ECCOMAS Congress 2016
VII European Congress on Computational Methods in Applied Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou, V. Plevris (eds.)
Crete Island, Greece, 5–10 June 2016

A SUPER-STRUCTURE BASED OPTIMISATION APPROACH FOR BUILDING SPATIAL DESIGNS

Koen van der Blom[1], **Sjonnie Boonstra**[2], **Hèrm Hofmeyer**[2] **and Michael T. M. Emmerich**[1]

[1] Leiden Institute of Advanced Computer Science,
Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
e-mail: {k.van.der.blom, m.t.m.emmerich}@liacs.leidenuniv.nl

[2] Eindhoven University of Technology, Department of the Built Environment,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: h.hofmeyer@tue.nl

Keywords: Evolutionary Algorithms, Super-structure, Mixed-Integer Optimisation, Building Spatial Design, Building Structural Design.

Abstract. *Building design can be supported effectively by computer-aided design exploration. This paper investigates optimisation based on a mixed-integer super-structure representation of the search space of building spatial designs. It can take into account parametric as well as topological variations. In the suggested super-structure – the so-called supercube representation – discrete and continuous variables determine the existence, respectively, dimensioning of spaces of the building spatial design. Constraints are formulated as closed form equations and can be used to numerically assess the feasibility of designs. A population-based constraint-handling evolutionary strategy is developed. In the constraint handling repair and penalty methods are combined in a domain specific way. The method is tested on different search space sizes and first promising results are reported.*

1 INTRODUCTION

The use of automated search for finding optimal buildings with respect to various criteria, such as, e.g. structural strength, and stiffness; and energy performance will be an important research topic in the 21st century. Many physical phenomena can be modelled and studied by computer simulations, e.g., mechanical stress, heat transport, and light radiation. However, via these means, search for improvement is still widely limited to a trial and error approach. New research is required for enabling the search in larger design spaces.

This paper is about the foundations of such research. Namely, it will address the question how to represent a search space for optimisation and how search methods can navigate this design space in search for better designs. For this, global optimisation methods for large search spaces, such as evolutionary algorithms, will be regarded here, for across various engineering domains they have shown their potential to discover new, often unexpected design solutions. A method that can take into account structural as well as continuous variables will be proposed. Moreover, it is discussed how to handle constraints on discrete and continuous variables. Both, repair and penalty methods are used and it is investigated how often constraints are violated in stochastic search. The results on building spatial designs of small and moderate size are promising and provide interesting indications on how to plan future research.

The paper is structured as follows: In Section 2 the problem of building design is introduced as well as a review of related work. The basic outline of the optimisation algorithm is provided in Section 3. In Section 4 the search space, objective functions and constraints are discussed in more detail. Then in Section 5 the coupling of optimisation and constraint handling is discussed and the setup of experiments used for validating the approach. An in-depth discussion of results is presented in Section 6. In Section 7 a summary of the main results is provided and directions for future work are indicated.

2 BUILDING DESIGN

Building design is traditionally performed by architects and engineers who create solutions for discipline specific design problems. These solutions are nowadays usually assessed by and modified in accordance with design analysis tools. Such tools are for example finite element methods (FEM), to simulate structural performance or computational fluid dynamics (CFD) to simulate simulate heat, air, and moisture problems. The division created by the different disciplines within the field also calls for tools that allow engineers within different disciplines to cooperate. Examples are computer aided design (CAD) that is used to create and share designs. However, currently, building information modelling (BIM) is on the rise. BIM is a method that uses data management in order to dynamically share information with other disciplines. This allows engineers to – among other things – take other disciplines into account in the early design phase. The early design phase is important for optimal building designs, because decisions in the conceptual design stage often affect performances across all disciplines. A single disciplinary design may therefore lead to a sub-optimal multi-disciplinary design. Optimisation in the built environment is mostly performed by parametrising building components, e.g. installation type, construction type, material type, dimensions or shapes. Software tools for building optimisation purposes have been developed: Palonen *et al.* [1] give an overview and present their own tool. In the tool of Palonen, design variables of a building design can be selected for optimisation, an optimisation strategy can be selected thereafter. Although these tools can improve and alter a designs appearance greatly, they cannot lead to new designs (e.g. a new window cannot appear). Very recently, advances in early design optimisation are made: Hofmeyer

& Davila Delgado [2] for example use a design process inspired optimisation approach to optimise a building spatial design for the structural performance of its related structural design. Attia *et al.* [3] present software that gives information about building physics performances during early design stages. Hopfe *et al.* [4] use statistical sensitivity analysis to predict the impact of design variables on the optimality of a building design. The analysis by Hopfe *et al.* is interesting for early design optimisation as the impact of each design variable in distinct design stages can be investigated.

Here it is tried to perform building optimisation for early stage building spatial design using a new super-structure design space representation. A building spatial design is here merely a layout of building spaces that can be rearranged and resized for optimal performance. Optimisation methods are investigated for two different objective functions on the design space: Structural performance, measured by minimal compliance, and building physics, measured by minimal outside surface area, are selected for the optimisation objectives. These disciplines are selected because they are known to be dependent on the building spatial design. It is intended to use a RC-network to analyse building heat-balance in the future, but for the sake of developing the optimisation strategy presented here only the minimal surface area is taken as objective function. Details on the setup of the analysis are deferred to Section 5.

