Super-Structure and Super-Structure Free Design Search Space Representations for a Building Spatial Design in Multi-Disciplinary Building Optimisation

S. Boonstra[†], K. van der Blom[‡], H. Hofmeyer[†], R. Amor[*], M.T.M. Emmerich[‡]
[†]Eindhoven University of Technology, The Netherlands
[‡]LIACS, Leiden University, The Netherlands
[*]University of Auckland, New Zealand
[†]h.hofmeyer@tue.nl

Abstract. In multi-disciplinary building optimisation, solutions depend on the representation of the design search space, the latter being a collection of all solutions. This paper presents two design search space representations and discusses their advantages and disadvantages: The first, a super-structure approach, requires all possible solutions to be prescribed in a so-called super-structure. The second approach, super-structure free, uses dynamic data structures that offer freedom in the range of possible solutions. It is concluded that both approaches may supplement each other, if applied in a combination of optimisation methods. A method for this combination of optimisation methods is proposed. The method includes the transformation of one representation into the other and vice versa. Finally, therefore in this paper these transformations are proposed, implemented, and verified as well.

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 design problems are too large for a single design tool, and engineers are 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 developing such a method and focuses on the question of how to represent design search spaces such that optimisation methods find efficient solutions.

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 (Voll et al. 2012). 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 (Voll et al. 2012), 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.

2. Related Work

In literature, research on building optimisation can be found taking into account objectives like energy consumption, as is carried out by Wetter (2004) and Tuhus-Dubrow & Krarti (2010); structural stiffness by Liang, Xie & Steven (2000) and Baldock & Shea (2006); construction costs by Wang, Zmeureanu & Rivard (2005); and thermal comfort by Wright, Loosemore & Farmani (2002). Also, optimisation is thoughtfully combined with Building Information Modelling, Rafiq & Rustell (2014). Different energy performance criteria are combined in Emmerich et al. (2008) and Hopfe et al. (2012). 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 Sigmund (2001), or the analytical derivation of an optimal truss layout by Chan (1960). 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 also 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 (1968) described the structure of flow configurations of chemical reactors with a super-structure, although without explicit mentioning the term. Various recent works (Belegundu & Rajan 1988; Sekulski, 2009; Bandaru & Deb 2015) 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. Floudas 1995).

Super-structure free optimisation has been suggested to overcome the limitations of super-structures for designing chemical process configurations. Emmerich, Grötzner & Schütz (2001) 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. (2012) suggest a more general

framework that uses generic replacement rules in evolutionary algorithms. A similar strategy is followed by Droste & Wiesmann (2000), who exemplified it for the optimisation of decision diagrams. Other examples of super-structure free design spaces include the work of Koza et al. (1996) and Baldock and Shea (2006). There are only a few optimisation methods that can handle super-structure free representations, namely simulated annealing, evolutionary algorithms, and heuristic local search. Simulated annealing has been used in the design of processes, e.g. in Dolan, Cummings & Le Van (1990). In the field of structural design, Kicinger, Arciszewski & De Jong (2005) describe a super-structure free approach in the optimisation of structural topologies. Moreover, Hofmeyer & Davila Delgado (2015) use simulations of a co-evolutionary design process (these simulations can also be interpreted as asymmetric subspace optimisation, Martins & Lambe 2013) to find a building spatial design for which the structural grammar applied structural design shows minimal compliance.

3. Super-Structure Based Representation

3.1 Design Search Space

A so-called "super cube" (SC) is suggested to describe a building spatial design by means of a super-structure design search space representation. A super cube consisting of cells is described by four vectors: $B\{\overrightarrow{w_i}, \overrightarrow{d_j}, \overrightarrow{h_k}, \vec{b}^{\,l}_{i,j,k}\}$. Equation (1) shows the variables used. Here $b^l_{i,j,k}$ describes the existence of the cell with indices i, j and k in space l, where a value "1" means the cell is active and describes a part of space l while "0" means the cell is inactive. Following this, i is the x-, j is the y- and k is the z-index of a cell, while l is the space index. Finally, w_i, d_j, h_k describe the continuous dimensioning of the super cube's cells. The entire super cube is used to perform design modifications, therefore the complete design space is described by the vectors $\overrightarrow{w_i}, \overrightarrow{d_j}, \overrightarrow{h_k}$ and $\vec{b}^{\,l}_{i,j,k}$. Figure 1 shows the super cube 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.

