Toolbox for super-structured and super-structure free multi-disciplinary building spatial design optimisation

S. Boonstra[a,*], K. van der Blom[b], H. Hofmeyer[a,*], M. T. M. Emmerich[b], J. van Schijndel[a], P. de Wilde[c]

[a]*Eindhoven University of Technology, The Netherlands*
[b]*Leiden Institute of Advanced Computer Sciences, Leiden University, The Netherlands*
[c]*Plymouth University, United Kingdom*

Abstract

Multi-disciplinary optimisation of building spatial designs is characterised by large solution spaces. Here two approaches are introduced, one being super-structured and the other super-structure free. Both are different in nature and perform differently for large solution spaces and each requires its own representation of a building spatial design, which are also presented here. A method to combine the two approaches is proposed, because the two are prospected to supplement each other. Accordingly a toolbox is presented, which can evaluate the structural and thermal performances of a building spatial design to provide a user with the means to define optimisation procedures. A demonstration of the toolbox is given where the toolbox has been used for an elementary implementation of a simulation of co-evolutionary design processes. The optimisation approaches and the toolbox that are presented in this paper will be used in future efforts for research into- and development of optimisation methods for multi-disciplinary building spatial design optimisation.

Keywords: Building optimisation, Multi-disciplinary optimisation, Super-structures, Structural design, Building physics

1. Introduction

Many engineers in the built environment experience optimisation as a challenging task. This is because it is usually a time consuming trial-and-error procedure, in which knowledge and experience are first needed to create designs, which are then assessed and possibly modified. Many research projects involve the development of optimisation methods to create and analyse designs to aid engineers. These developments concern advanced optimisation methods, often specialised to small sub problems (for a single discipline) in the design process. Such a specialisation exists because building spatial design problems are too large for a single design tool. Engineers are therefore invaluable to the design process since their experience can reduce a design problem drastically. However, it cannot be expected that an individual engineer oversees the complete design problem, and thus complex relationships between the disciplines might go unnoticed, leading to suboptimal designs. For this, multi-disciplinary building optimisation could be supportive, but it needs a method to handle the large design search spaces involved. This paper aims at the development of

*Corresponding author
Email addresses: `s.boonstra@tue.nl` (S. Boonstra), `k.van.der.blom@liacs.leidenuniv.nl` (K. van der Blom), `h.hofmeyer@tue.nl` (H. Hofmeyer), `m.t.m.emmerich@liacs.leidenuniv.nl` (M. T. M. Emmerich)

such a method by means of a toolbox that is presented here and asks the question of how to represent design search spaces such that optimisation methods find efficient solutions. This paper is an extension of [1], in addition to the contribution in [1] (a consideration and proposition for building spatial design optimisation) this paper discusses: a toolbox for building spatial design optimisation; and a toolbox demonstration.

Prior to reading this paper it is important to understand the terminology concerning optimisation and data structures in optimisation. Optimisation aims to minimise or maximise an objective value by the variation of design variables, while at the same time satisfying certain constraints. What is important for optimisation is the representation of the design search space, which is the selection of design variables that are used to parametrise the solutions for the problem (design variables not part of the selection are constant or depend on the representation itself). The representation affects the possibilities and performance of the optimisation methods, e.g. a complex dynamic data structure might be too difficult to handle by most types of optimisation methods. In this paper, terminology will be used as found for optimal process synthesis in chemical engineering, where super-structure representations are distinguished from super-structure free representations [2]. In a super-structure, the design search space has a fixed number of design variables, meaning all design alternatives are pre-encoded, which makes for a static data structure. This enables the search for an optimum in a systematic manner by using classical parameter-based optimisation methods. Super-structure free optimisation uses a design search space in which new design variables may originate or disappear, which can be seen as a dynamic data structure. Such a design search space allows for discovering unexpected new alternatives that were not pre-encoded. Typically, super-structures allow for formulating optimisation problems in the language of mathematical programming (using equations and inequalities). Free representations are formulated differently, for instance by describing initialisation procedures and variation operators that form the design search space. The difference between super-structure versus super-structure free approaches is a recurrent theme in specific fields of optimisation [2], whereas this topic has hardly been addressed for building design.

The design search space used in this paper entails the layout and dimensioning of building spaces, i.e. the building spatial design. For this design search space, a super-structure and a super-structure free approach have been developed and compared. Moreover, a method to carry out transformations between the two representations will be discussed, which is envisioned to enable both approaches to efficiently cooperate on a large design space. Finally a toolbox is presented, which is created to develop and investigate different methods of building spatial design optimisation.

2. Related work

In the literature, research on building optimisation can be found that takes into account objectives concerning energy consumption, as is carried out in [3, 4]; structural design in [5, 6, 7]; construction costs in [8]; and thermal building design [9, 10]. Also, optimisation is thoughtfully combined with Building Information Modelling [11, 12, 13]. Different energy performance criteria are combined in [14, 15].

A commonly used optimisation method is evolutionary optimisation, where design variables are stored in a so called genome that can be modified by means of mutation and recombination operators. Other optimisation methods are applied as well, like gradient-based optimisation for topology optimisation in [16], or the analytical derivation of optimal truss layouts in [17]. The use of optimisation methods for building performance optimisation is however still not widespread and many issues need to be solved. One difficulty is to allow for more degrees of freedom in the optimisation. This is addressed in this paper by defining design search space representations that allow for variations of the (global) building spatial design.

The super-structure terminology finds its origins in the process industry, where the optimal configurations of chemical engineering plants are sought. For example, Jackson [18] described the structure of flow configurations of chemical reactors with a super-structure, although without explicitly mentioning the term. Various recent works [19, 20, 21] use the terminology for other engineering fields too. A super-structure prescribes the possible design alternatives to be considered in optimisation, which results in a selection of alternatives. This limited and fixed number of alternatives improves the chance of finding the global optimum. A super-structure enables an optimisation problem to be solved by mathematical programming, for which standard solvers exist (e.g. [22]).

Super-structure free optimisation has been suggested to overcome the limitations of super-structures for designing chemical process configurations. Emmerich et al [23] propose to use replacement, insertion, and deletion rules to modify (mutate, recombine) designs in evolutionary algorithms. However, the development of these local modification operators requires domain knowledge. Voll et al. [2] suggest a more general framework that uses generic replacement rules in evolutionary algorithms. A similar strategy is followed in [24], where it is exemplified for the optimisation of decision diagrams. Other examples of super-structure free design spaces include the work found in [25, 6]. There are only a few optimisation methods that can handle super-structure free representations, namely simulated annealing, evolutionary algorithms, and heuristic local searches. Simulated annealing has been used in the design of processes, e.g. in [26]. In the field of structural design, [27] describes a super-structure free approach in the optimisation of structural topologies. Moreover, in [28] simulations of a co-evolutionary design process (these simulations can also be interpreted as asymmetric subspace optimisation [29]) are used to find a building spatial design for which a structural design created by certain design rules shows minimal strain energy.

3. Building optimisation representations

A building spatial design representation determines—to a large extent—the design space of the building spatial design problem. Designs can be constrained by how they are represented e.g. a representation that is restricted to orthogonal shapes cannot represent curves in a building design. Optimisation efficiency and success is dependent on the solution space (i.e. design space), therefore it is important to consider the used representation for building design optimisation. In this section two representations are suggested, the supercube representation and the movable and sizeable representation, which are based on the super-structured and the super-structure free approaches respectively.

3.1. Super-structure based representation

Design search space. A supercube (SC) is introduced to describe a building spatial design B by means of a super-structure design search space representation. A supercube consisting of cells is described by four vectors: $\mathbf{w}, \mathbf{d}, \mathbf{h}, \mathbf{b}$. Equation 1 shows the variables used. Here $\mathbf{b}$ describes the existence of the cell with indices i, j and k in space ℓ, where $b^{\ell}_{i,j,k}$ with a value "1" means the cell i, j, k is active and describes a part of space ℓ while "0" means the cell is inactive. A space ℓ can thus be constructed out of the supercube cells that are activated for that space. Finally, w_i, d_j and h_k describe the continuous dimensioning of the supercube's cells. The entire supercube is used to perform design modification, therefore the complete design space is described by the vectors $\mathbf{w}$, $\mathbf{d}$, $\mathbf{h}$ and $\mathbf{b}$. Figure 1 shows the supercube notation for an example building spatial design. Building spaces are indicated by normal lines (and coarsely dashed hidden lines), whereas cells can be recognised by finely dotted lines. Each cell in the figure has a number in the left front corner that indicates the building space it belongs to.

$$
\begin{aligned}
&\mathbf{w}\,\{w_1, w_2, w_3, w_4\}\,,\\
&\mathbf{d}\,\{d_1, d_2\}\,, \mathbf{h}_k\,\{h_1\}\,,\\
&\mathbf{b}\left\{\mathbf{b}^1, \mathbf{b}^2, \mathbf{b}^3, \mathbf{b}^4\right\}\\
&\qquad \mathbf{b}^1\,\{1,0,0,0,0,0,0,0\}\,,\\
&\qquad \mathbf{b}^2\,\{0,0,1,0,1,0,0,0\}\,,\\
&\qquad \mathbf{b}^3\,\{0,1,0,0,0,0,0,0\}\,,\\
&\qquad \mathbf{b}^4\,\{0,0,0,1,0,1,0,0\}
\end{aligned}
$$

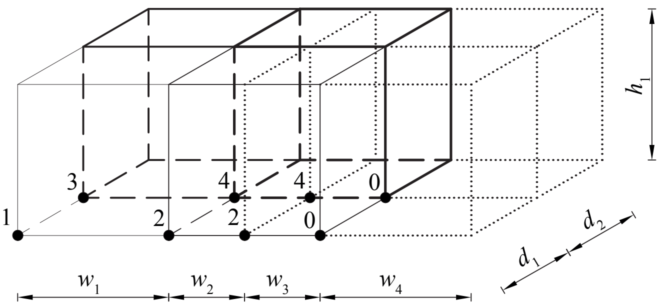


Figure 1: Supercube representation of a building spatial design, space 2 and 4 are described by two cells each, the two right cells are not used to describe a room

$$
\begin{array}{lll}
i \in \{1, 2, ..., N_w\} & w_i & \in \mathbb{R}\\
j \in \{1, 2, ..., N_d\} & d_j & \in \mathbb{R}\\
k \in \{1, 2, ..., N_h\} & h_k & \in \mathbb{R}\\
\ell \in \{1, 2, ..., N_{spaces}\} & b^{\ell}_{i,j,k} = & \begin{cases} 1, & \text{if cell } i,j,k \in \text{space } \ell \\ 0, & \text{otherwise} \end{cases}
\end{array} \tag{1}
$$

Constraints and design modification. Building spatial design modification is performed by re-assigning cells to building spaces through changes of the binary variables and by modifying distance values of the supercube's grid. Constraints are introduced to the design search space so the search can focus on physically and technically feasible solutions. Constraints can be checked by algorithms or, when stated as equations, they can be part of the selection and generation of solutions. Stating constraints as equations has the advantage that their algebraic structure can be exploited by the employed optimisation algorithms. The supercube representation is suitable for such algebraic expression of constraints, three constraints are presented here to demonstrate this suitability. The expressions enable the use of mathematical programming techniques like mixed integer non-linear programming (MINLP) which contribute to the efficiency of the optimisation. It should be noted that there may be differences between constraint representations and constraint implementations (not shown here). For example only "1"-values in binary variables are stored in memory to avoid inefficient constraint checking by large zero spaces in vector **b**.