3 OPTIMISATION

Evolutionary algorithms subsume different algorithms that mimic natural evolution, in order to find improved or optimised technological designs [5]. Population-based evolutionary algorithms generally work according to a basic loop structure – the so-called generational loop. It starts after an initialisation phase where an initial population of individuals (solution candidates) is generated and evaluated. In the evolutionary loop first a ranking among individuals according to their fitness (evaluation results) is established. The following step is to select the parents to generate an offspring population. In this step the ranking of the population might be taken into account, although in Evolution Strategies – an important EA variant – parent individuals are chosen randomly. From the selected parent individuals λ offspring individuals are created. Recombination is applied to allow parts of the genome from multiple parents to together from a new genome. In order to introduce new, possibly not previously considered, information into the genome, random perturbations are applied through mutation of the newly produced genome. When applicable, this is followed by constraint evaluation, where invalid individuals may either be repaired, penalised or discarded. Finally, the offspring population is evaluated on the objective function and a new parent population is produced.

The specific evolutionary algorithm type chosen in this paper is the $(\mu + \lambda)$-Evolution Strategy. Evolution Strategies (ESs) were developed by Ingo Rechenberg and Hans-Paul Schwefel at the TU Berlin in the 60s and are especially well suited for solving engineering design problems [6]. They are interesting for this work, as they can deal with discrete as well as with continuous design variables, as outlined in Li *et al.* [7]. In Figure 1 the main loop of a $(\mu + \lambda)$-ES is summarised. Basically, the population (multiset) of individuals is used as a template to generate the offspring population M of size λ and from the combination of parents and offspring the best individuals are selected as the parents of the next round. The initialisation, mutation and recombination operators are chosen in a domain specific way, as will be discussed in more detail in Section 5. For a more detailed discussion on evolution strategies and their properties the reader is referred to [5] and [8].

1. Initialise Population $P_0 \in S$
2. Evaluate P_0
3. $t \to 0$
4. While (Termination criterion is not satisfied)
5. $\quad t \to t + 1$
6. $\quad C_t =$ Randomly select λ pairs of individuals from P_{t-1}
7. $\quad R_t = \{recombine(c, c') | (c, c') \in C_t\}$
8. $\quad M_t = \{mutate(r) | \, r \in R_t\}$
9. $\quad$ Evaluate M_t
10. $\quad P_t =$ Select μ best individuals from $M_t \cup P_{t-1}$
11. End While
12. Return best individual in P_t

Figure 1: $(\mu + \lambda)$-Evolution Strategy.

4 PROBLEM

One important task before optimisation can be used is to formulate a search space. In this work, it is also intended to use parameter optimisation to optimise the building design. This makes it necessary to encode the existence or non-existence of spaces by discrete variables. Moreover the connectivity of spaces – the topology of the building spatial design – is encoded by discrete variables. In contrast, the sizing of the spaces is encoded by continuous variables.

In the following, the supercube representation of building spatial designs will be used. It can encode a large search space of building spatial designs by a fixed size vector of discrete and continuous variables. This makes it easy to apply mixed-integer optimisation algorithms directly for optimisation, as will be exemplified by using a mixed-integer evolution strategy later in this work.

The supercube representation has recently been introduced in [9]. It is here assumed a building spatial design consists of maximally N_{spaces} spaces that can be seen as mapped to a $N_w \times N_d \times N_h$ 3D rectangular (cuboid) grid, consisting of $N_w \times N_d \times N_h$ cells. Here N_w, N_d and N_h are the number of subdivisions in width, depth and respectively height. An example of a supercube grid is shown in Figure 2. The size of the cells is determined by the dimensioning variables w_i, d_j and h_k for the width, depth and respectively height of the cells. Here, $i \in \{1, \ldots, N_w\}$, $j \in \{1, \ldots, N_d\}$ and $k \in \{1, \ldots, N_k\}$. The building spatial design is represented by binary variables $b^{\ell}_{i,j,k}$ where i, j, k are again the indices to the width, depth and respectively height and ℓ refers to the different spaces. If $b^{\ell}_{i,j,k} = 1$ this means that the cell with indices i, j, k is part of space ℓ, otherwise this is not the case. Not all configurations of cells make sense from a building spatial design perspective. Constraints will therefore be formulated to restrict the design space to reasonable solutions.

In summary the following variables will be subject to optimisation:

Discrete variables: (all binary)

$$b^{\ell}_{i,j,k}, i \in \{1, \ldots, N_w\}, j \in \{1, \ldots, N_d\}, k \in \{1, \ldots, N_h\}, \text{ and } \ell \in \{1, \ldots, N_{spaces}\}$$

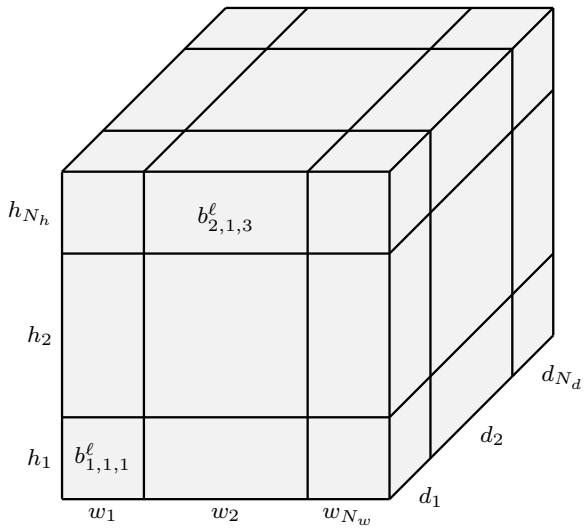


Figure 2: Grid used in the supercube representation.

Continuous variables:

$$w_i, i \in \{1, \ldots, N_w\}, d_j, j \in \{1, \ldots, N_d\}, \text{ and } h_k, k \in \{1, \ldots, N_h\}$$

4.1 Constraints on the discrete variables

In the following four types of topology constraints are introduced, see also [9]. To avoid overlap of spaces every cell can belong to at most one space. This will be enforced by the following constraint in Equation 1.