$$\begin{aligned} i &\in \{1,2,\dots,N_w\} \qquad w_i \in \mathbb{R} \geq 0 \\ j &\in \{1,2,\dots,N_d\} \qquad d_j \in \mathbb{R} \geq 0 \\ k &\in \{1,2,\dots,N_h\} \qquad h_k \in \mathbb{R} \geq 0 \\ l &\in \{1,2,\dots,N_{spaces}\} \qquad b^l_{i,j,k} = \begin{cases} 1 & \text{if cell } (i,j,k) \text{ belongs to space } l \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

3.2 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 super cube's grid. Constraints are introduced to the design 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 optimisation algorithms employed. The super cube representation is suitable for such algebraic expression of constraints, and 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 $\vec{b}^{l}_{i,j,k}$.

$\vec{w}\{w_1, w_2, w_3, w_4\}$,

$\vec{d}\{d_1, d_2\}$, $\vec{h}\{h_1\}$

$\overrightarrow{b^{l}_{i,j,k}}\{b^{1}_{i,j,k}, b^{2}_{i,j,k}, b^{3}_{i,j,k}, b^{4}_{i,j,k}\}$

$b^{1}_{i,j,k}\{1,0,0,0,0,0,0,0\}$

$b^{2}_{i,j,k}\{0,0,1,0,1,0,0,0\}$

$b^{3}_{i,j,k}\{0,1,0,0,0,0,0,0\}$

$b^{4}_{i,j,k}\{0,0,0,1,0,1,0,0\}$

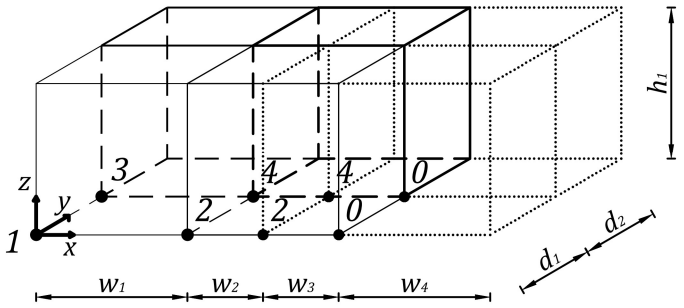

Figure 1 "Super cube" representation of a building spatial design, space 2 and 4 are described by two cells each, the two right cells are not used to describe a room

Condition 1: Non Overlap. Overlaps of building spaces are not allowed for they are not practical and might cause erroneous results in subsequent design analysis. And since every space is represented by a separate bit-mask (l) of all cells of the super cube, which does not automatically prevent non-overlap, this needs to be checked. 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.

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

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 super cube will be extended with a single layer of cells all around, and these new cells will be set to be not related to a space ("0"), the latter described by equation (3):

$$\begin{gathered}\forall_l : \forall_{i,j,k} \in \{0, \dots, N_w + 1\} \times \{0, \dots, N_d + 1\} \times \{0, \dots, N_h + 1\}: \\ i = 0 \vee j = 0 \vee k = 0 \vee i = N_w + 1 \vee j = N_d + 1 \vee k = N_h + 1 \Rightarrow b^{l}_{i,j,k} = 0\end{gathered} \tag{3}$$

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, $b^{l}_{i,j,k}$ 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.

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 ensured that building spaces have a fully occupied cuboid shape.

$$
\begin{gathered}
\forall_{\boldsymbol{l}}: \\
\forall_{i_1,j_1,i_2,j_2}: \left(\left(\sum_{k=1}^{N_h} k\left(1-b^l_{i_1,j_1,k-1}\right)b^l_{i_1,j_1,k}\right)-\left(\sum_{k=1}^{N_h} k\left(1-b^l_{i_2,j_2,k-1}\right)b^l_{i_2,j_2,k}\right)\right) \\
\left(\sum_{k=1}^{N_h} b^l_{i_1,j_1,k}\right)\left(\sum_{k=1}^{N_h} b^l_{i_2,j_2,k}\right)=0 \\
\forall_{i_1,j_1,i_2,j_2}: \left(\left(\sum_{k=1}^{N_h} k b^l_{i_1,j_1,k}\left(1-b^l_{i_1,j_1,k+1}\right)\right)\right. \\
\left.-\left(\sum_{k=1}^{N_h} k b^l_{i_2,j_2,k}\left(1-b^l_{i_2,j_2,k+1}\right)\right)\right)\left(\sum_{k=1}^{N_h} b^l_{i_1,j_1,k}\right)\left(\sum_{k=1}^{N_h} b^l_{i_2,j_2,k}\right)=0
\end{gathered}
\tag{4}
$$