Condition 1: Non Overlap Overlaps of building spaces are not allowed since they are not practical and might cause erroneous results in subsequent design analysis. This needs to be checked because every space is represented by a separate bit-mask (enumerated by ℓ) of all cells in the supercube, thus non-overlap is not automatically prevented in the representation. Equation 2 achieves this by taking the sum of each cell over all masks. As a result of the binary representation, only if such a sum is smaller or equal to one, no overlap exists at that position.

$$
\forall_{i,j,k} \sum_{\ell=1}^{N_{spaces}} b^{\ell}_{i,j,k} \leq 1 \tag{2}
$$

Condition 2: Cuboid Spaces are constrained to cuboid shapes for practicality and to delimit the design space to a manageable size. To check this condition by means of an equation, first the supercube will be extended with a single layer of cells all around, and these new cells will be set to have no relation to any space ("0"), the latter is described by equation 3:

$$\begin{aligned} &\forall_{\ell} : \forall_{i,j,k} \in \{0, ..., N_w + 1\} \times \{0, ..., N_d + 1\} \times \{0, ..., N_h + 1\} : \\ &i = 0 \lor j = 0 \lor k = 0 \lor i = N_w + 1 \lor j = N_d + 1 \lor k = N_h + 1 \Rightarrow b^{\ell}_{i,j,k} = 0 \end{aligned} \tag{3}$$

Then for each building space ℓ, in each direction (x, y, and z) pairs of adjoining lines that run through the middles of the cells are imagined (e.g. for the z-direction a pair would be a line through all cells $i_1 = 2$, $j_1 = 2$ and a line through all cells $i_2 = 2$, $j_2 = 3$). Moving along a pair of lines, $b^{\ell}_{i,j,k}$ values are processed as shown in equation 4 for the z-direction (as an example, of course all directions should be studied). To obtain a cubic building space, if there is a change from zero to one in the binary string it should occur at the same position (k-value) for both lines. Otherwise in the equation the sums as shown will hold different values and the difference will be non-zero. The same should hold for changes from one to zero, as seen in the second part of the equation. Note that equation 4 allows for the occurrence of multiple changes from one to zero and from zero to one. In other words a space could be cuboid, however could still have internal voids, e.g. a courtyard. Therefore condition 3 is introduced next.

$$\begin{aligned} &\forall_{\ell} : \\ &\forall_{i_1,j_1,i_2,j_2} : \left(\left(\sum_{k=1}^{N_h} k \left(1 - b^{\ell}_{i_1,j_1,k-1}\right) b^{\ell}_{i_1,j_1,k} \right) - \left(\sum_{k=1}^{N_h} k \left(1 - b^{\ell}_{i_2,j_2,k-1}\right) b^{\ell}_{i_2,j_2,k} \right) \right) \left(\sum_{k=1}^{N_h} b^{\ell}_{i_1,j_1,k} \right) \left(\sum_{k=1}^{N_h} b^{\ell}_{i_2,j_2,k} \right) = 0 \\ &\forall_{i_1,j_1,i_2,j_2} : \left(\left(\sum_{k=1}^{N_h} k \left(1 - b^{\ell}_{i_1,j_1,k+1}\right) b^{\ell}_{i_1,j_1,k} \right) - \left(\sum_{k=1}^{N_h} k \left(1 - b^{\ell}_{i_2,j_2,k+1}\right) b^{\ell}_{i_2,j_2,k} \right) \right) \left(\sum_{k=1}^{N_h} b^{\ell}_{i_1,j_1,k} \right) \left(\sum_{k=1}^{N_h} b^{\ell}_{i_2,j_2,k} \right) = 0 \end{aligned} \tag{4}$$

Condition 3: Ortho-Convexity This condition enforces spaces to have a connected, ortho-convex shape. Note that, like condition 2, this also relies on the layer of "zero" cells as described by equation 3. With equation 5 the sum is taken of the number of times a change occurs from cell values zero to one in a building space for each direction. Any building space where there are multiple changes from zero to one is not fully connected and therefore invalidated. Note, that in conjunction with condition 2 this ensures that building spaces have a fully occupied cuboid shape.

$$\begin{aligned} &\forall_{\ell} : \\ &\forall_{i,j} : \sum_{k=0}^{N_h} \left(1 - b^{\ell}_{i,j,k}\right) b^{\ell}_{i,j,k+1} \leq 1 \quad \forall_{i,k} : \sum_{j=0}^{N_d} \left(1 - b^{\ell}_{i,j,k}\right) b^{\ell}_{i,j+1,k} \leq 1 \quad \forall_{j,k} : \sum_{i=0}^{N_w} \left(1 - b^{\ell}_{i,j,k}\right) b^{\ell}_{i+1,j,k} \leq 1 \end{aligned} \tag{5}$$

3.2. Super-structure free based representation

Design search space. A movable and sizeable (MS) representation for spaces is introduced for the super-structure free design space representation. For this, a building is described with a vector **s** that lists all the spaces. This vector is described by equation 6, in which s_i represents a space, C the coordinates of the space origin and D the geometry of the space with w, d and h the width in x-, depth in y-, and height in z-direction, respectively. Figure 2 shows the building spatial design of figure 1 in the movable and sizeable representation.

$$\mathbf{s} = \left\{ s_1, s_2, ..., s_{N_{spaces}} \right\} \qquad \text{where} \qquad s_i = [C, D]\,; \qquad C = [x, y, z]\,; \qquad D = [w, d, h] \tag{6}$$

$s_1 \left\{ \{x_1, y_1, z_1\}, \{w_1, d_1, h_1\} \right\}$
$s_2 \left\{ \{x_2, y_2, z_2\}, \{w_2, d_2, h_2\} \right\}$
$s_3 \left\{ \{x_3, y_3, z_3\}, \{w_3, d_3, h_3\} \right\}$
$s_4 \left\{ \{x_4, y_4, z_4\}, \{w_4, d_4, h_4\} \right\}$

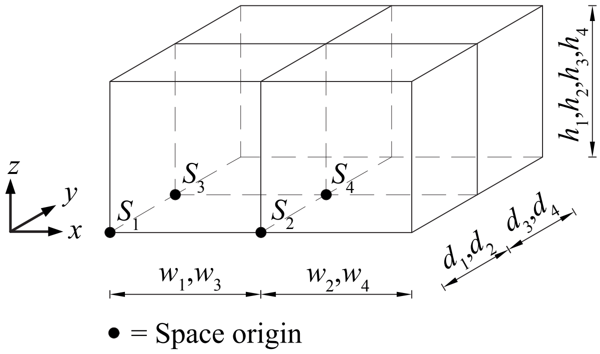


Figure 2: Movable and sizeable representation of the building spatial design (first shown in figure 1)

Constraints and design modification. The definition of spaces by location and dimensions allows an engineer to imagine the spatial properties of the space, the engineer can therefore intuitively define additional properties or modifications for that space. This intuitivity does however not count for the building design itself, as relationships between spaces are defined implicitly. The movable sizable (MS) representation is thus most suitable for design modifications that operate on spaces rather than the entire building design, given that such operations do not interfere with possible relations between spaces. In the super-structure free approach, constraints are implicitly enforced by using design modifications that naturally follow the constraints. Here, this is carried out via removal, scaling and division of spaces. As an example, a modification of the building spatial design in figure 2 will be performed. Assume that after (e.g. structural or building physics performance) analyses, it is concluded that building space S_3 performs least well and thus could better be removed as shown in equation 7. Accordingly, the remaining spaces are scaled (equation 8) to restore the initial volume (V_0) of the building design. To restore the number of spaces, hereafter a (e.g. randomly selected) space is divided (equation 9) into two new spaces, resulting in a new spatial design (equation 10). This process is further illustrated in figure 3 and has been used by [28] for real-world optimisation scenarios.

$$\mathbf{s}\left\{s_1, s_2, s_3, s_4\right\} \rightarrow \mathbf{s}\left\{s_1, s_2, s_4\right\} \tag{7}$$

$$\mathbf{s} \rightarrow \mathbf{s} \cdot \sqrt{\frac{V_0}{V}} \tag{8}$$

$$s_1 \left\{ \{x_1, y_1, z_1\}, \{w_1, d_1, h_1\} \right\} \rightarrow \begin{cases} s_5 \left\{ \{x_1, y_1, z_1\}, \left\{ \frac{1}{2} w_1, d_1, h_1 \right\} \right\} \\ s_6 \left\{ \left\{ x_1 + \frac{1}{2} w_1, y_1, z_1 \right\}, \left\{ \frac{1}{2} w_1, d_1, h_1 \right\} \right\} \end{cases} \tag{9}$$

$$\mathbf{s}\left\{s_2, s_4, s_5, s_6\right\} \tag{10}$$

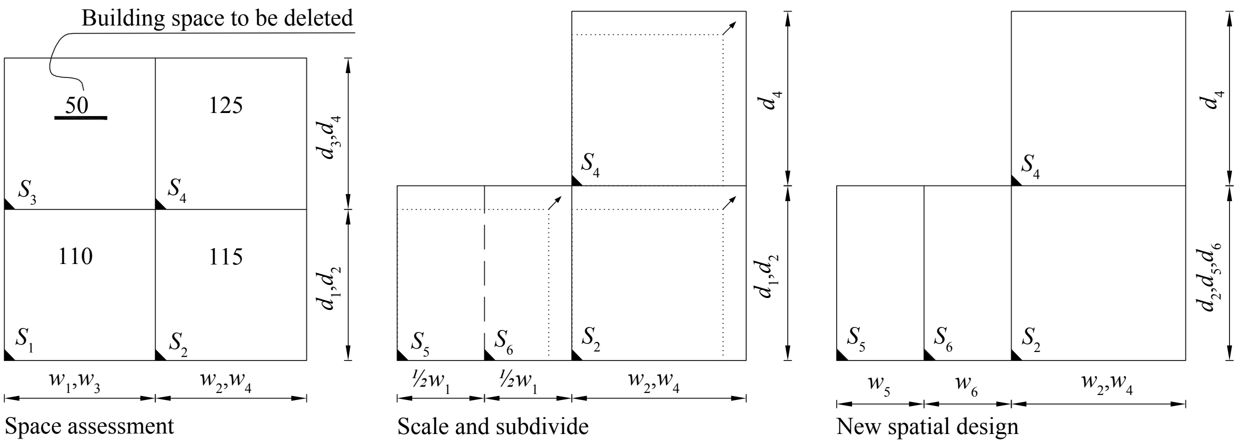


Figure 3: Super-structure free modification, numbers in spaces in the most left figure represent performances of spaces (e.g. structural or building physics)

3.3. Discussion

So far two design space representations have been defined for building spatial design optimisation: one suitable for the super-structure approach and another for the super-structure free approach. This subsection discusses the properties of the two approaches on a conceptual level with reference to the two presented representations. From the super-structure based representation it becomes clear that its use requires expertise in the fields of mathematics, optimisation, and the built environment. This requirement should not however exclude building engineers from using this representation, because it can lead to the optimum design with a high confidence level. Additionally it can lead to new design insights when multiple solutions are assessed, e.g. relationships between design variables may be discovered. However, a design search space representation draws a limit on which solutions can be considered by an optimisation algorithm. For the super-structure approach, this means all solutions are pre-defined by the engineer who developed the representation. This means that an optimum is only the best out of the pre-defined solutions, and better solutions outside the design space representation will never be found. A larger design space representation could solve this issue, but will almost always lead to a significant increase of computational time, and this without a prior guarantee of better optima.

The super-structure free based approach to building optimisation can be developed even when only expertise of the built environment is available. Rules for modification of the considered design are then based on knowledge and experience in the field. This approach can combine design variables in (mathematically) unexpected ways and may therefore lead to new building designs that would otherwise not have been considered. It also provides a fast way to navigate a large design space, since it is not an exhaustive search of the entire design search space. The approach rather is a selection of other interesting parts of the design search space based on engineering knowledge and experience. However, this dynamic approach prevents the use of many classical search algorithms (global and parameter based search) and instead heuristic rules should be used to navigate the design space. Such heuristics are prone to find local optima and cannot provide high levels of confidence concerning these optima (although comparisons between heuristics and global searches sometimes result in matching results). Compared to the super-structured approach, new design insights are more difficult to find when using heuristics, because fewer solutions are analysed and design evolution follows a path that is defined by the heuristics.

To consider large design spaces, it can be concluded that both approaches are eligible, although both have disadvantages as well: The super-structured approach is too costly in terms of computational effort and the super-structure free approach cannot provide the optimum with a high level of confidence. Therefore it is proposed to combine both approaches. Additionally, such a combination could enable the optimisation to discover both surprising designs and new design insights.

The presented representations are—in combination with the presented constraints—limited to only cuboid spaces. Releasing the cuboid spaces constraint will allow more complex spaces, which is desirable in real world design scenarios. This is possible with both representations, although the SC representation would require a redefinition of some of the constraints and the MS representation requires a space to be defined as a collection of subspaces. This is however not implemented in the toolbox to avoid the additional complexity in the toolbox as it would distract from the focus of this research, namely to research and develop optimisation methodologies.