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

Spaces should have the form of a cuboid (3D rectangle). To enforce this constraint one can first extend the supercube grid by adding a layer of cells with binary variables equal to zero around it (Equation 2).

$$\begin{gathered} \forall_{\ell} : \forall_{i,j,k} \in \{0, \ldots, N_w + 1\} \times \{0, \ldots, N_d + 1\} \times \{0, \ldots, N_h + 1\} : \\ i = 0 \vee j = 0 \vee k = 0 \vee i = N_w + 1 \vee j = N_d + 1 \vee k = N_h + 1 \Rightarrow b^{\ell}_{i,j,k} = 0 \end{gathered} \tag{2}$$

Moreover a i, j beam is defined as a set of cells that share the same i, j index. Accordingly j, k and i, k beams are defined. For all i, j beams and all spaces ℓ transitions from zero to one (Equation 3) should always occur at the same position and transitions from one to zero (Equation 4) for space ℓ should also always occur at the same position. The same holds for j, k and i, k beams (not in equations here).

$$\begin{gathered} \forall_{\ell} : \forall_{i_1,j_1,i_2,j_2} : \left(\left(\sum_{k=1}^{N_h} k \left(1 - b^{\ell}_{i_1,j_1,k-1}\right) b^{\ell}_{i_1,j_1,k} \right) - \left(\sum_{k=1}^{N_h} k \left(1 - b^{\ell}_{i_2,j_2,k-1}\right) b^{\ell}_{i_2,j_2,k} \right) \right) \\ \left(\sum_{k=1}^{N_h} b^{\ell}_{i_1,j_1,k} \right) \left(\sum_{k=1}^{N_h} b^{\ell}_{i_2,j_2,k} \right) = 0 \end{gathered} \tag{3}$$

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

Moreover, to ensure connectedness (if the previous two equations also hold) of the cells of a space ℓ all beams (in all directions) should have at most one transition from zero to one (Equation 5).

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

Building spatial designs normally stand on the ground. This is difficult to check by a simple equation if vertical gaps are allowed in the spatial design. Therefore it is suggested to enforce a no vertical gaps constraint as well. As a result of this constraint it is not possible to describe structures with cantilevers, overhangs or archways. The no vertical gaps constraint could be abandoned if one is willing to use more complex procedures to check constraints. These constraints are simultaneously enforced by disallowing transitions from zero to one for i, j beams. Let $b_{i,j,k}$ be the outcome of a logical OR of all ℓ bits belonging to cell i, j, k. In equations: $\forall_{i,j,k} : b_{i,j,k} = sgn(\sum_{\ell=1}^{N_{spaces}} b^\ell_{i,j,k})$, where the $sgn()$ may be omitted if the no-overlap constraint (Equation 1) is satisfied. If Equation 6 holds, the building spatial design has no vertical gaps and stands on the ground.

$$\forall_{i,j} : \left(\sum_{k=1}^{N_h - 1} \left(1 - b_{i,j,k}\right) b_{i,j,k+1} \right) = 0 \tag{6}$$

The number of described spaces is kept constant. This is achieved by ensuring every space is described by at least one cell (Equation 7).

$$\forall_\ell : \sum_{i=1}^{N_w} \sum_{j=1}^{N_d} \sum_{k=1}^{N_h} b^\ell_{i,j,k} \geq 1 \tag{7}$$

4.2 Constraints on the continuous variables

In the design of buildings the total volume V_0 of the building is could be provided as a constraint. To exclude inactive cells (not part of any building space) from the volume computation here $b_{i,j,k}$ is again taken to be the result of a logical OR over all ℓ bits of a cell i, j, k. This yields the equality constraint in Equation 8 below:

$$\sum_{i=1}^{N_w} \sum_{j=1}^{N_d} \sum_{k=1}^{N_h} b_{i,j,k} w_i d_j h_k = V_0 \tag{8}$$

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

4.3 Objective function

Optimisation is performed with respect to two objective functions from two different disciplines: Minimum compliance, which relates to structural design performance, and total outer surface area, which relates to energy performance.

The *structural design performance* is assessed by first providing the building spatial design with a building structural design. This is carried out by applying a so-called structural grammar on each building space. The grammar selected here adds four walls ($t = 150mm$), with a slab on top ($t = 150mm$), all made of concrete (elasticity modulus: $E = 30000N/mm^2$ and Poisson's ratio: $v = 0.3$). The building spatial design is then loaded with a live load on each floor surface ($1.8kN/m^2$) and wind loads on each outside surface ($1.0kN/m^2$ for pressure, $0.5kN/m^2$ for suction and $0.4kN/m^2$ for shear) from eight general directions (N, N/W, W, etc.). After the transfer of the loads to the building structural design and meshing of the structure, finite element analyses are carried out to find the total compliance for all loads together. Detailed information can be found in the paper by Hofmeyer & Davila Delgado [2]. In summary the first optimisation task is to minimise the total compliance S_C subject to the given constraints.

The *total surface area* objective can be computed for the supercube representation as follows. Note that for this computation it is required that the building spatial design contains no cantilevers or archways, this is ensured by the no vertical gaps constraint previously described with Equation 6. Additionally the computation requires a layer cells with their binary variables equal to zero around the supercube (Equation 2).