$$
\begin{gathered}
\forall_l: \\
\forall_{i,j}: \sum_{k=0}^{N_h}\left(1-b^l_{i,j,k}\right)b^l_{i,j,k+1} \le 1 \qquad \forall_{i,k}: \sum_{j=0}^{N_d}\left(1-b^l_{i,j,k}\right)b^l_{i,j+1,k} \le 1 \\
\forall_{j,k}: \sum_{i=0}^{N_w}\left(1-b^l_{i,j,k}\right)b^l_{i+1,j,k} \le 1
\end{gathered}
\tag{5}
$$

4. Super-Structure Free Based Representation

4.1 Design Search Space

A “movable and sizable” (MS) representation for spaces is suggested for the super-structure free design space representation. For this, a building is described with: $B\{\vec{S}\}$, in which a single $\vec{S}$ 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 sizable" representation.

$$
\begin{gathered}
\vec{S} = \left\{S_1, S_2, \dots, S_{N_{spaces}}\right\} \\
S_i = \{C, D\}; \quad C = \{x, y, z\}; \quad D = \{w, d, h\}
\end{gathered}
\tag{6}
$$

$S_1\{\{x_1, y_1, z_1\}, \{w_1, d_1, h_1\}\}$

$S_2\{\{x_2, y_2, z_2\}, \{w_2, d_2, h_2\}\}$

$S_3\{\{x_3, y_3, z_3\}, \{w_3, d_3, h_3\}\}$

$S_4\{\{x_4, y_4, z_4\}, \{w_4, d_4, h_4\}\}$

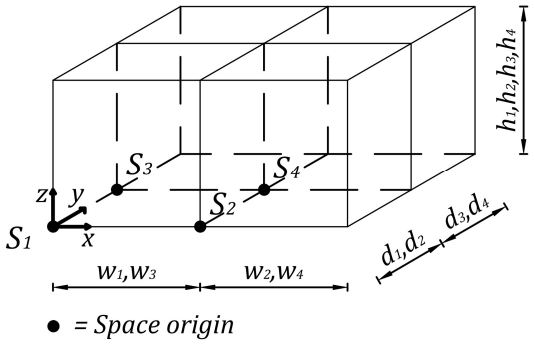

Figure 2 "Movable and sizable" representation of the building spatial design (first shown in figure 1)

4.2 Constraints and Design Modification

In the super-structure free approach, constraints are implicitly enforced by using design modification that naturally follow the constraints. The MS representation is intuitive for building engineers and allows for straight forward design modification. Here, this is carried out via removal, scaling and division of spaces. As an example, a modification of the spatial building 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 Hofmeyer & Davila Delgado (2015) for real-world optimisation scenarios.

Deletion: $$\vec{S}\{S_1, S_2, S_3, S_4\} \rightarrow \vec{S}\{S_1, S_2, S_4\} \tag{7}$$

Scaling: $$\begin{pmatrix} S_1\{\{x, y\}, \{w, d\}\}, S_2\{\{x, y\}, \{w, d\}\}, \\ S_4\{\{x, y\}, \{w, d\}\} \end{pmatrix} \cdot \sqrt{\frac{V_0}{V}} \tag{8}$$

Division: $$S_1\{\{x_1, y_1, z_1\}, \{w_1, d_1, h_1\}\} \rightarrow \begin{cases} S_5\left\{\{x_1, y_1, z_1\}, \left\{\frac{1}{2}w_1, d_1, h_1\right\}\right\}, \\ S_6\left\{\left\{x_1 + \frac{1}{2}w_1, 0{,}0\right\}, \left\{\frac{1}{2}w_1, d_1, h_1\right\}\right\} \end{cases} \tag{9}$$

New design: $$S\{S_2, S_4, S_5, S_6\} \tag{10}$$

5. 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 section 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 space representation draws a limit on what 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.