In this paper each approach, super-structured and super-structure free, is supplemented with one representation each. It could be questioned if other representations are also suitable or in some aspects even better for the proposed optimisation approaches. The above mentioned limitations might then be lifted. An extensive study into such alternative representations could also lead to well argued choices for specific representations. Additional representations are however not considered for this paper as the presented representations are sufficient for the objectives of this research and are therefore considered good. Moreover, an extensive study would both elaborate and distract from the before mentioned focus of the research.

3.4. Combination of super-structured and super-structure free approaches

The combination of the approaches above is proposed by alternately employing each approach during the optimisation process for the same problem. This alternation requires mutual transformation between the two representations. To enable this, two algorithms have been developed which are presented in this subsection.

Supercube to movable and sizeable. To transform a building spatial design's supercube representation into the movable and sizeable representation, it is suggested here to first find the smallest and largest indices i, j, k for the set of cells describing each space ℓ as shown in equation 11. Space coordinates x, y, z can then be found as shown in equation 12, with the notion that if the smallest index equals 1, there is no term in the sum, and the degenerated sum is evaluated as 0 (which is appropriate here). The space dimensions are computed in a similar way using the minimum and maximum indices as shown in equation 13.

$$\begin{aligned} i^\ell_{min} &= \min(\{i \mid b^\ell_{i,j,k}\}) & i^\ell_{max} &= \max(\{i \mid b^\ell_{i,j,k}\}) \\ j^\ell_{min} &= \min(\{j \mid b^\ell_{i,j,k}\}) & j^\ell_{max} &= \max(\{j \mid b^\ell_{i,j,k}\}) \\ k^\ell_{min} &= \min(\{k \mid b^\ell_{i,j,k}\}) & k^\ell_{max} &= \max(\{k \mid b^\ell_{i,j,k}\}) \end{aligned} \tag{11}$$

$$x^\ell = \sum_{p=1}^{i^\ell_{min}-1} w_p, \qquad y^\ell = \sum_{q=1}^{j^\ell_{min}-1} d_q, \qquad z^\ell = \sum_{r=1}^{k^\ell_{min}-1} h_r \tag{12}$$

$$w^\ell = \sum_{i=i^\ell_{min}}^{i^\ell_{max}} w_i, \qquad d^\ell = \sum_{j=j^\ell_{min}}^{j^\ell_{max}} d_j, \qquad h^\ell = \sum_{k=k^\ell_{min}}^{k^\ell_{max}} h_k \tag{13}$$

Movable and sizeable to supercube. A transformation from movable and sizeable to supercube first requires three steps to compute the supercube dimensions **w**, **d**, **h**. Step one—for each space—the minimum and maximum coordinate values should be found, i.e. for each space: $\{x, x + w\}$; $\{y, y + d\}$; $\{z, z + h\}$. Step two, all these values are grouped into three lists (each for either x, y or z values), duplicate values are removed, and then each list is sorted in ascending order. Finally in the third step, vectors **w**, **d**, **h** are computed from these lists. For example, **w** is computed as $w_i = x_{i+1} - x_i$ for every $i \in [1, ..., n - 1]$ where n is the number of values stored in the sorted list.
Regarding vector **b**, for each space ℓ and for each cell i, j, and k the (derived) cell's coordinates are compared with the coordinates of the considered space. A cell is assigned to the considered space if the cell coordinates are completely within the coordinates of the space, e.g. for the x-direction if: $x_{space} \leq x_{cell} < x_{space} + w_{space}$.

Validation. The above algorithms have been validated in [1] for overlaps in spaces, non-connected spaces, truncation errors, alterations in space identification, and fragmented spaces. Although truncations and fragmented spaces may cause changes during the transformation it was found that these errors will not occur or are insignificant.

4. Building analysis toolbox

A toolbox to evaluate building spatial designs has been developed in the form of a C++-library. This library forms an environment in which building spatial design optimisation can be developed and researched. The toolbox currently contains the following: structural design analysis, building physics analysis, spatial design representations and a visualisation of these. Figure 4 shows the UML class diagram of the toolbox plus the modules that a user should still define, the toolbox's visualisation is omitted for brevity and clarity. The diagram shows that a user should define an optimisation method but also the so-called design grammars. These grammars generate domain specific information that is required to evaluate the objective functions in that domain. A grammar will as such take a building spatial design as input to generate domain specific information based on user defined design rules. The toolbox can be expanded to other disciplines as well by introducing new grammars, for example monetary or environmental costs could be included by implementing design rules to compute a model to calculate these costs for a building spatial design. This section first discusses the building spatial design representations then structural- and building physics design analysis in the toolbox and finally a benchmark is presented.

4.1. Spatial design

The spatial design package consists of three main parts, namely the models of the MS-representation and the SC-representation but also a conformal model. Here a conformal model is the representation of a building design in which geometry entities like line segments, rectangles or cuboids do not intersect with each other, but their vertices are allowed to coincide. For example when two walls are connected by a T-joint then the continuous wall is split into two rectangles at the intersecting wall, see figure 5. This and similar splitting procedures are repeated in the conformation process until all intersections between spaces, surfaces, and line segments are represented in a model of smaller geometry entities. A conformal model is useful because domain relevant relationships vary over building edges, walls or spaces. For example, two walls with a T-joint connection will in a finite element model only be structurally connected if the nodes—at the joint—of both walls coincide. The conformation procedure enables a structural grammar to find such a joint so an appropriate design can be created accordingly.

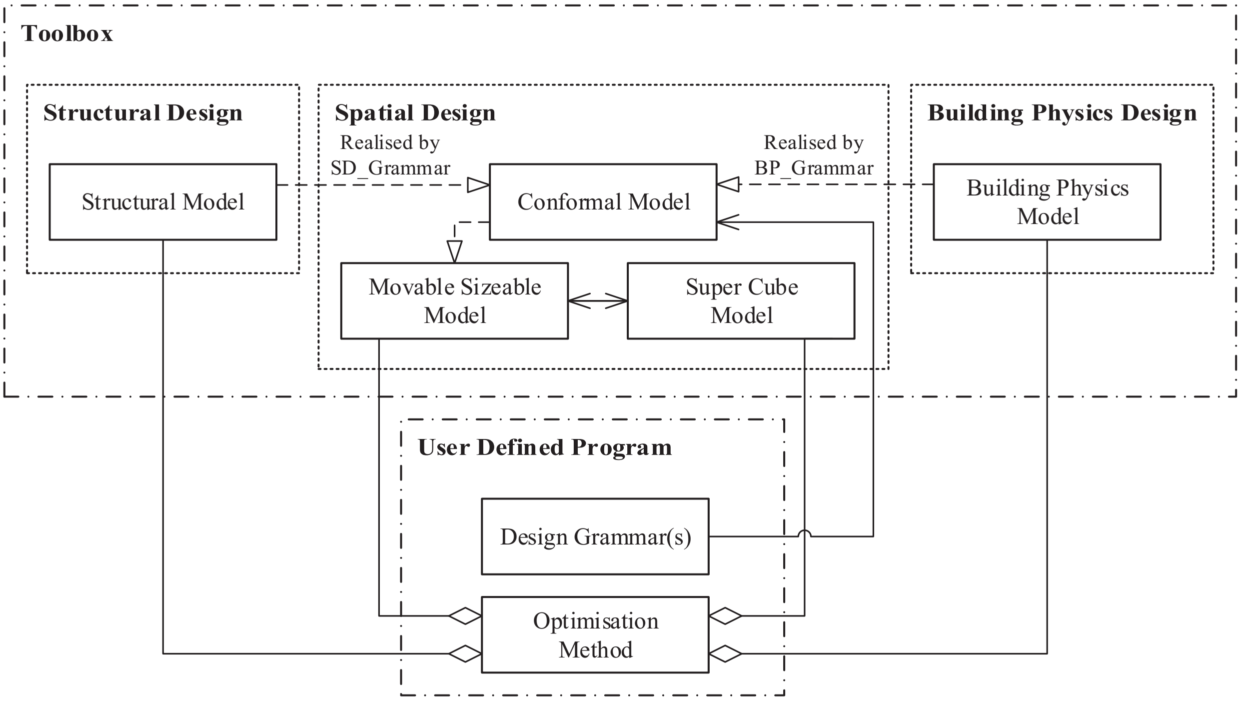
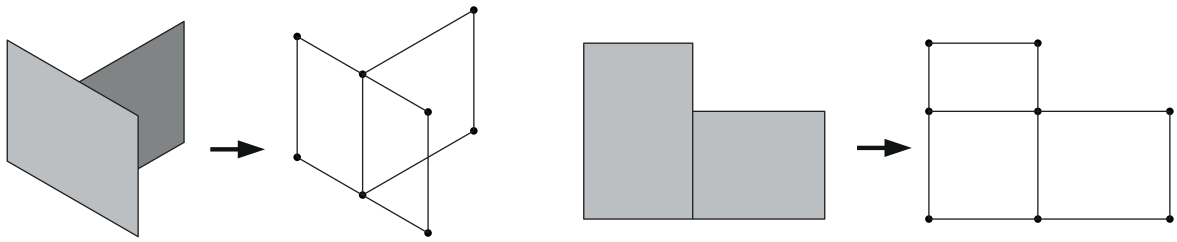


Figure 4: UML class diagram of the toolbox and the user defined modules



T-joint of surfaces **Varying dimensions of surfaces**

Figure 5: Examples of non conformal surfaces that can be represented in a conformal model by geometry entities like vertices, lines, and rectangles

Building representations. Both the SC- and MS-representation have been implemented in separate classes, as illustrated in figure 6. Conversion in either direction between the SC- and MS representations is implemented within those classes as well.

Figure 6: UML class diagram of the movable sizeable and the supercube building representation classes

Conformation. The conformal building model class is elaborated in figure 7, the subclasses that form the conformal model class are grouped into geometry entities and building design entities. Building design entities describe the topology of a building spatial design of the conformal model based on a spatial design in the MS-representation (figure 4). Geometry entities describe a building spatial design in a geometry model such that it is completely conformal. It is important to distinguish between the two because domain specific properties can depend on both geometric relations and building design relations. For example when a wind load is acting on a building wall that is described by multiple rectangles, then the rectangles are used to generate structural slabs, but the wall's surface information is used to find the loads acting on these slabs. Geometry entities and building design entities are realised with lower dimensional entities and they are associated with the higher dimensional entities within their typology (i.e. design or geometry), e.g. a rectangle is realised with four line segments that on their turn are realised with two vertices each and also an association from that rectangle to one or more cuboids is made. Finally, relations between corresponding design and geometry entities are stored, e.g. a surface is associated to the rectangles that describe its conformal geometry and all surfaces that are described by a specific rectangle are associated to that rectangle. Adding and maintaining the mapping of figure 7 during the conformation of a design prevents an iterative search for relevant relationships between geometry and building design entities.