Every ray through the supercube in width and depth measures the number of changes from zero to one and then multiplies this number of changes by the area of the fixed indices and then by two to take into account both the entry and exit points. In case of the height direction the multiplication by two is omitted, because for the height direction the connection with the ground layer is not counted as surface area. Since there are no vertical gaps, the height direction is essentially the sum of areas of rays where space exists. The total sum $S_A := S_h + S_d + S_w$ of Equations 9, 10 and 11 below is then the total surface area.

$$S_h = \sum_{i=1}^{N_w} \sum_{j=1}^{N_d} \left(\left(\sum_{k=1}^{N_h+1} \left(1 - b_{i,j,k-1}\right) b_{i,j,k} \right) w_i d_j \right) \tag{9}$$

$$S_d = \sum_{i=1}^{N_w} \sum_{k=1}^{N_h} \left(2 \left(\sum_{j=1}^{N_d+1} \left(1 - b_{i,j-1,k}\right) b_{i,j,k} \right) w_i h_k \right) \tag{10}$$

$$S_w = \sum_{j=1}^{N_d} \sum_{k=1}^{N_h} \left(2 \left(\sum_{i=1}^{N_w+1} \left(1 - b_{i-1,j,k}\right) b_{i,j,k} \right) d_j h_k \right) \tag{11}$$

In summary the second optimisation task is to minimise the surface area S_A subject to the given constraints.

5 OPTIMISATION METHOD SETUP

Next, a description follows of how the earlier introduced $(\mu + \lambda)$-ES is customised for the optimisation based on the supercube representation, including the handling of constraints.

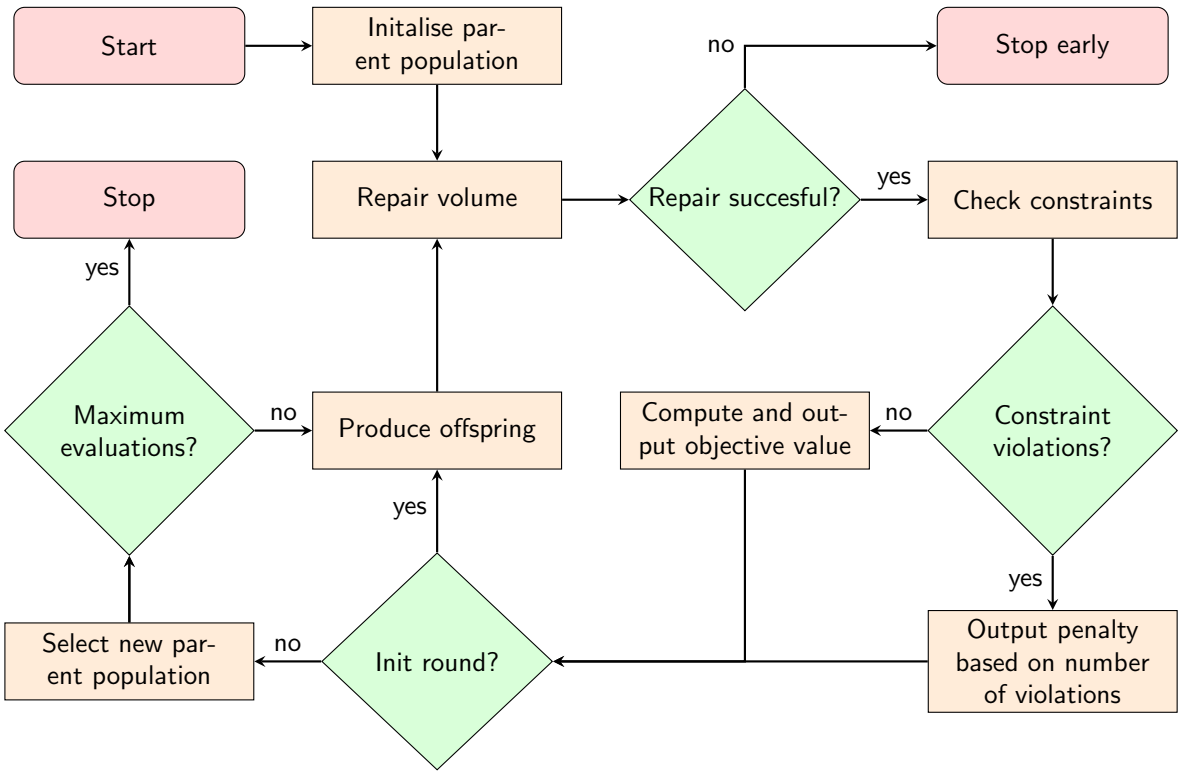


Figure 3: Optimisation outline.

The optimisation procedure is outlined in Figure 3. For convenience three definitions are introduced, the number of cells: $N_{cells} := N_w \times N_d \times N_h$, the number of continuous parameters: $N_{cont} := N_w + N_d + N_h$, and the total number of dimensions: $N_{dims} := N_{cont} + N_{cells} \times N_{spaces}$. The process starts by initialising the parent population of size $\mu = 20$. Here continuous parameters $(x_1, \ldots, x_{N_{cont}})$ are initialised to a uniformly random value in $[lb = 1.5, \ldots, ub = 9.9]$, and the binary parameters $(x_{N_{cont}+1}, \ldots, x_{N_{dims}})$ to one with probability $1/N_{cells}$ or zero otherwise. Step sizes of the continuous parameters are initialised to 0.1. Following this the volume of all new individuals is repaired to be within 1% of the desired volume $2^3 \times N_{cells}$ (a detailed description follows after this outline). In order to check constraints the continuous parameters are then converted to millimetres (multiply by 2000). In case any constraints are violated either a single penalty value of $pen = 999,999,999$ is output or a penalty value equal to $pen + CV - 1$ based on the number of constraint violations $CV \in 1, \ldots, 5$ is output as objective value. Five constraints are considered for the value of CV: All spaces exist, no-overlap, cuboid shape, connected cuboid and no vertical gaps. When no constraints are violated the objective value is computed and output, either surface area in square metres or compliance in Newton metres. For the initialisation round $\lambda = 100$ offspring are produced immediately, all later iterations first make a selection of the μ best (lowest objective value) individuals from the $\mu+\lambda$ individuals (parents+offspring). For every to be produced offspring individual two parents are selected (possibly the same twice) uniformly at random. Using these two parents, crossover is applied to produce a new individual, which is then mutated, and finally repaired if it exceeds the bounds. This is done by Modified Interval Bounds Treatment, where for parameters the previously defined lb and ub parameters are used and for the step sizes bounds $lb_s = 0.01$ and $ub_s = ub \times 0.1$ are used. Intermediate crossover, taking the mean value of the parents, is applied to the continuous parameters as well as their corresponding step sizes. Gaussian mutation with