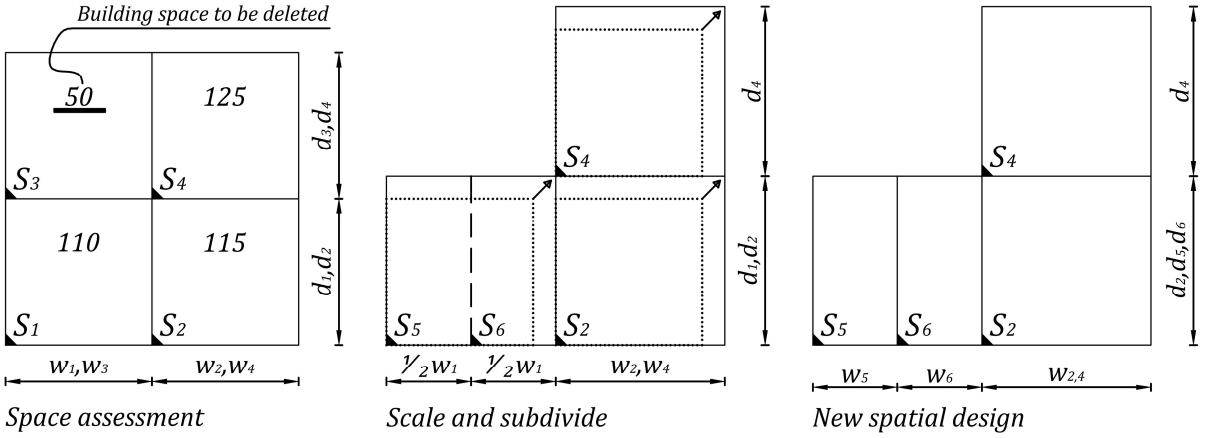

Figure 3 Super-structure free modification, numbers in spaces in the most left figure represent (structural or building physics) performance.

The super-structure free based approach to building optimisation can be developed even when only expertise in 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 the size of the design space does not influence the search for an improved design. 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 the finding of 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-structure approach, new design insights are more difficult to find when using heuristics, because less 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-structure 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.

6. Combination of Super-Structure and Super-Structure Free Approaches

The combination of the approaches above is proposed by alternately employ each approach during the optimisation process for the same problem. This alternation requires mutual transformation between the two representations, this section will presents two suitable algorithms.

6.1 "Super Cube" (SC) to "Movable and Sizable" (MS)

To transform a building spatial design's "Super Cube" representation into the "Movable and Sizable" 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).

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

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

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

6.2 "Movable and Sizable" (MS) to "Super Cube" (SC)

A transformation from "Movable and Sizable" to "Super Cube" first requires three steps to compute the super cube dimensions $\vec{w}$, $\vec{d}$, $\vec{h}$. First, for each space, the minimum and maximum coordinate values should be found, i.e. for each space $\{x, x+w\}, \{y, y+d\}, \{z, z+h\}$. Second, all these values are brought together, duplicate values are removed, and then values are grouped in three lists (each for either x, y or z values), and sorted in ascending order. Finally, vectors $\vec{w}$, $\vec{d}$, $\vec{h}$ are computed from these lists. For example, w is computed as $w_i = x_{i+1} - x_i$, for every $i \in [1, \ldots, n-1]$.

Regarding vector $\vec{b}^l_{i,j,k}$, for each space and for each cell 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 $x_{space} \leq x_{cell} < x_{space} + w_{space}$.