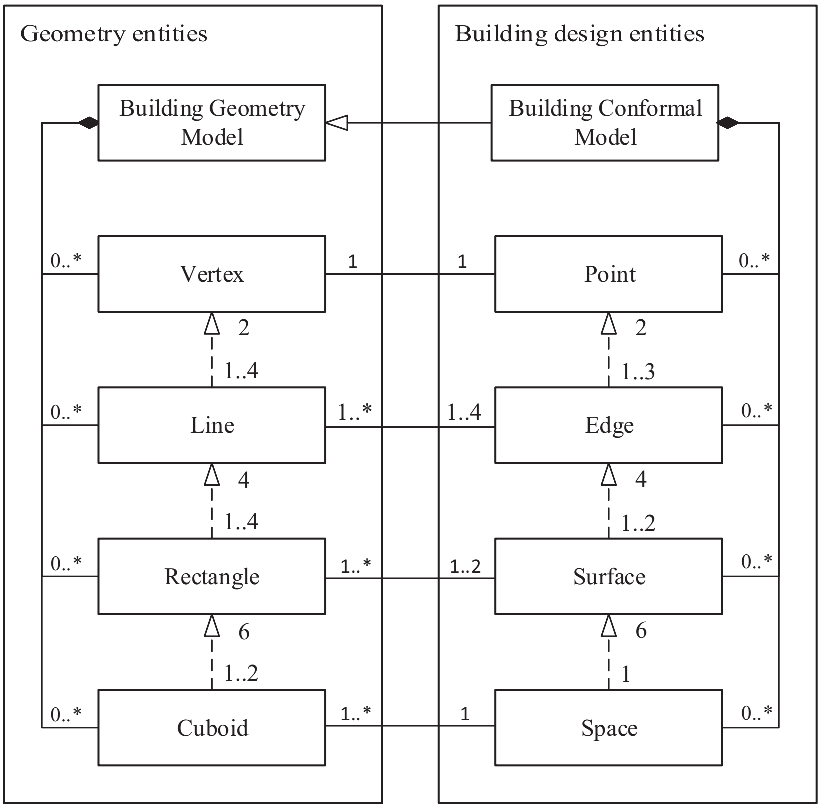


Figure 7: UML class diagram of the (orthogonal) conformation model

Conformation can be started after a conformal model is initialised with all the building design entities, which can be derived from a building spatial design in the MS-representation. While initialised, each building design entity is provided with one corresponding geometry entity and all relevant relationships between those entities are mapped subsequently. Conformation then starts with a search in the geometry model for intersections between line segments and rectangles and other line segments, a vertex is added to the geometry model if such an intersection exists, see figure 8. Accordingly the cuboids, rectangles, and line segments in the geometry model are checked with all the vertices in that model. When a vertex lies within a cuboid, rectangle or line segment then immediately this geometry entity is split at the location of the vertex by a splitting algorithm, see figure 8 for the example of a line-line intersection. A splitting algorithm provides new geometry entities while updating all the relational mappings that were held by their parent and its associated entities, the parent is then tagged for deletion. It should be noted that a new geometry entity is only added to the geometry model when geometrically unique within the model, the relational mapping of the parent is in that case updated to the mapping of the already existing entity. Splitting of geometry entities invokes a recursion because new vertices can be created when an entity is split (figure 8). These new vertices are first checked with all associated entities, which can be found by using the mapped relationships of the split entity. When new intersections are found while splitting a geometry entity then these will first be split, thereby a recursion of splitting algorithms is invoked in the conformation process. Geometry entities that were tagged for deletion during the conformation process are deleted after all geometry entities have been checked for intersecting vertices.

find intersections:

$$\mathbf{p}_5 = \mathbf{p}_1 + t \cdot (\mathbf{p}_2 - \mathbf{p}_1) \quad \text{if} \quad \begin{cases} (\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_4 - \mathbf{p}_3) = 0 \\ 0 < t < 1 \\ 0 < u < 1 \end{cases}$$

$$\text{with} \quad \begin{cases} t = \dfrac{(\mathbf{p}_3 - \mathbf{p}_1) \times (\mathbf{p}_4 - \mathbf{p}_3)}{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_4 - \mathbf{p}_3)} \\ u = \dfrac{(\mathbf{p}_1 - \mathbf{p}_3) \times (\mathbf{p}_2 - \mathbf{p}_1)}{(\mathbf{p}_4 - \mathbf{p}_3) \times (\mathbf{p}_2 - \mathbf{p}_1)} \end{cases}$$

split geometry:

old lines: $\{\{\mathbf{p}_1, \mathbf{p}_2\}, \{\mathbf{p}_3, \mathbf{p}_4\}\}$

new lines: $\{\{\mathbf{p}_1, \mathbf{p}_5\}, \{\mathbf{p}_2, \mathbf{p}_5\}, \{\mathbf{p}_3, \mathbf{p}_5\}, \{\mathbf{p}_4, \mathbf{p}_5\}\}$

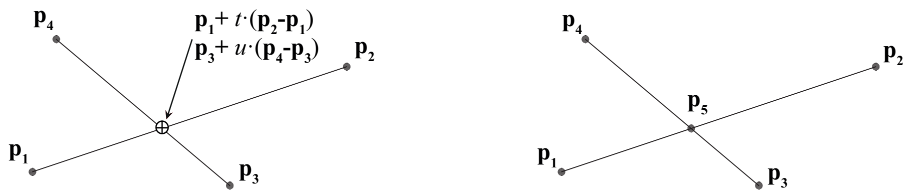


Figure 8: Splitting of a line, first intersections are found then geometries are split. Rectangles and cuboids have similar procedures

4.2. Structural design

The structural design of a building is here an assembly of structural components, loads, and boundary conditions, e.g. columns; beams; slabs; wind loads; floor loads; and the constraints that are imposed by a foundation (in a respective order). A structural design of a building needs to be evaluated on structural safety by assessing the strength, stiffness, and stability in the design. Such an evaluation can for example be carried out analytically or by means of the commonly used Finite Element Method (FEM). The toolbox employs FEM, in which the structural components of a design are modelled into smaller structural elements, nodal loads, and nodal constraints. The structural stiffness of each element is then derived for each node with respect to the positions of all other nodes in the element. The stiffness terms of each element can then be assembled into a so-called global stiffness matrix $\mathbf{K}$, and together with the nodal loads vector $\mathbf{f}$ and boundary conditions it is used to solve for the nodal displacements vector $\mathbf{u}$ given the equilibrium condition in equation 14.

$$\mathbf{f} = \mathbf{K}\mathbf{u} \tag{14}$$

The optimisation objectives, i.e. the structural responses, can be calculated once vectors $\mathbf{u}$ and $\mathbf{f}$ and matrix $\mathbf{K}$ have been computed. Responses that are traditionally used for structural design evaluation are strains, stresses, reaction forces or the displacements themselves and recently—for optimisation purposes—strain energies are used as well.

Element formulations. Three different element formulations have been implemented for structural design analysis in the toolbox: one for trusses, one for beams and one for flat shell elements. The element stiffness matrix of the truss elements is derived for an element with two nodes, each having three degrees of freedom (ux, uy, uz; with u for displacement and r for rotation) as is presented in [30]. The beam elements use an element stiffness matrix that has been derived for a two node element with each six degrees of freedom (ux, uy, uz, rx, ry, rz). The element formulation—as presented in [31]—accounts for axial forces, bending and torsional moments, and shear forces in two directions. Finally the formulation for a flat shell element is derived for a four node shell element with six degrees of freedom per node (ux, uy, uz, rx, ry, rz). The formulation is a combination of a derivation for in-plane-behaviour as presented in [30] and out-of-plane behaviour [32] for which 2×2 numerical integration (Gaussian quadrature) is used to represent the displacement fields in the elements. Also a drilling stiffness is added to the stiffness matrix, its terms are equal to the mean of all terms in the element stiffness matrix in which the in- and out-of-plane behaviour are already determined. A flat shell element using this formulation will offer resistance to axial forces, a shear force, a torsional moment and bending moments.

Meshing. Meshing is the process of generating a number of finite elements, nodal loads and nodal constraints that together make up the structural components in a structural design. As such each structural component is meshed into a given number of elements or into a given size of elements. The toolbox currently only supports a meshing method based on a given number of elements, in which all structural components in a structural design model are meshed into an equal number of elements in each of their dimensions. This meshing method requires one input variable for meshing, i.e. n for the number of equally sized divisions along each side of an element. The method meshes one dimensional components into a number of elements equal to n, two dimensional components into grid of n^2 elements and three dimensional components into a grid of n^3 elements. Where the grids of the two and three dimensional components are formed by connecting the dividing points on opposite sides to each other. This method is a simple meshing approach but still results in qualitatively good meshes as long as the meshed components stay orthogonal and as long as aspect ratios of component shapes do not become too large (i.e. > 5:1).

Elements, nodes, nodal loads and nodal constraints can be added to the FE-model once a component has been meshed. Elements and nodes are initialised using the meshed points and the properties that are stored for a component. Constraints on a component are simply applied to all nodes that were meshed for that component. Finally loads are also applied to all meshed nodes, however their magnitude should still be determined. This is carried out by splitting each element using the midpoints of line edges and quadrilaterals as shown in figure 9, the division temporarily creates new line segments or areas that are used to determine the magnitude of a load on a node in the element. Loads from different elements that share a common node are summed for that node.

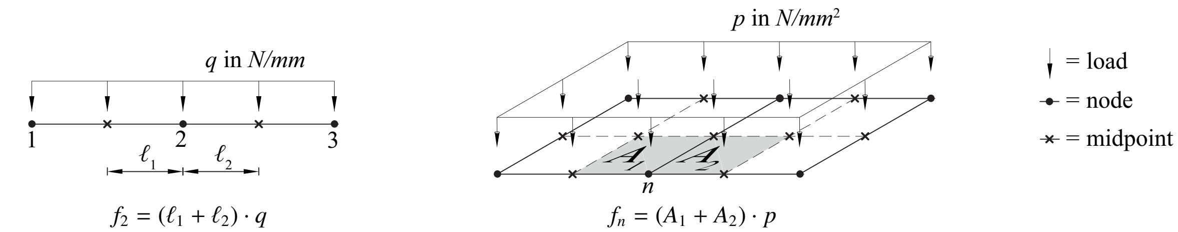


Figure 9: Meshing of loads on nodes that have two line elements in common(left) or two quadrilateral elements in common (right)

Assembly and solving. A number of steps must to be finished before the assembly of the FE-model into the form of equation 14 can start. To begin with is the initialisation of the nodes, where nodes are first checked for duplicity before they are added to the model. Elements are initialised thereafter, this process includes the following steps: associating nodes to the element; ordering of the associated nodes (the order of which is inherent to the derivation of the element formulation); updating which degrees of freedom (DOF's) are active in the FEM-model; and finally determining the value of the stiffness terms in the element stiffness matrix. After all the elements in a component have been initialised, then also the loads and constraints that act on it will be added to the nodes to which they have been meshed. Assembly of the FE-model can begin after all nodes, elements, loads and constraints have been initialised, and starts with indexing all DOF's in the system by iterating over each element's nodal freedom signature. Accordingly each term in each element stiffness matrix can be transformed into triplet form using the global DOF-indices, the complete global stiffness matrix $\mathbf{K}$ is as such defined in sparse form by a collection of triplets. Accordingly the load vector $\mathbf{f}$ is computed by initialising a null vector to the size of the number of DOF's, each load in each node is iterated and added to the load vector using the global indices of the nodal DOF's. Constraints are handled as follows, global stiffness terms that depend on a constrained DOF are replaced with 1.0 if they are on the diagonal (to prevent singular systems) and with 0.0 in any other case, terms in the load vector that act in a constrained DOF are replaced with 0.0.

The toolbox uses the Eigen C++ template library [33] for all linear algebra in the finite element analysis, which provides vector templates, matrix templates, solvers and other linear algebra related algorithms. As such the stiffness matrix and the load vector have been assembled into instances of classes from the Eigen library and accordingly the system can be solved by using one of the solvers in the library.

Topology Optimisation. Another function that has been added to the structural design package is topology optimisation [16]. Topology optimisation aims to minimise an objective—e.g. strain energy—in an FE-model by varying element densities between 0 and 100% while the total available material volume is constrained to a fraction of the total volume of elements. This method leads to structural topologies within an FE-model, figure 10, which are then to be interpreted as a new structural design by either a designer or computer algorithm. An example to illustrate the possible application of topology optimisation using the toolbox that is presented here is presented in [28], where optimised element densities determine the performance of a structural design. Another application is found in a structural design grammar, in which topology optimisation can for example generate a structural design [34].

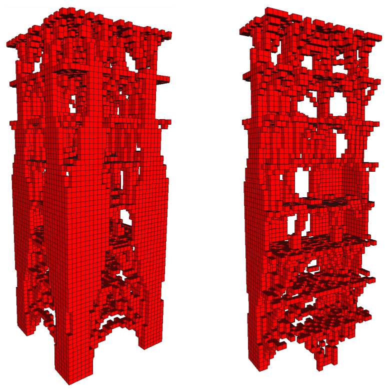


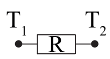
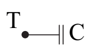
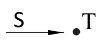
Figure 10: Optimised topology of a solid structural design with live loads at floor heights and wind loading on the surfaces [35]

4.3. Building physics

Building physics is a broad research field, it includes studies in acoustic-, moisture-, insolation-, or thermal behaviour of a building. Building physics analysis in the toolbox is currently limited to only an evaluation of thermal building behaviour. Several different methods can be used to simulate this behaviour, for example the Finite Element Method (FEM), Computational Fluid Dynamics (CFD) or Resistor-Capacitor-networks (RC-networks). Each of these methods are particularly suitable for different levels of detail, however the simulation time and complexity of the method also increase with a higher level of detail. The RC-network approach is used in the toolbox for two reasons, firstly only a low level of detail is required, this is preferred because all information in the building physics model is generated by a design grammar thus more detail would also imply that a more sophisticated grammar is required. Secondly it is fast and thus it can be used to evaluate many designs in a relatively small amount of time, which is relevant for some optimisation methods. In [36] it is investigated how different simulation methods can work together by inversely model (i.e. fitting a model to data) the building thermal design to results from a more complex and detailed model or from real world data. It is concluded that the simple surogate model could still simulate the same results when comparing it with the base model. An RC-network of a building can itself also have different levels of detail, for example phenomena like ventilation or solar irradiation add extra detail to the network. In [37] it is investigated how different levels of detail in an RC-network influence the simulation results. It was concluded that the most simplified RC-network models still simulate results that are close to real world thermal behaviour of buildings. It should be noted that the aforementioned research uses inverse modelling to define the parameters in the RC-networks, as such a direct modelling approach may not yield realistic values. However it can be concluded from the mentioned research that RC-networks do simulate realistic behaviour. These notions are important when real world problems are modelled, but for the building physics designs in the toolbox—that are derived from only a spatial design—a model is not expected to yield realistic quantitative values, but they are expected to yield realistic qualitative behaviour.