individual step sizes is applied to the continuous parameters. Their step sizes are mutated by the local learning rate $\tau_1 = 1/\sqrt{2\sqrt{N_{dims}}}$ and global learning rate $\tau_2 = 1/\sqrt{2 \times N_{dims}}$ as in Li *et al.* [7]. Implementation wise a global value g_1 is drawn from a Gaussian distribution $g_1 = G(0, 1)$ for every individual. For every parameter separately a second value $g_2 = G(0, 1)$ and third value $g_3 = G(0, 1)$ are drawn. The step size s of a parameter is then mutated as: $s' = s \times exp(g_1 \times \tau_2 + g_2 \times \tau_1)$. The newly mutated step size is then used to mutate its corresponding parameter x as: $x' = x + g_3 \times s'$. Binary parameters apply crossover by copying the value from one of the parents, chosen uniformly at random for each bit. Binary parameters are mutated by flipping each bit with a probability of $1/(N_{cells} \times N_{spaces})$. Following this the volume of the new offspring individuals is repaired and the loop repeats until the desired number of evaluations is reached. Note that here evaluations are counted based on the number of valid (non-constraint violating) solutions.

A detailed description of volume repair follows. When a newly created individual does not satisfy the volume constraint in Equation 8 its volume is repaired, which is done by scaling the continuous parameters. The desired total volume V_0 is defined for each experiment. The current total volume V_c is simply the outcome of the left-hand side of Equation 8. These two values allow for the computation of a factor $\alpha = V_0/V_c$. The desired volume is then reached by multiplying the continuous parameters by the cubic root of α as shown in Equation 12. Note that this should only be done for active cells in the supercube (cells occupied by at least one building space) since as defined before inactive cells do not contribute to the volume. As a result of the creation process (recombination, mutation) of new individuals and the described volume repair, continuous parameters of individuals may end up exceeding their lower bound lb or upper bound ub. To correct for this, parameters exceeding the lower bound are set to the lower bound, parameters exceeding the upper bound are multiplied by 0.95 until they are within the bound. Clearly the corrections for the bounds affect the volume. Therefore the volume repair and bound corrections are iteratively solved up to 26 times until both requirements are satisfied. When this number of corrections is insufficient for any of the individuals the optimisation process is stopped and considered as unsuccessful. This did not occur in the later presented experiments, the likelihood of this occurring depends on the chosen bounds and the desired volume.

$$\forall_i : w_i = \sqrt[3]{\alpha w_i} \qquad \forall_j : d_j = \sqrt[3]{\alpha d_j} \qquad \forall_k : h_k = \sqrt[3]{\alpha h_k} \tag{12}$$

Based on the described optimisation outline a number of experiments have been carried out. Namely, individual optimisation of both the surface area and the compliance using a single penalty value is used to test how well the proposed supercube description works in practice. Moreover, this will set a baseline for the objective value of both functions. In addition a comparison is made to individual optimisation of the same objectives when using a penalty based on the number of constraint violations. This comparison aims to provide some first insights into constraint handling for this heavily constrained problem.

All experiments are conducted with an evaluation budget of 1000, for six different configurations. A supercube of size two ($2 \times 2 \times 2$ cells) with one, three and five spaces, and a supercube of size three ($3 \times 3 \times 3$ cells), also with one, three and five spaces. After generating one million candidate solutions the optimisation is stopped, even if 1000 valid candidates are not yet found.

6 RESULTS

First experiments with a simple, single penalty, constraint handling technique are reported as shown in Figures 4 and 5. Here infeasible solutions are penalised by a constant penalty that

is higher than all objective function values of feasible solutions. For each plot mean values for different numbers of spaces are shown. Different problem configurations are denoted by four numbers. For instance '1234' would refer to a configuration with a width of one, depth of two and height of three cells, and describing four spaces. Five runs of the optimisation process are performed for every configuration.

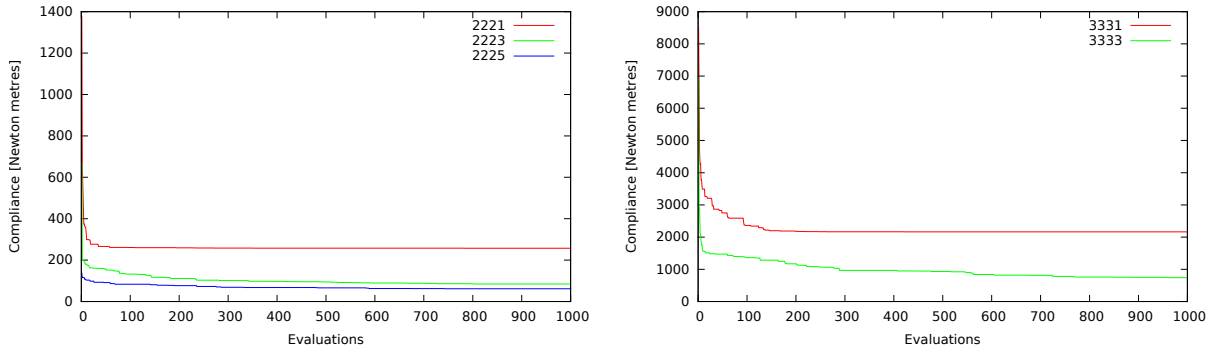


Figure 4: Mean convergence of the compliance for all completed runs (maximum of five) of single penalty 222x and 333x configurations, for one, three and five spaces. 2225 and 3333 completed 3 of 5 runs and 3335 completed 0 of 5 runs. All incomplete runs did not find any valid spatial designs. All not specifically mentioned configurations completed all five runs.