6.3 Verification

The preceding transformation algorithms have been implemented and tested, including the modification of an existing visualisation tool to provide graphical support. The verification concerned: overlaps in spaces (figure 4), non-connected spaces, truncation errors, alterations in space identification, and fragmented spaces (figure 5). From the verifications it could be concluded that overlaps in spaces and non-connected spaces are preserved throughout transformations. Truncation errors can in special cases lead to small grid sizes in the "super cube" representation, which can be solved by deleting these grid size values ($\vec{w}$, $\vec{d}$, $\vec{h}$ and related $\vec{b}^l_{i,j,k}$). Further it could be concluded that building space identifications may get lost, however if desired this is easily solved by allocating a separate variable for space-identification. Finally, fragmented spaces in SC are not preserved. This is because in MS such

a building space is naturally not possible, see figure 5. As a conclusion, it is expected that an alternation between search space representations is possible, provided that overlaps and fragmented spaces in SC are not allowed (i.e. the constraints in section 3.2 must be followed).

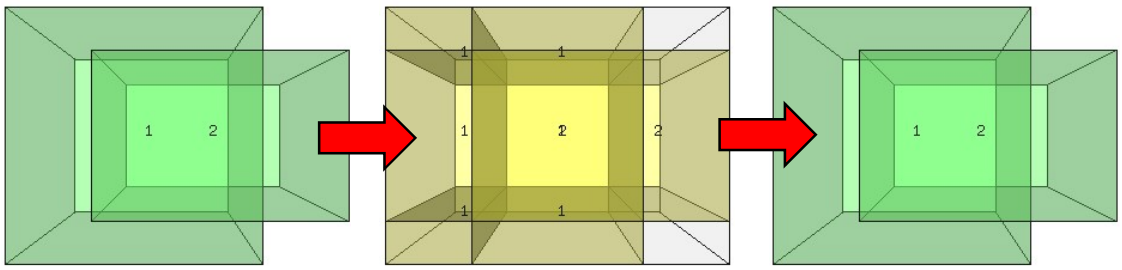

Figure 4 Building space overlaps subject to transformations are preserved. From left to right: Initial MS to SC, then SC to MS-representation.

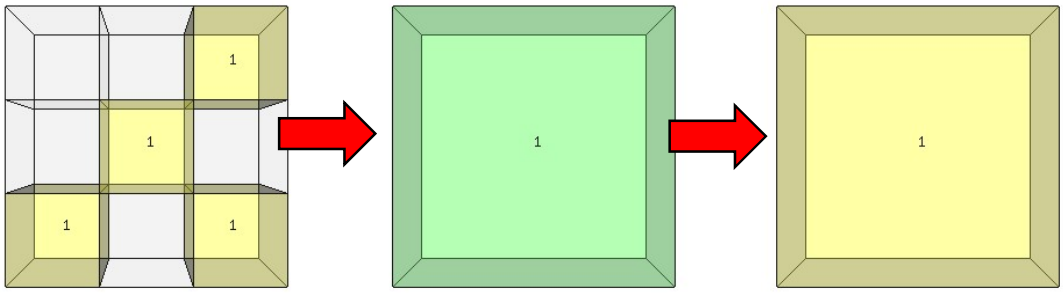

Figure 5 Fragmented space gets lost during transformation. From left to right: Initial SC to MS, then MS to SC representation.

7. Conclusions and Outlook

The difference between super-structure versus super-structure free approaches is a recurrent theme in specific fields of optimisation (Voll et al. 2012). In this paper, for the super-structure approach, a "super-cube" approach has been proposed, in which a fixed number of cells can be switched on and off to generate different building spatial designs, and constraints ensure practical designs, e.g. no overlap of spaces should occur. A super-structure free approach has been developed by a "movable and sizable" 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 "super-cube" representation into the "movable and sizable" representation and vice versa, and these algorithms have been verified for successful operation for overlaps in spaces, non-connected spaces, truncation errors, alterations in space identification, and fragmented spaces.

In the near future, an optimisation approach will be developed where both representations are used alternately: The super-structure approach will allow a dedicated optimisation algorithm to find a global optimum (Van der Blom et al. 2016), whereas this solution in a super-structure free approach can be used 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).

References

Baldock, R. & Shea, K., 2006. Structural Topology Optimization of Braced Steel Frameworks Using Genetic Programming. *13th EG-ICE Workshop*, pp.54–61.

Bandaru, S. & Deb, K., 2015. Temporal Innovization: Evolution of Design Principles Using Multi-objective Optimization. In *8th International Conference of Evolutionary Multi-Criterion Optimization*. pp. 79–93.