The terminology for RC-networks is borrowed from electrical engineering, where voltages and currents are simulated in a network of resistors and capacitors. Electrical components i.e. resistors and capacitors form a network in which each component describes a relationship that can be expressed in differential form (table 1). Thermal building properties can be mapped in a similar fashion, where a resistor is now modelled by the thermal conduction properties- and a capacitor by the heat capacity of the constructions and spaces in the building, see table 1. A system of first order ordinary differential equations (ODE's) can be assembled from the relations that each of the components in the network describe. The system of ODE's can then be used to simulate the dynamic problem that is described by the RC-network by solving the system over a specified simulation time, e.g. by an Eulerian method.

Table 1: RC-network components, the relations describing heat flux Φ_q, and the units for temparature T; heat resistance R; heat capacitance C; time t; and heat irradiation S

Component	Relation	Units
	$\Phi_q = \frac{T_2 - T_1}{R}$	T [K] R [K/W]
	$\Phi_q = C \cdot \frac{dT}{dt}$	C [J/K] T [K] t [s]
	$\Phi_q = S$	S [W]

A building thermal RC-network is here modelled by first defining at which points in a building spatial design the temperature is of interest for the user or computer algorithm. A network is then created by connecting these temperature states to other temperature states based on their geometric relations. Each connection enables a heat flux from one temperature state to another and should be defined with one or more resistances against this flux, i.e. the resistors. The resistance is computed from the heat conduction properties of all material that resists a heat flux between two temperature states, e.g. the insulation or construction in a wall. Capacitors are defined by the heat capacitance of a specific amount of material that is located around a temperature node, e.g. material in a wall or the air inside a space. Different building spatial detail levels can be modelled using this methodology e.g. a single building wall but also a complete building, see figure 11.

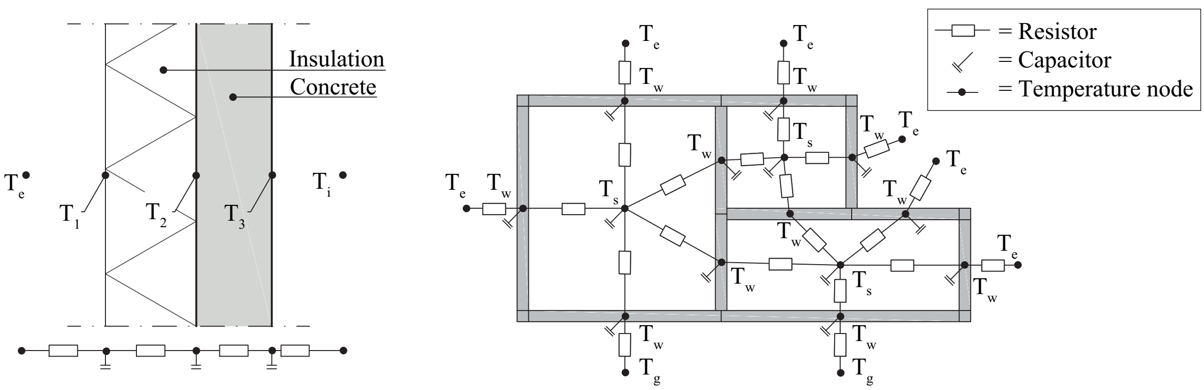


Figure 11: Two different levels of spatial detail for building thermal models using an RC-network model.

System of temperature states. A building physics model in the toolbox is structured in a system of temperature state objects, see figure 12 for the UML class diagram. Here temperature states are specified into two child classes: one to resemble dependent and the other to resemble independent temperature states. Dependent temperature states (e.g. walls, floors and spaces) are simulated, whereas independent temperature states are input (such as weather data) and thus non dependent on the modelled system. Each dependent state is defined with a capacitance, and each association between states is defined with a resistance. The system of state objects can then be translated into a system of ordinary first order differential equations as expressed in equation 15, where **x** are the dependent states, **u** are the independent states, and **A** and **B** describe the system of resistors and capacitors. Additionally, for implementation purposes, two different dependent states—namely building constructions and building spaces—are characterised in the toolbox.

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{u} \tag{15}$$

Building constructions are here (parts of) walls and floors that consist out of one or more layers of material that each have a certain thickness and are represented by one temperature point in the RC-model. In the toolbox a construction is implemented as an aggregation of layers that each consist of a material. The resistance of a construction is not constant over its cross section. Therefore a location within the construction should be selected at which a lumped value for resistances and capacitances is to be determined, see figure 13. In the toolbox this point is by default selected at half the thickness of the modelled construction. A construction's resistances [K/W] from that point to its adjacent temperature states is calculated according to equation 16, where A is the wall's surface in [m^2], j denotes each contributing layer, ℓ thickness in [m], and

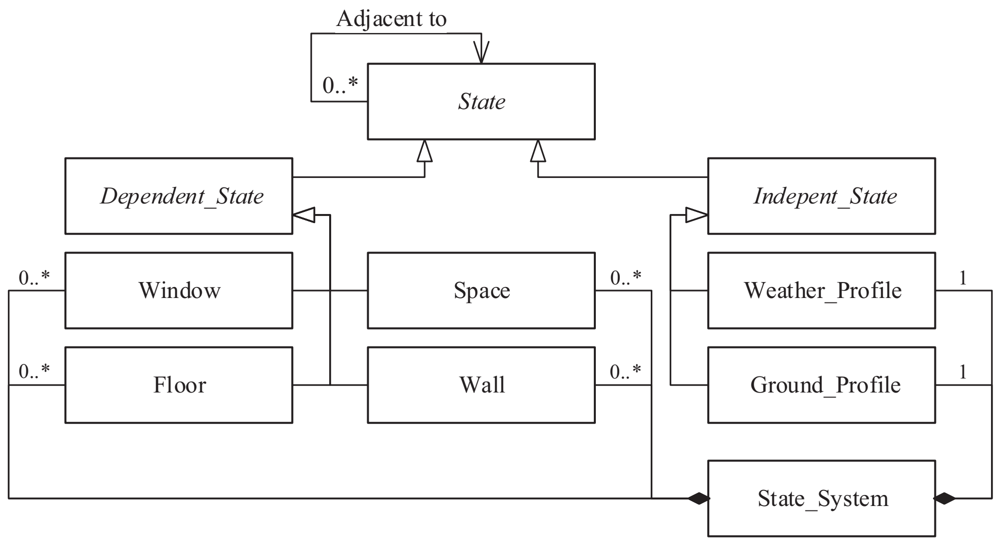
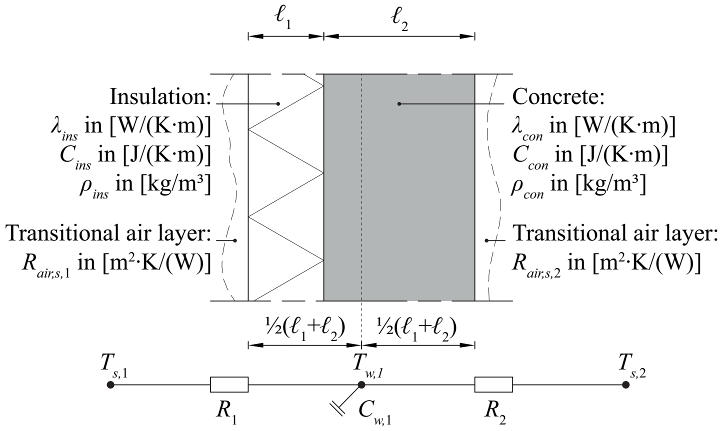


Figure 12: UML class diagram of the building physics package

λ a heat conduction coefficient in [W/(K·m)]. The capacitance of a wall C_w [J/K] is calculated as the sum of the capacitances of each material k in the building construction. This can be obtained following equation 17, where C_k is the specific heat capacity in [J/(kg·K)], and ρ the specific weight in [kg/m^3] of each material. The location of the temperature state over the surface can be left undefined, under the assumption that the capacitances and resistances of a modelled construction are constant over its surface.

$$R = \left(\sum_{j=1} \frac{\ell_j}{\lambda_j} \right) / A \tag{16}$$

$$C = \left(\sum_{k=1} C_k \cdot \rho_k \cdot \ell_k \right) \cdot A \tag{17}$$



$$R_1 = \left(\frac{\ell_1}{\lambda_{ins}} + \frac{\ell_2 - \ell_1}{2 \cdot \lambda_{con}} + R_{air,s,1} \right) / A \quad \text{[K/(W)]}$$

$$R_2 = \left(\frac{\ell_1 + \ell_2}{2 \cdot \lambda_{con}} + R_{air,s,2} \right) / A \quad \text{[K/(W)]}$$

$$C_{w,1} = \ell_1 \cdot A \cdot C_{ins} \cdot \rho_{ins} + \ell_2 \cdot A \cdot C_{con} \cdot \rho_{con} \quad \text{[J/(K)]}$$

·) A is the wall's surface area [m^2]

·) ρ a materials specific weight [kg/m^3]

·) λ is a materials heat conduction coefficient [W/(K·m)]

Figure 13: Calculation example of lumped resistance and capacitance of a construction

Spaces are a special type of dependent temperature state in an RC-network model, because they are strongly influenced by heating, cooling, occupation, and ventilation. Currently heating, cooling and ventilation are accounted for in the simulation program, but thermal loads of e.g. people and equipment are not accounted for. This is to avoid an over-complication in the design grammar for a building physics design, since these loads would require design information such as room function, occupation, and time profiles. Currently only the number of Air Changes per Hour (ACH) and the total available heating and cooling power in spaces have been defined in a constant time profile.