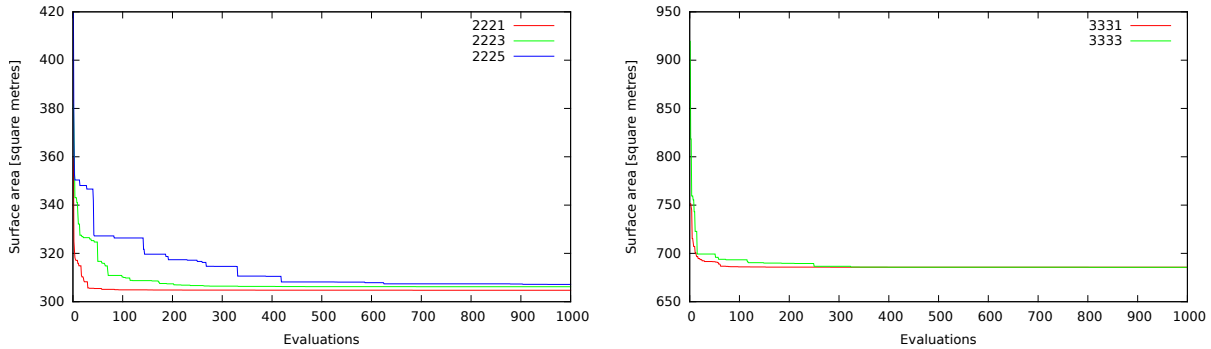


Figure 5: Mean convergence of the surface area for completed runs (maximum of five) of single penalty 222x and 333x configurations, for one, three and five spaces. 2225 completed 2 of 5 runs, 3333 completed 3 of 5 runs and 3335 completed 0 of 5 runs. All incomplete runs did not find any valid spatial designs. All not specifically mentioned configurations completed all five runs.

The results in Figure 4 show convergence plots for the compliance objective. After a rapid decrease in the function value during the first few hundred evaluations the optimisation process tends to stagnate. Configurations 2225, 3333 and 3335 did not find any valid solutions in all of their runs. Namely configurations 2225 and 3333 both completed three of the five runs and configuration 3335 completed none of the five runs. Mean convergence values in the plots are computed over the completed runs, that is, runs that found 1000 feasible solutions. For surface area similar results were found as shown in Figure 5. Here too, some configurations did not complete all their runs, respectively configurations 2225, 3333 and 3335 completed two, three and zero runs.

For the ease of discussion the constraints are summarised here with a reference to the relevant equations. *Existence*, all spaces exist, i.e. they are described by at least one cell (Equation 7). *No-overlap*, each cell belongs to no more than one space (Equation 1). *Cuboid shape*, the cells describing a space together form a cuboid shape, possibly with voids (Equations 3 and 4). *Connected cuboid*, given the cuboid shape constraint holds, all cells forming a space are connected and form a cuboid without voids (Equation 5). *No vertical gaps*, all cells are either on the ground level, or have an active cell on the level below them (Equation 6). Note that for some problem configurations certain constraints are always satisfied, these are indicated as not applicable (N/A).

Table 1 shows the ratios of constraint violations for every configuration and constraint type for the minimal compliance objective with a single penalty value. For small supercube sizes the existence constraint poses no problem. The no-overlap constraint depends on the ratio between the number of spaces and the number of cells. The probability of constraint violation for the 3333 and 3335 configurations is extremely high (0.999218016 and 0.999294014), making it impractical to search. The remaining three constraints show a similar pattern to the no-overlap constraint. The constraint violations of the surface area in Table 2 show a largely similar behaviour, with a large portion of the constraints occurring with high probabilities. Constraint violation appears to be a major problem for this approach.

Configuration	Existence	No-overlap	Cuboid shape	Connected cuboid	No vertical gaps
2221	0.072030329	N/A	0.430918281	N/A	0.350463353
2223	0.148770246	0.373125375	0.528494301	N/A	0.393521296
2225	0.197997775	0.610956619	0.565072303	N/A	0.482480534
3331	0.014877790	N/A	0.590860786	0.160821821	0.501239816
3333	0.000360993	0.999218016	0.999928001	0.999147017	0.999986000
3335	0.816557669	0.999294014	0.999885002	0.999008020	0.999954001

Table 1: Mean constraint violation probability over five runs for minimal compliance optimisation with a single penalty value for various problem configurations.

Configuration	Existence	No-overlap	Cuboid shape	Connected cuboid	No vertical gaps
2221	0.054300608	N/A	0.477410947	N/A	0.288010426
2223	0.176321781	0.433341766	0.589931697	N/A	0.348343031
2225	0.021001580	0.999803004	0.999840003	N/A	0.998622028
3331	0.040989160	N/A	0.525406504	0.168021680	0.483739837
3333	0.080888636	0.195670749	0.656508117	0.158074623	0.558245514
3335	0.816465671	0.999535009	0.999898002	0.999337013	0.999976001

Table 2: Mean constraint violation probability over five runs for surface area optimisation with a single penalty value for various problem configurations.