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

Chan, A.S.L., 1960. The Design of Michell Optimum Structures. *The college of aeronautics* (Vol. 142), Cranfield.

Dolan, W.B., Cummings, P.T. & Le Van, M.D., 1990. Algorithmic Efficiency of Simulated Annealing for Heat Exchanger Network Design. *Computers Chem. Eng*, 14, pp.1039–1050.

Droste, S. & Wiesmann, D., 2000. Metric Based Evolutionary Algorithms, Proceedings of EuroGP'2000, Berlin: Springer-Verlag, pp. 29-43.

Emmerich, M., Grötzner, M. & Schütz, M., 2001. Design of Graph-Based Evolutionary Algorithms: A Case Study for Chemical Process Networks. *Evolutionary Computation*, 9(3), pp.329–354.

Emmerich, M. T. M., Hopfe, C. J., Marijt, R., Hensen, J. L. M., Struck, C., & Stoelinga, P. (2008). 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) (Vol. 29, p. 7).

Floudas, C.A., 1995. *Nonlinear and mixed-integer optimization: fundamentals and applications*, Oxford University Press.

Hofmeyer, H. & Davila Delgado, J.M., 2015. Co-evolutionary and Genetic Algorithm Based Building Spatial and Structural Design. *AIEDAM 29*, pp.351-370.

Hopfe, C. J., Emmerich, M. T., Marijt, R., & Hensen, J. (2012). Robust multi-criteria design optimisation in building design. Proceedings of Building Simulation and Optimization, Loughborough, UK, pp. 118-125.

Jackson, R., 1968. Optimization of chemical reactors with respect to flow configuration. *Journal of Optimization Theory and Applications*, 2(4), pp.240–259.

Kicinger, R., Arciszewski, T. & De Jong, K., 2005. Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers and Structures*, 83(23-24), pp.1943–1978.

Koza, J.R., Andre, D., Bennett, F.H. & Keane, M.A., 1996. Use of Automatically Defined Functions and Architecture-Altering Operations in Automated Circuit Synthesis Using Genetic Programming. In *Genetic Programming 1996: Proceedings of the First Annual Conference*. pp. 132–149.

Liang, Q.Q.; Xie, Y.M.; Steven, G.P., 2000. Optimal topology design of bracing systems for multistory steel frames. *Structural engineering*, 126(7), pp.823–829.

Martins, J.R.R.A. & Lambe, A.B., 2013. Multidisciplinary Design Optimization: A Survey of Architectures. *AIAA Journal*, 51(9), pp.2049–2075.

Rafiq, M.Y. & Rustell, M.J., 2014. Building Information Modeling Steered by Evolutionary Computing. Journal of Computing in Civil Engineering, 28(4), pp.05014003-1-11.

Sekulski, Z., 2009. Least-weight topology and size optimization of high speed vehicle-passenger catamaran structure by genetic algorithm. *Marine Structures*, 22(4), pp.691–711.

Sigmund, O., 2001. A 99 line topology optimization code written in matlab. *Structural and Multidisciplinary Optimization*, 21, pp.120–127.

Tuhus-Dubrow, D. & Krarti, M., 2010. Genetic-algorithm based approach to optimize building envelope design for residential buildings. *Building and Environment*, 45(7), pp.1574–1581.

Van der Blom, K., Boonstra, S., Hofmeyer, H., Emmerich, M.T.M., 2016. A Super-structure Based Optimisation Approach for Building Spatial Topologies. Accepted for ECCOMAS 2016, Greece, June 5-10, 2016.

Voll, P., Lampe, M., Wrobel, G. & Bardow, A., 2012. Superstructure-free synthesis and optimization of distributed industrial energy supply systems. *Energy* 45(1), pp. 424-435.

Wetter, M. 2004. Simulation-Based Building Energy Optimization. *Ph.D. thesis*. University of California, Berkeley.

Wang, W., Zmeureanu, R. & Rivard, H., 2005. Applying multi-objective genetic algorithms in green building design optimization. *Building and Environment*, 40(11), pp.1512–1525.

Wright, J.A., Loosemore, H.A. & Farmani, R., 2002. Optimization of building thermal design and control by multi-criterion genetic algorithm. *Energy and Buildings*, 34(9), pp.959–972.