The capacitance C_s of a space i is calculated with equation 18, where C_{air} is the specific heat of air in [J/(K·kg)] (set to 1000 J/(K·kg)), ρ_{air} the specific weight of air in [kg/m^3] (set to 1.2 kg/m^3) and V the volume of the space in [m^3]. The factor 3 in the equation is an arbitrary number that takes into account any additional capacitance in the space, e.g. furniture. The resistance from a space to a construction is set to 0.14 K/W which is an empirical value for an air layer of approximately 10 mm.

$$C_{s,i} = V \cdot \rho_{air} \cdot C_{air} \cdot 3 \tag{18}$$

Ventilation of a space is modelled as a loss of heat via a resistance to the weather profile, this is based on an air mass flow between the space and outside. The heat flux due to ventilation $\Phi_{q,vent}$ in [J/s] (i.e. Watt) in equation 19 is first expressed based on the air mass flow and subsequently also equated to the heat loss as modelled by a resistance R_{vent} in [K/W]. Solving the equation for the resistance yields equation 20 in which the flow of mass $\dot{m}$ in [kg/s] can be substituted by equation 21 to yield equation 22. Here T is the temperature in [K], R the resistance that models the heat loss due to ventilation with air of another temperature state [K/W] and ACH is the ventilation rate in number of air changes per hour.

$$\Phi_{q,vent} = \dot{m} \cdot C_{air} \cdot (T_2 - T_1) = \frac{T_2 - T_1}{R_{vent}} \tag{19}$$

$$R_{vent} = \frac{1}{\dot{m} \cdot C_{air,}} \tag{20}$$

$$\dot{m} = \rho_{air} \cdot V \cdot \frac{ACH}{3600} \tag{21}$$

$$R_{vent} = \left(C_{air} \cdot \rho_{air} \cdot V \cdot \frac{ACH}{3600}\right)^{-1} \tag{22}$$

Heating and cooling of spaces is modelled as a direct flux on the capacitance of the space's temperature state. A temperature control switches these fluxes on or off whenever the temperature in a space rises or falls below a set temperature point. This temperature control should be a gradual process, to prevent an overreaction when a set temperature point was exceeded by only a small amount. This is achieved with a P-switch, that expresses the flux as a tri-linear function in which the simulated heating power is dependent on the temperature of a state. Equation 23 and figure 14 illustrate the function of such a P-switch for heating, here T_{set} is the temperature set point, T_{var} is the length of the temperature range over which the heating (Q_{heat}) or cooling power is variable (set to 10 °C), and Q_{max} is the maximum amount of power.

$$Q_{heat} = \begin{cases} Q_{max} & \text{for} \quad T < (T_{set} - T_{var}) \\ Q_{max} \cdot \frac{T_{set} - T}{T_{set} - T_{var}} & \text{for} \quad (T_{set} - T_{var}) \leq T < T_{set} \\ 0 & \text{for} \quad T \geq T_{set} \end{cases} \tag{23}$$

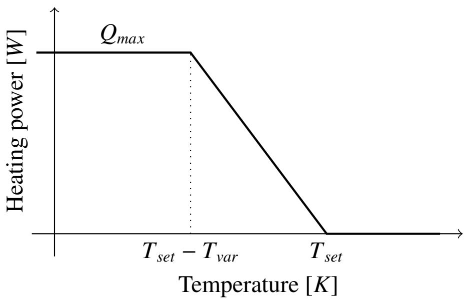


Figure 14: P-switch that controls heating in spaces, a similar function is used for cooling

Independent state objects resemble external influences to the model, e.g. weather and soil. Information regarding these states should be provided in the form of a time profile of temperatures or irradiations by the user of the toolbox. Time profiles can be arbitrary design values or real world measurements. Currently the toolbox can only use air temperature for simulations, data like solar irradiation is not considered.

Assembly and simulation. The assembly of the model starts by initialising temperature states of all spaces to the system. Accordingly the temperature states of building constructions are initialised to the system, this process also handles the association with neighbouring states. On initialisation, dependent and independent temperature are indexed with respect to their positions in state vectors $\mathbf{x}$ and $\mathbf{u}$. The state matrices $\mathbf{A}$ and $\mathbf{B}$ can be initialised once all temperature states have been added to the system. Once the RC-network is assembled into a system of ODE's in the form of equation 15 it is solved for every consecutive time step in the simulation. After each time step the values of the independent temperature states are updated. A C++ library that offers generic implementations of algorithms for numerical solving of ordinary differential equations is employed to solve the system, which is the odeint library [38] that is part of an overarching library: Boost [39].

4.4. Toolbox benchmark

Building spatial design optimisation has been carried out in [28] by means of a simulation of co-evolutionary processes to minimise the strain energy in the structural design. The toolbox presented here has successfully been benchmarked with one of the simulations that were performed in this paper, see figure 15. The used design grammar creates—for each space—four flat shell components for the walls of a space and one flat shell component at the top of a space. Each flat shell component in the structural design is assigned a thickness of 150 mm and material properties that resemble concrete, i.e. a Young's modulus of 30000 N/mm^2 and a Poission's ratio of 0.3. A live load case of 1.8 kN/m^2 in negative z-direction is applied to each horizontally aligned flat shell component. Additionally four wind load cases (in $+x, +y, -x, -y$ directions) are applied to each vertically aligned flat shell that does not have a space at both sides of the flat shell. Each wind load case consists of three different types, i.e. pressure (1.0 kN/m^2), suction (0.8 kN/m^2) and shear (0.4 kN/m^2), which are applied to a flat shell corresponding to the wind direction and the direction of the normal of the flat shell on the side where no space is present. Constraints are applied to each of the bottom corners of spaces that are located at the bottom of the building spatial design. Each structural component is then meshed into 10 by 10 elements, completing the structural design grammar. The optimisation procedure is done by first performing topology optimisation on the structural design, which results in an optimal density for each structural finite element. Element densities are clustered into eight clusters using the k-means algorithm and subsequently the elements in the four lowest clusters are deleted. The number of deleted elements is then used as measure for the performance of each space and each space is then sorted with respect to the number of deleted elements. Accordingly half of the spaces with the highest number of deleted elements is removed from the building spatial design and additionally any remaining spaces that have an equal amount of deleted elements as one of the removed spaces are also removed. Finally all remaining spaces are split and subsequently scaled in x- and y-dimensions by a factor of $\sqrt{2}$ to bring the design back to its original number of spaces and volume, although it should be noted that spaces and thus volume may be lost in the previous step. This procedure is performed iteratively until a stopping criterion has been reached, which is here set to the third iteration.

It should be noted that some mistakes were found in code used in [28]. Firstly in the distribution of live loading it is described that loads are a half at the edges and a quarter at the corners of structural components, however this is only the case when these loads are located somewhere along the bounding box of the complete building spatial design. Secondly, clustering of element densities is not performed after clusters are initiated. Accordingly, for topology optimisation it should be noted that element volume sensitivities with respect to changes in element density are not considered in the computation of the gradient and that the volume constraint is erroneously implemented as a constraint that keeps the average density over all elements constant. Finally the magnitude of the live loading is given as 1.8 kN/m^2, while it is actually simulated four times as high at 7.2 kN/m^2. These mistakes were implemented in the toolbox presented here to succesfully benchmark it to that used in [28]. To evaluate program efficiency both the code as used in [28] and the toolbox that is presented in this paper have been used to simulate the problem of figure 15 on an HP Z440 workstation (Intel Xeon E5-2690 v3 @3.5 GHz @6 cores, 16 GB RAM @1600 MHz), simulation times were around 22 hour for the code used in [28] and 33 minutes for the toolbox presented here.

5. Toolbox demonstration

This section presents some early work in the development of simulations of co-evolutionary design processes, of which those presented here are algorithms that remove and add spaces based on space performances, see figure 16. The presented work shows the promise of simulations of co-evolutionary design processes over a super-structured approach, but it also shows the challenges that should still be overcome. Only super-structure free optimisation is demonstrated here, application of the toolbox in super-structured optimisation can be found in [40, 41, 42], in which the supercube representation is used with a multi-disciplinary evolutionary optimisation algorithm to optimise for structural performance and building surface area.

Simulation of co-evolutionary design processes. The simulation of a co-evolutionary design process is here elaborated as a process of design modifications that are based on design performances, this with the goal to improve the performances of the design at hand, see figure 16. Design modification is the process of removing and adding spaces at locations where it would be appropriate with respect to a design's performance. Before modification all performances are stored in matrix $\mathbf{F}$ which is indexed by space i and discipline j. $\mathbf{F}$ is normalised into matrix $\mathbf{P}$ using equation 24, for which then each space i and each discipline j the normalised space performances are stored. For single disciplinary modification all spaces are sorted in a list in ascending order of normalised space performance. Multi-disciplinary modification would require to first evaluate the normalised space performances of each discipline per space and express this evaluation into one normalised space performance before such a list can be computed. The top half of the spaces in the ordered list of spaces is then removed from the design, and to ensure a symmetric design also any remaining spaces that have the same normalised space performance as the last removed space are removed. Note that this could lead to a loss of spaces, which is allowed for the demonstration. Accordingly all spaces are split in half along their longest horizontal dimensions, or if both are equally long then they are split in half along

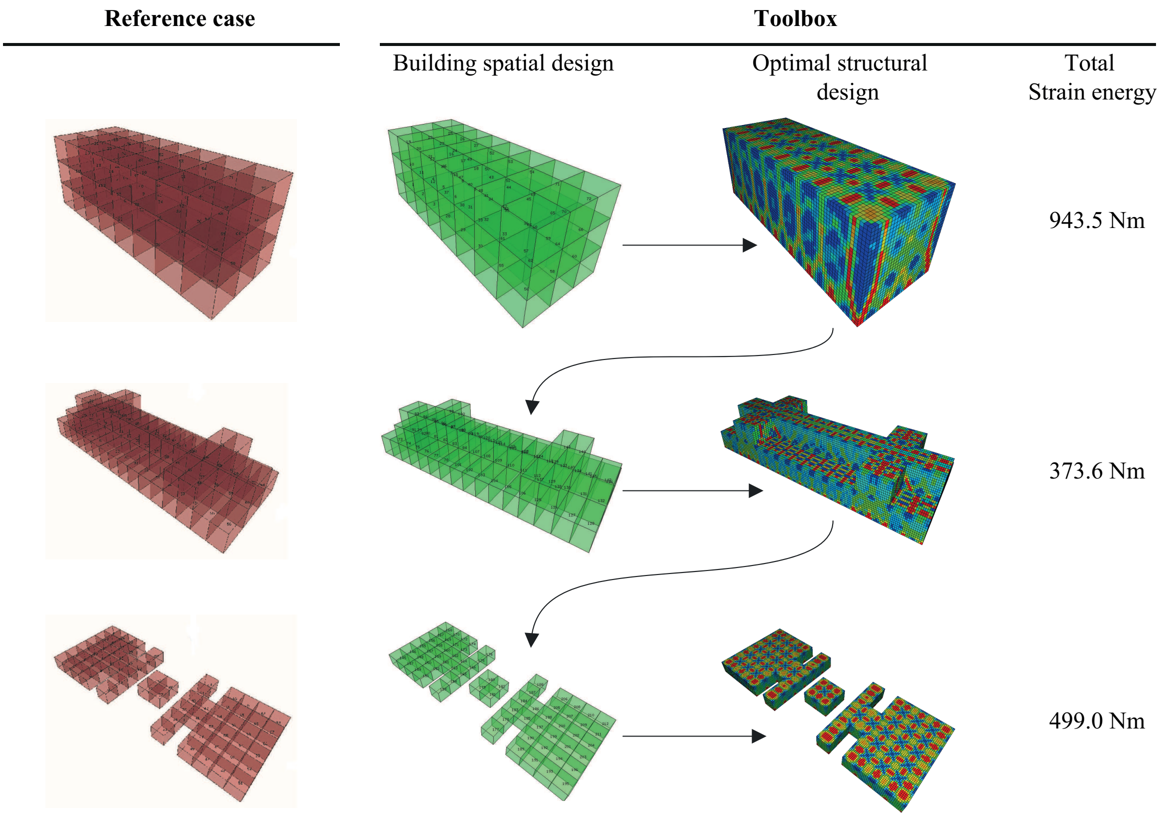


Figure 15: Succesful benchmark of the toolbox with a reference case that is presented in [28]

the x-direction. Finally the horizontal dimensions in the design are scaled with a factor of $\sqrt{2}$ to bring the design back to its original volume (assuming that no spaces were removed). One cycle of the simulation of co-evolutionary design processes is then completed, a stopping criterion terminates the process, which is in this demonstration met after two cycles have been completed.

$$P_{i,j} = \frac{F_{i,j} - min_j}{max_j - min_j} \tag{24}$$

where min_j and max_j are respectively the minimum and maximum terms in the j^{th} column.

It should be noted that the process described above is not an explicitly directed search for better performances. As such it can also not be defined as a global or local search. Also no hard constraints to guarantee valid designs are defined. However knowledge and experience can be used to define design modification such that better and valid designs can be found. Moreover, using different design modifications together can improve the chance to find better performing designs. Although this is an interesting topic, it is not elaborated here for brevity and it is not the purpose of the demonstration to address this topic. Moreover it should be noted that the demonstration entails only single disciplines. A multi-disciplinary search would introduce multiple new challenges to this paper, multiple disciplines have—for clarity and brevity—not been considered in the demonstration.