To remedy this problem a graduated penalty function was created. It penalises based on the number of constraints that are violated. This allows evolutionary search to gradually correct bits and find feasible solutions faster. Clearly Figures 6 and 7 show the success of this strategy where all runs converge and it is now possible to deal with bigger grid sizes (e.g. 3335).

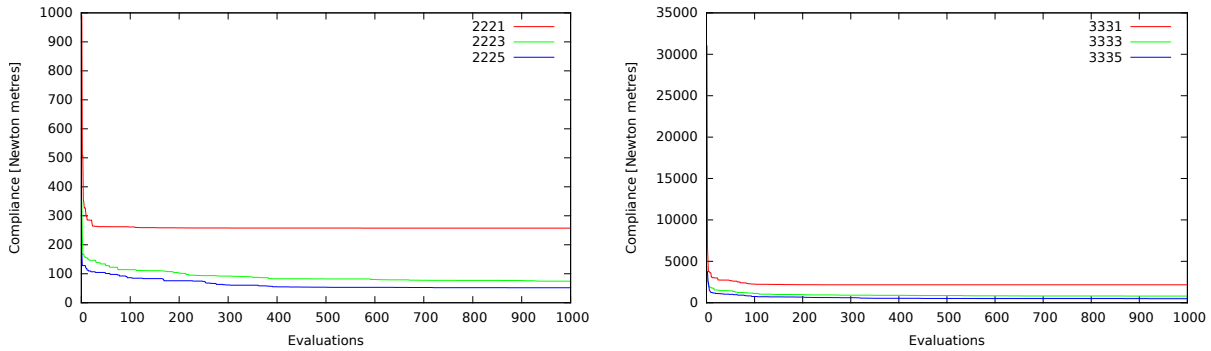


Figure 6: Mean convergence of the minimal compliance optimisation with constraint penalties based on the number of violations for five runs of 222x and 333x configurations, for one, three and five spaces.

In terms of constraint violations it is observed that penalising based on the number of violated constraints vastly improves the chance of finding valid solutions, as shown in Tables 3 and 4.

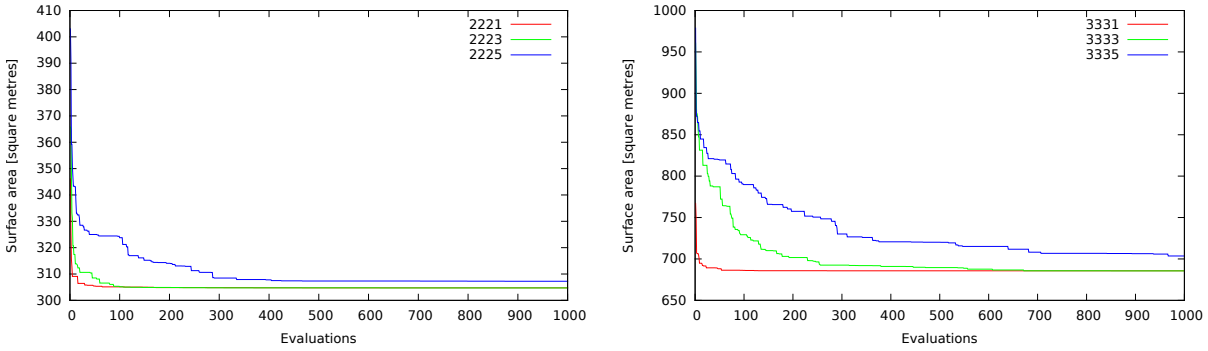


Figure 7: Mean convergence of the surface area optimisation with constraint penalties based on the number of violations for five runs of 222x and 333x configurations, for one, three and five spaces.

Where with a single penalty a significant number of constraints were violated with probabilities above 80% this is now reduced to a single case. Only the existence constraint remains a major problem, and even this is only true for the largest configuration (3335). Even so, many other constraint violations still occur around 50% of the time, and clearly these would benefit from further improvement as well.

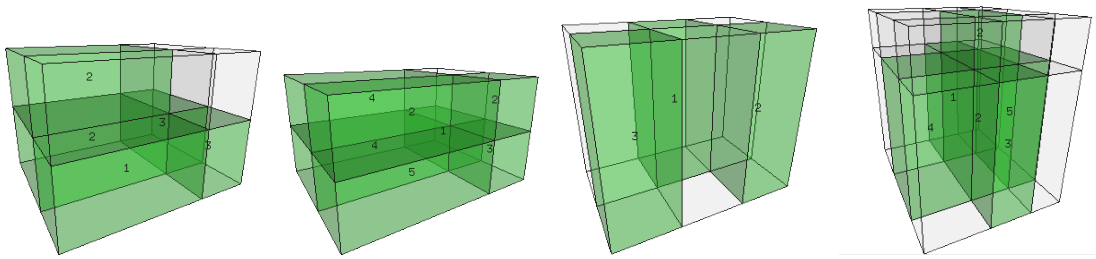
Configuration	Existence	No-overlap	Cuboid shape	Connected cuboid	No vertical gaps
2221	0.066305819	N/A	0.423545332	N/A	0.328371673
2223	0.156955204	0.322667565	0.467497474	N/A	0.313910408
2225	0.305234899	0.484563758	0.455570470	N/A	0.334228188
3331	0.013302295	N/A	0.605919521	0.119388094	0.518789491
3333	0.143656716	0.112873134	0.559701493	0.078358209	0.456778607
3335	0.840781999	0.154225102	0.462875022	0.082638026	0.422732114

Table 3: Mean constraint violation probability with penalties based on the number of violations over five runs for minimal compliance optimisation for various problem configurations.

