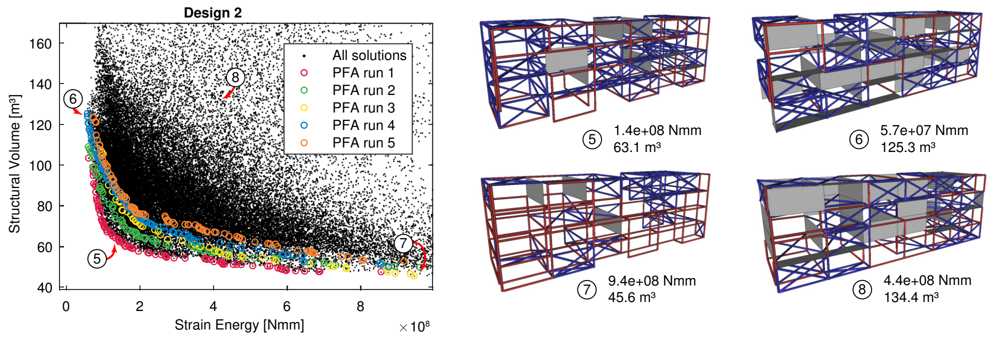


Figure 9.13: Optimisation results of a low rise building by MOMIES (left), and examples of corresponding structural designs (right).

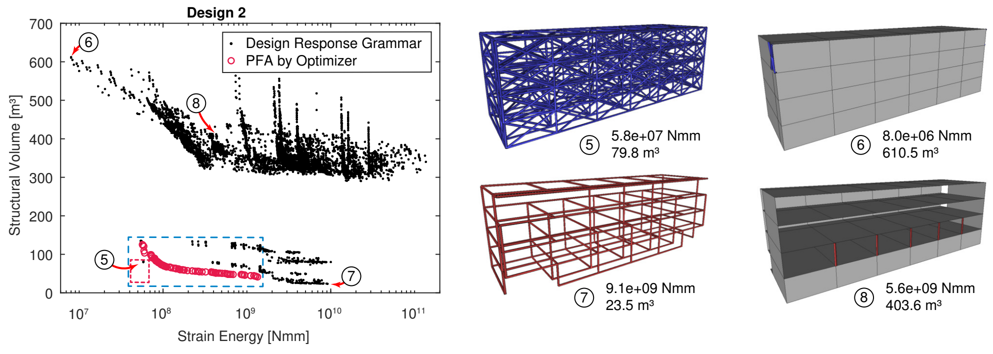


Figure 9.14: Results of the design response grammar for a low rise building compared to the PFA of MOMIES (left), and examples of corresponding structural designs (right).

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

Based on the results of the design response grammar, the parameter configurations in Table 9.4 were chosen for their good performance. Figure 9.15 visualises the results achieved by these parameter configurations, compared to the MOMIES algorithm. MOMIES clearly shows better performance, but the results of the design response grammar are competitive with the PFA. Most importantly, this shows that the design response grammar is indeed capable of finding high quality solutions for minimal computational cost. Each of the parameter configurations of the design response grammar used only two evaluations, whereas MOMIES used 50 000.

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

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

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

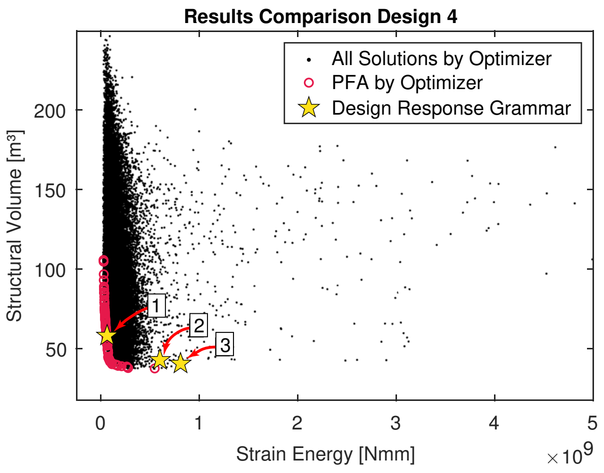


Figure 9.15: Comparison of results by the design response grammar and MOMIES on the portal design.

9.4 Conclusion

9.4.1 Summary

Over the course of this chapter applications of the algorithms developed in this thesis have been explored with the goal of answering RQ6. This question asks how the developed algorithms benefit applications in the real world design process. To this end, the first section of this chapter investigated the interaction of the superstructure based SMS-EMOA-SC algorithm (Section 6.3.2) with a superstructure free co-evolutionary design process that more closely resembles the design practice. In the second section, interaction between an optimisation algorithm (again SMS-EMOA-SC) and a building information modelling environment was studied for practicality. Finally, the third section looked at the use of the generally applicable multi-objective mixed-integer evolution strategy (MOMIES, Chapter 8) in structural design optimisation.

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

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

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

9.4.2 Future Work

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

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

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

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