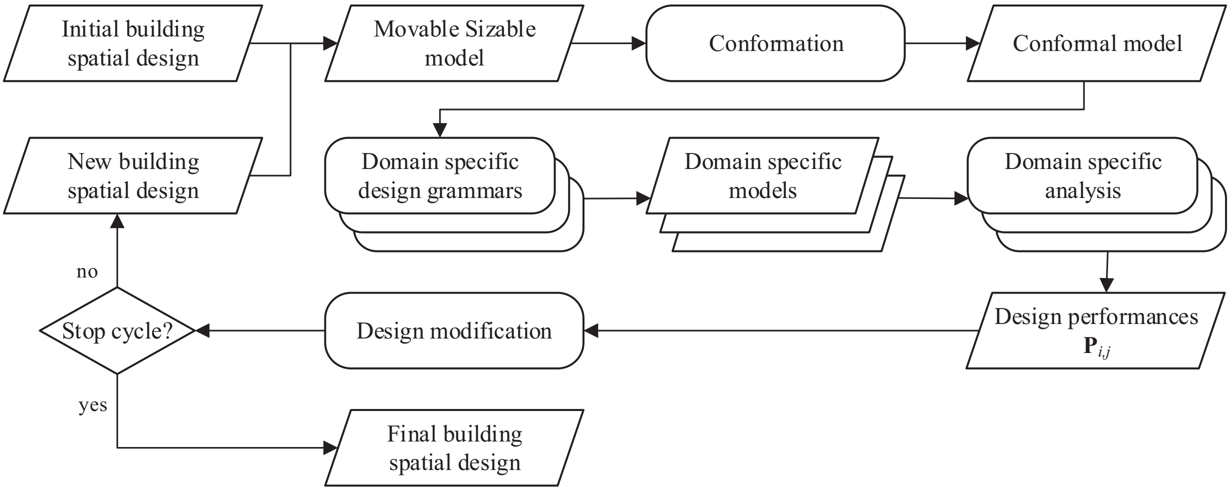


Figure 16: Process diagram of the simulation of co-evolutionary design processes that is used for the toolbox demonstration

5.1. Structural building design

The objective is to minimise the strain energy of a building spatial design (sometimes referred to as compliance), which is measured here by determining the total sum of strain energy that is acting in all structural design elements in the structural design that has been created for the spatial design. The structural performance per building space is measured by the sum of all strain energy acting in elements that are in or adjacent to a space (note that one element's strain energy might contribute to more than one space). For this simulation a design grammar is defined by assigning a flat shell component with a thickness of 150 mm, Youngs modulus of 30000 N/mm^2 and a Poisson's ratio of 0.3 to all rectangles in the conformal building spatial design that belong to a surface. A live load case is defined with loads of 5.0 kN/m^2 in $-z$ direction that are added to each horizontal flat shell component and wind loads are assigned to each surface in the conformal design that is not related to more than one space. Four load cases are defined for these wind loads, $+x, +y, -x$ and $-y$, a wind load itself is divided into three components, pressure 1.0 kN/m^2, suction 0.8 kN/m^2 and shear 0.4 kN/m^2, which are each added to a surface depending on its orientation and the wind direction. Finally the design grammar applies line constraints to each edge at the bottom of the building spatial design, the structural design is then meshed using 10 divisions in each dimension and it is solved using an LDLT solver [33].

Figure 17 shows the results after two cycles. After the first cycle there is a clear improvement of the strain energy in the structural design, however after the second cycle the strain energy is even higher than that of the initial design. The results are somewhat similar as the benchmark in figure 15, where a similar effect is observed. This shows that this approach is not a directed search, however it also shows that significant improvements could be found after just one iteration. These quick improvement steps suggest that a super-structure free approach may influence optimisation times significantly when this insight is used to limit a super-structured design search space to for example a maximum of two stories. From a structural point of view the results may be explained by the fact that flat buildings are more optimal since tall buildings lead to an accumulation of structural loads, whereas flat buildings transfer loads towards the foundation in a shorter path.

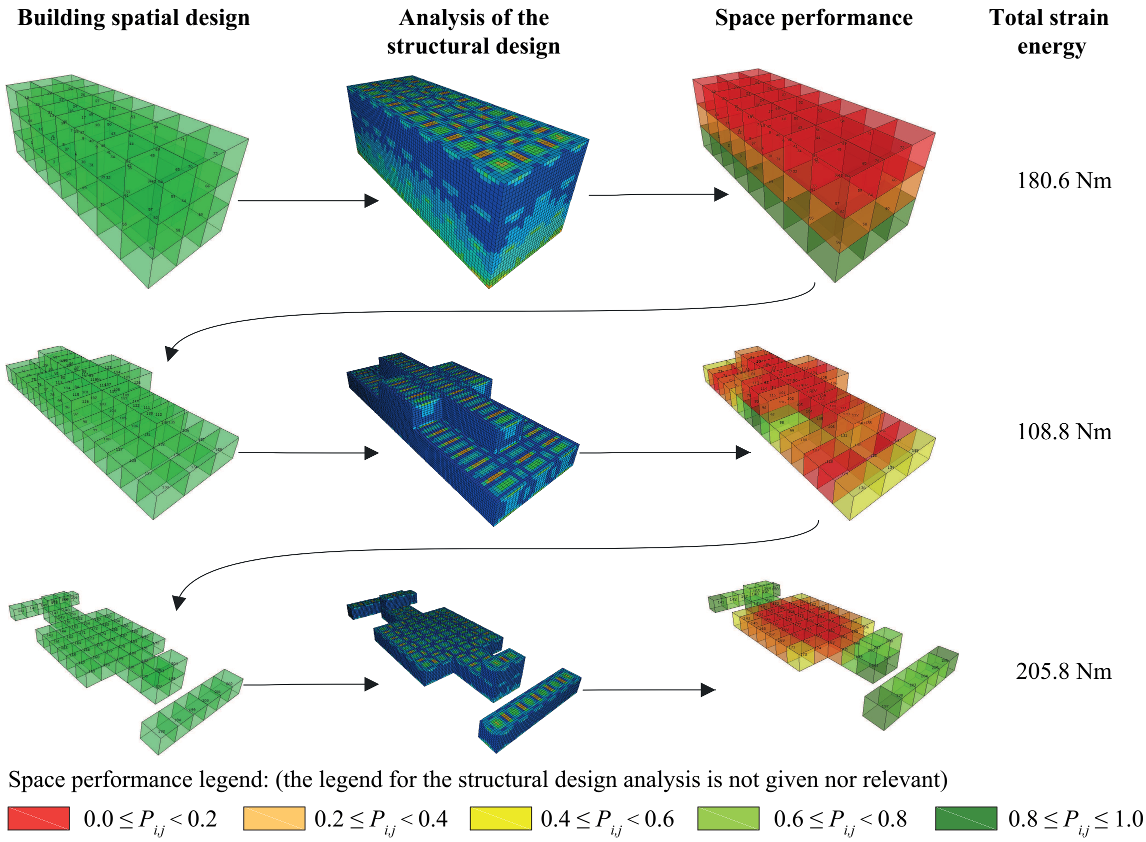


Figure 17: Simulation of building structural design process, normalised space performances are determined according equation 24

5.2. Thermal building design

The objective is to minimise the heating and cooling energy that is required to maintain the building between set temperatures. This is measured by simulating the heating and cooling energy demand in each space, the total energy demand is then computed as the sum of heating and cooling energies over the simulation time and over each space. To realise a thermal simulation, the building physics grammar assigns one building construction to each of the rectangles that belong to a surface in the building spatial design that consists of a 150 mm thick layer of concrete with a specific weight of 2400 kg/m^3, a specific heat capacity of 850 J/(K·kg) and a thermal conduction coefficient of 1.8 W/(K·m). Rectangles that belong to only one surface (i.e. one adjacent space or external wall) are assigned an additional layer to their construction, namely a layer of insulation of 150 mm thick with a specific weight of 60 kg/m^3, a specific heat capacity of 850 J/(K·kg) and a thermal conduction coefficient of 0.04 W/(K·m). The temperature set point for heating is set at 20 °C and the set point for cooling at 25 °C, the total available heating and cooling power in spaces is set to 100 W/m^3. The ventilation rate for each space in the design is one air change per hour. Real world data that was measured in De Bilt in The Netherlands by the Royal Netherlands Meteorological Institute (KNMI) [43] is used for the temperature profile of the weather and a constant temperature of 10 °C is used for the temperature profile of the ground. The building physics model is built up as follows, an object for a space is initialised for each space in the building spatial design. Accordingly objects for walls or floors are initialised for each rectangle that belongs to at least one surface, where the type is determined depending on the rectangle's orientation. Instances of walls and floors are linked to instances of spaces using the relational mappings of the conformal model. If a wall or floor is linked to only one space, then also a link to either the weather profile or the ground profile is added, depending on orientation and location. The simulation runs from the first of January 2014 until and including the last day of December 2014, i.e one year. Before the simulation period starts first a warm up period of six days is simulated by backwards traversing the first six days of both temperature profiles. The simulation time is discretised into four time steps per hour, the error controlled `runge-kutta-dopri-5` algorithm [38] is used to solve the system for each of those time steps using a value of 1e−6 for both the absolute and relative errors.

Figure 18 shows the results after two iterations. From these results it can be observed that the used design modification cannot find a better solution in the first two iterations, which suggests that a different design modification should be used. From a thermal point of view the spaces at a corner of a building spatial design will be suboptimal since these have the most surface through which heat is lost and looking at the results it can be observed that those spaces are in fact removed. However in the worst case when a corner space is removed this will introduce three new corner spaces, as such it can indeed be concluded that a different design modification should be used to find a thermally optimal building spatial design. A more suitable design modification would not only take into account the performance of spaces, but could for example also take into account their relative location in the building.

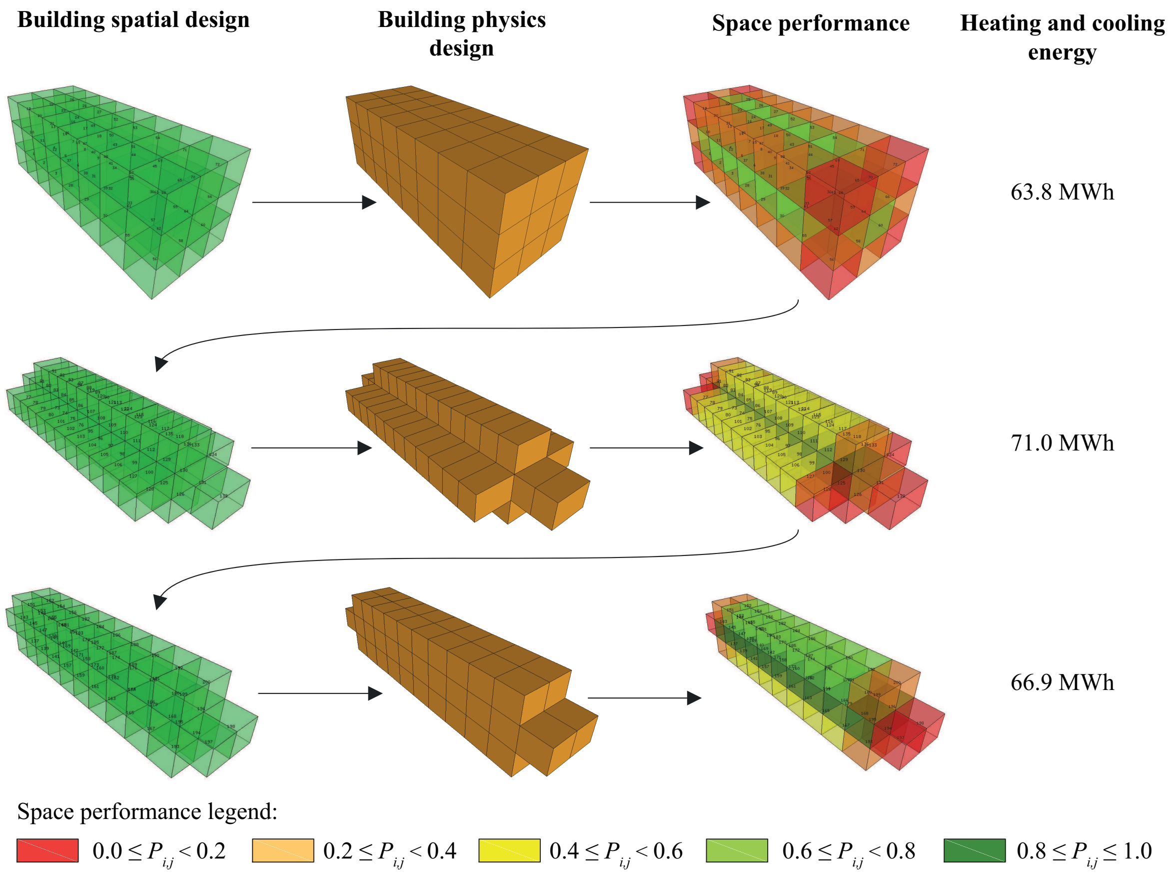


Figure 18: Simulation of building thermal design process, normalised space performances are determined according equation 24

6. Conclusions and outlook

This paper has elaborated on different optimisation approaches for building spatial design and has presented a toolbox to effectuate these approaches for further research. Conclusions and outlooks that have been presented in this paper are summarized below.