Configuration	Existence	No-overlap	Cuboid shape	Connected cuboid	No vertical gaps
2221	0.063636365	N/A	0.501581028	N/A	0.269960474
2223	0.144760533	0.320849838	0.453366943	N/A	0.142599928
2225	0.197621226	0.532174443	0.471790170	N/A	0.365965233
3331	0.063481457	N/A	0.521550284	0.105913799	0.467758102
3333	0.142694064	0.146689498	0.582191781	0.099029680	0.469463470
3335	0.859725404	0.193019717	0.495911166	0.101861297	0.420656555

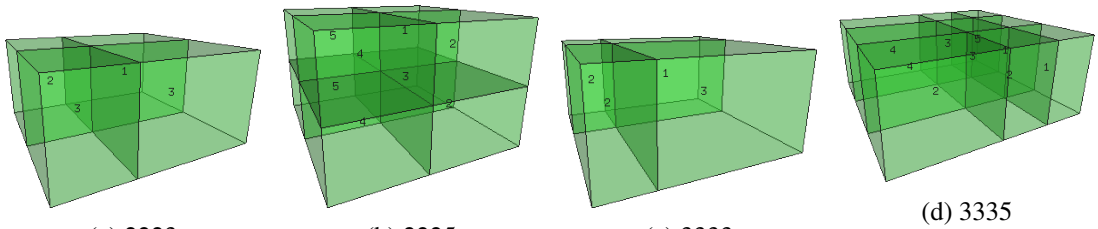
Table 4: Mean constraint violation probability with penalties based on the number of violations over five runs for surface area optimisation for various problem configurations.

The example results of some building spatial designs are given here, it was observed that different configurations resulted in different spatial designs. There is a clear distinction between the results from minimal compliance optimisation (Figure 8) and those of surface area optimisation (Figure 9). Surface area optimisation leads to compact cuboid, or near cuboid, shapes, as might be expected. Minimal compliance optimisation on the other hand produces a variety of shapes. The similarity between both 2225 configurations is striking, but probably the result of the limited variety of space arrangements within an order two supercupe. On the other hand, little use is made of the extra space in the 333x configurations. For the 3333 configurations this may be explained by the availability of only three spaces; there is a limited number of valid building spatial designs that can make use of the larger number of cells with only three spaces. Note that while it is possible to produce spaces consisting of a large number of cells, reaching such a situation becomes increasingly difficult with the number of cells while also satisfying the constraints. For example the largest space in the 3335 configuration for surface area optimisation (Figure 9d) consists of just two cells. This is likely to play a role in the limited use of space for both the 3333 and 3335 configurations. These frequent issues with constraints complicate exploring all feasible solutions in the search space, in particular transitions between different feasible parts of the search space are challenging when many moves end up in infeasible parts of the search space.



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

Figure 8: Examples of output spatial designs of compliance optimisation.



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

Figure 9: Examples of output spatial designs of surface area optimisation.

7 DISCUSSION

A newly described optimisation outline for building spatial designs in early stages of building design has been shown to make effective use of a mixed-integer super-structure representation. The optimisation process was shown to converge for both small and medium sized building spatial designs for two individual objectives, compliance and surface area. Moreover, different constraint handling techniques were applied to significantly improve the ability to traverse a search space in search of valid designs. Both repair procedures and graded penalties were used. Altogether these results form a promising first step in multi-disciplinary design optimisation.

A problem that was encountered is that different runs found different topologies. A possible explanation is that transitions from one feasible subspace to another are very unlikely. In future work this should be addressed by global optimisation strategies such as niching [10]. Another issue is that even the largest problem configurations considered here are not very large compared to most practical building spatial designs. Given a significant portion of the constraints are violated with frequencies in the 40 to 50% range and in the worst cases over 80%, even when using the multi-penalty approach, it is clear that for larger designs more effective constraint handling methods have to be introduced.

Acknowledgements: The authors gratefully acknowledge the financing of this project by the Dutch STW via project 13596 (*Excellent Buildings via Forefront MDO, Lowest Energy Use, Optimal Spatial and Structural Performance*).

REFERENCES

[1] Matti Palonen, Mohamed Hamdy, and Ala Hasan. Mobo a new software for multi-objective building performance optimization. In *Proceedings of the 13th Internationcal Conference of the IBPSA*, pages 2567–2574, 2013.

[2] Hèrm Hofmeyer and Juan Manuel Davila Delgado. Coevolutionary and genetic algorithm based building spatial and structural design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 29(04):351–370, 2015.

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

[4] Christina J. Hopfe, Michael T.M. Emmerich, Robert Marijt, and Jan L.M. Hensen. Robust multi-criteria design optimisation in building design. *Proceedings of Building Simulation and Optimization, Loughborough, UK*, pages 118–125, 2012.

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

[6] Ingo Rechenberg. *Evolutionsstrategie '94, volume 1 of Werkstatt Bionik und Evolutionstechnik*. Frommann Holzboog, Stuttgart, 1994.

[7] Rui Li, Michael T.M. Emmerich, Jeroen Eggermont, Thomas Bäck, Martin Schütz, Jouke Dijkstra, and Johan H.C. Reiber. Mixed integer evolution strategies for parameter optimization. *Evolutionary computation*, 21(1):29–64, 2013.

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

[9] Sjonnie Boonstra, Koen van der Blom, Hèrm Hofmeyer, Robert Amor, and Michael T.M. Emmerich. Super-structure and super-structure free design search space representations for a building spatial design in multi-disciplinary building optimisation. Submitted for EG-ICE 2016, Poland June 29 – July 1, 2016.

[10] Ofer M. Shir, Michael T.M. Emmerich, and Thomas Bäck. Adaptive niche radii and niche shapes approaches for niching with the CMA-ES. *Evolutionary Computation*, 18(1):97–126, 2010.