The difference between super-structured versus super-structure free approaches is a recurrent theme in specific fields of optimisation [2]. In this paper, for the super-structured approach, a supercube approach has been proposed, in which a fixed number of cells can be switched on and off to generate different building spatial designs, while constraints ensure practical designs, e.g. no overlap of spaces should occur. A super-structure free approach has been developed by a movable and sizeable representation, listing the building spaces with their position and dimensions, and allowing these spaces to be deleted, split, and resized, as such automatically following the constraints.

Algorithms have been derived to transform the supercube representation into the movable and sizeable representation and vice versa. These algorithms have been verified in [1] for successful operation when overlaps in spaces, non-connected spaces, truncation errors, alterations in space identification, and fragmented spaces occur.

A toolbox has been developed in which the presented spatial design representations can be evaluated for their structural and thermal behaviour. The toolbox enables users to develop and write their own optimisation procedures and design grammars. Also a benchmark has been presented in which the toolbox has successfully simulated a problem that is presented in other work.

The toolbox has been applied in [40, 41, 42], where evolutionary algorithms were employed to find optimal building spatial design configurations. Moreover an elementary implementation of a simulation of co-evolutionary design processes has been presented to demonstrate the use and versatility of the toolbox and also to show the promises and the challenges of this method.

In the near future, a multi-disciplinary design modification will be developed based on a simulation co-evolutionary design processes. Subsequently an optimisation approach will be developed where both representations are used alternately: The super-structured approach will allow a dedicated optimisation algorithm to find a global optimum [40, 42], whereas this solution in a super-structure free approach can be used by the developed design modification to explore more freely another (possibly local) optimum. As such the design space is cyclically both explored in-depth (via the super-structure) and globally (via the super-structure free representation).

Acknowledgements

The authors wish to express their gratitude towards the Dutch foundation for applied and engineering sciences NWO-TTW (Previously STW, project number: 13596), who have granted funding to the research presented here. The work of J.M. Davila Delgado is acknowledged here for his contributions in the research on simulations of co-evolutionary design processes for building spatial designs. The authors would also like to thank R.C.G.M. Loonen for his assistance with and his knowledge of building thermal simulations.

References

[1] S. Boonstra, K. van der Blom, H. Hofmeyer, R. Amor, M. T. M. Emmerich, Super-structure and super-structure free design search space representations for a building spatial design in multi-disciplinary building optimisation, in: EG-ICE 2016, Electronic proceedings of the 23rd EG-ICE workshop, Jagiellonian University ZPGK, 2016, pp. 1–10.

[2] P. Voll, M. Lampe, G. Wrobel, A. Bardow, Superstructure-free synthesis and optimization of distributed industrial energy supply systems, Energy 45 (1) (2012) 424–435. doi:10.1016/j.energy.2012.01.041.

[3] M. Wetter, Simulation-based building energy optimization, Ph.D. thesis, University of California, Berkeley (2004).

[4] D. Tuhus-Dubrow, M. Krarti, Genetic-algorithm based approach to optimize building envelope design for residential buildings, Building and Environment 45 (7) (2010) 1574–1581. doi:10.1016/j.buildenv.2010.01.005.

[5] Q. Q. Liang, Y. M. Xie, G. P. Steven, Optimal Topology Design of Bracing Systems for Multistory Steel Frames, Journal of Structural Engineering 126 (7) (2000) 823–829. doi:10.1061/(ASCE)0733-9445(2000)126:7(823).

[6] R. Baldock, K. Shea, Structural Topology Optimization of Braced Steel Frameworks Using Genetic Programming, in: Intelligent Computing in Engineering and Architecture: 13th EG-ICE Workshop 2006, Ascona, Switzerland, June 25-30, 2006, Revised Selected Papers, Springer-Verlag Berlin Heidelberg, 2006, pp. 54–61.

[7] B. Steiner, E. Mousavian, F. M. Saradj, M. Wimmer, P. Musialski, Integrated structural–architectural design for interactive planning, in: Computer Graphics Forum, Wiley Online Library, 2016.

[8] W. Wang, R. Zmeureanu, H. Rivard, Applying multi-objective genetic algorithms in green building design optimization, Building and Environment 40 (11) (2005) 1512–1525. doi:10.1016/j.buildenv.2004.11.017.

[9] J. A. Wright, H. A. Loosemore, R. Farmani, Optimization of building thermal design and control by multi-criterion genetic algorithm, Energy and Buildings 34 (9) (2002) 959–972. doi:10.1016/S0378-7788(02)00071-3.

[10] M. Hamdy, A.-T. Nguyen, J. L. Hensen, A performance comparison of multi-objective optimization algorithms for solving nearly-zero-energy-building design problems, Energy and Buildings 121 (2016) 57–71. doi:10.1016/j.enbuild.2016.03.035.

[11] M. Y. Rafiq, M. J. Rustell, Building Information Modeling Steered by Evolutionary Computing, Journal of Computing in Civil Engineering 28 (4) (2014) 05014003. doi:10.1061/(ASCE)CP.1943-5487.0000295.

[12] M. R. Asl, S. Zarrinmehr, M. Bergin, W. Yan, Bpopt: A framework for bim-based performance optimization, Energy and Buildings 108 (2015) 401–412.

[13] P. Geyer, Multidisciplinary grammars supporting design optimization of buildings, Research in Engineering Design 18 (4) (2008) 197–216.

[14] M. T. M. Emmerich, C. J. Hopfe, R. Marijt, J. L. M. Hensen, C. Struck, P. Stoelinga, Evaluating optimization methodologies for future integration in building performance tools, in: Proceedings of the 8th Int. Conf. on Adaptive Computing in Design and Manufacture (ACDM), 2008, pp. 1–7.

[15] C. J. Hopfe, M. T. M. Emmerich, R. Marijt, J. Hensen, Robust multi-criteria design optimisation in building design, in: Proceedings of Building Simulation and Optimization, Loughborough, UK, 2012, pp. 118–125.

[16] O. Sigmund, A 99 line topology optimization code written in Matlab, Structural and Multidisciplinary Optimization 21 (2) (2001) 120–127. doi:10.1007/s001580050176.

[17] A. S. L. Chan, The design of Michell optimum structures, Tech. rep., College of Aeronautics Cranfield (1960).

[18] R. Jackson, Optimization of chemical reactors with respect to flow configuration, Journal of Optimization Theory and Applications 2 (4) (1968) 240–259. doi:10.1007/BF00937370.

[19] A. Belegundu, S. Rajan, A shape optimization approach based on natural design variables and shape functions, Computer Methods in Applied Mechanics and Engineering 66 (1) (1988) 87–106. doi:10.1016/0045-7825(88)90061-8.

[20] Z. Sekulski, Least-weight topology and size optimization of high speed vehicle-passenger catamaran structure by genetic algorithm, Marine Structures 22 (4) (2009) 691–711. doi:10.1016/j.marstruc.2009.06.003.

[21] S. Bandaru, K. Deb, Temporal Innovization: Evolution of Design Principles Using Multi-objective Optimization, in: Evolutionary Multi-Criterion Optimization: 8th International Conference, EMO 2015, Guimarães, Portugal, March 29 –April 1, 2015. Proceedings, Part I, Springer International Publishing, 2015, pp. 79–93.

[22] C. A. Floudas, Nonlinear and mixed-integer optimization: fundamentals and applications, Oxford University Press on Demand, 1995.

[23] M. Emmerich, M. Grötzner, M. Schütz, Design of Graph-Based Evolutionary Algorithms: A Case Study for Chemical Process Networks, Evolutionary Computation 9 (3) (2001) 329–354. doi:10.1162/106365601750406028.

[24] S. Droste, D. Wiesmann, Metric Based Evolutionary Algorithms, in: Genetic Programming: European Conference, EuroGP 2000, Edinburgh, Scotland, UK, April 15-16, 2000. Proceedings, Springer Berlin Heidelberg, 2000, pp. 29–43.

[25] J. R. Koza, D. Andre, F. H. Bennett III, M. A. Keane, Use of automatically defined functions and architecture-altering operations in automated circuit synthesis with genetic programming, in: Proceedings of the 1st annual conference on genetic programming, MIT Press, 1996, pp. 132–140.

[26] W. Dolan, P. Cummings, M. Le Van, Algorithmic efficiency of simulated annealing for heat exchanger network design, Computers & Chemical Engineering 14 (10) (1990) 1039–1050. doi:10.1016/0098-1354(90)85001-Q.

[27] R. Kicinger, T. Arciszewski, K. De Jong, Evolutionary computation and structural design: A survey of the state-of-the-art, Computers & Structures 83 (23-24) (2005) 1943–1978. doi:10.1016/j.compstruc.2005.03.002.

[28] H. Hofmeyer, J. M. Davila Delgado, Coevolutionary and genetic algorithm based building spatial and structural design, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 29 (04) (2015) 351–370. doi:10.1017/S0890060415000384.

[29] J. R. R. A. Martins, A. B. Lambe, Multidisciplinary Design Optimization: A Survey of Architectures, AIAA Journal 51 (9)

(2013) 2049–2075. doi:10.2514/1.J051895.
[30] R. D. Cook, D. S. Malkus, M. E. Plesha, R. J. Witt, Concepts and applications of finite element analysis, 5th ed., New york, Wiley, 2002.
[31] J. S. Przemieniecki, Theory of matrix structural analysis, Courier Corporation, 1985.
[32] J. L. Batoz, M. B. Tahar, Evaluation of a new quadrilateral thin plate bending element, International Journal for Numerical Methods in Engineering 18 (11) (1982) 1655–1677.
[33] G. Guennebaud, B. Jacob, et al., Eigen v3: a c++ linear algebra library.
URL `http://eigen.tuxfamily.org`
[34] H. Hofmeyer, N. ten Heggeler, Structural topologies by iterative multi-structural grammars and separate volume fraction topology optimisation, in: Proc. of the 33rd CIB W78 Conference 2016, Oct. 31st Nov. 2nd 2016, Brisbane, Australia, 2016, pp. 1–10.
[35] H. Hofmeyer, M. Schevenels, S. Boonstra, The generation of hierarchic structures via robust 3d topology optimisation, Advanced Engineering Informaticsdoi:10.1016/j.aei.2017.02.002.
[36] J. van Schijndel, R. Kramer, Combining three main modeling methodologies for building physics, in: Proceedings of the 10th Nordic Symposium on Building Physics (NSB 2014). Lund, Sweden, 2014, pp. 558–565.
[37] R. Kramer, J. van Schijndel, H. Schellen, Inverse modeling of simplified hygrothermal building models to predict and characterize indoor climates, Building and Environment 68 (2013) 87–99.
[38] K. Ahnert, M. Mulansky, Odeint-solving ordinary differential equations in c++, arXiv:1110.3397. doi:10.1063/1.3637934.
[39] B. Dawes, D. Abrahams, R. Rivera, et al., Boost c++ libraries.
URL `http://www.boost.org`
[40] K. van der Blom, S. Boonstra, H. Hofmeyer, M. T. M. Emmerich, Multicriteria building spatial design with mixed integer evolutionary algorithms, in: J. Handl, et al. (Eds.), Parallel Problem Solving from Nature – PPSN XIV: 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings, Springer International Publishing, Cham, 2016, pp. 453–462.
[41] K. van der Blom, S. Boonstra, H. Hofmeyer, M. T. M. Emmerich, A super-structure based optimisation approach for building spatial designs, in: V. Plevris, M. Papadrakakis, V. Papadopoulos, G. Stefanou (Eds.), Congress proceedings of the VII European Congress on Computational Methods in Applied Sciences and Engineering ECCOMAS, 2016, pp. 1–14.
[42] K. van der Blom, S. Boonstra, H. Hofmeyer, T. Bäck, M. T. M. Emmerich, Configuring advanced evolutionary algorithms for multicriteria building spatial design optimisation, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017.
[43] Koninklijk Nederlands Meteorologish Instituut, Measured weather data in The Netherlands.
URL `http://www.knmi.nl/nederland-nu/klimatologie/daggegevens`