The Prism-Net Search Space Representation for Multi-objective Building Spatial Design

Ksenia Pereverdieva[1], Michael Emmerich[1(✉)], André Deutz[1], Tessa Ezendam[2], Thomas Bäck[1], and Hèrm Hofmeyer[2]

[1] Leiden University, Leiden, The Netherlands
m.t.m.emmerich@liacs.leidenuniv.nl
[2] Eindhoven University of Technology, Eindhoven, The Netherlands

Abstract. A building spatial design (BSD) determines external and internal walls and ceilings of a building. The design space has a hierarchical structure, in which decisions on the existence or non-existence of spatial components determine the existence of variables related to these spaces, such as sizing and angles. In the optimization of BSDs it is envisioned to optimize various performance indicators from multiple disciplines in concert, such as structural, functional, thermal, and daylight performance. Existing representations of design spaces suffer from severe limitations, such as only representing orthogonal designs or representing the structures in parametric superstructure, allowing only for limited design variations. This paper proposes *prism nets* - a new way of representing the search space of BSDs based on triangulations defining space filling collections of triangular prisms that can be combined via coloring parameters to spaces. Prism nets can accommodate for non-orthogonal designs and are flexible in terms of topological variations. We follow the guidelines for representation and operator design proposed in the framework of metric-based evolutionary algorithms. The main contribution of the paper is a detailed discussion of the search space representation and corresponding mutation operators. Moreover, a proof of concept example demonstrates the integration into multi-objective evolutionary algorithms and provides first results on a simple, but reproducible, benchmark problem.

Keywords: Building spatial design · Mutation operators · Geometry optimization · Non-standard representations · Multi-disciplinary design

1 Introduction

One of the advantages of evolutionary algorithms, compared to most classical optimization algorithms, is that they can accommodate complex search spaces

The authors gratefully acknowledge financial support by NWO, The Netherlands, TOP Grant 18036: Excellent Buildings: Realistic geometries for optimal structural, thermal, and lighting performance.

M. Emmerich et al. (Eds.): EMO 2023, LNCS 13970, pp. 476–489, 2023.
https://doi.org/10.1007/978-3-031-27250-9_34

with variable dimension, such as the space of mathematical expressions [19], the chemical space consisting of molecules represented by chemical graphs [20], or various types of structures in engineering design [14] or neural architectures [15]. In the following we propose a non-standard representation of a search space for multi-objective BSD optimization.

The domain of building spatial design (BSD) is concerned with finding optimal layouts for buildings, including internal and external walls, floors and ceilings. The building spatial design crucially governs the performance of a building in terms of various performance indicators (objectives), such as energy performance (which is related to the outer surface area), structural performance (strength, stiffness, and stability), and daylight performance (related to the size and positioning of windows). In a previous project, an open source building spatial design optimization toolbox (BSO toolbox [4,7]) has been developed by researchers of the Eindhoven University of Technology, The Netherlands, and of Leiden University, The Netherlands. The toolbox supports the human designer in the task of multi-criteria and multi-disciplinary building spatial design. So far it is restricted to BSDs based on orthogonal space partitioning and it features building physics (energy performance) and structural engineering disciplines (structural performance) [6]. The BSO toolbox uses a collection of quad-hexahedrons to represent a Building Spatial Design (BSD). It then includes adaptive grammars that provide a discipline related design to the BSD (e.g. a structure system with among others flat shells, loads, and boundary conditions) including the properties for discipline specific analysis. The grammars can also function via evolutionary algorithms as described in Boonstra et al. [5]. Finally, the toolbox includes a Finite Element Method (FEM) simulation-based evaluation of the structural performance of BSDs, a Resistor Capacitor (RC) network based evaluation of thermal performance, and various design modification and constraint handling techniques. Another example of approach to BSD is generating floorplan designs [13]. Main features of this approach is simplicity of usage in practice and a new model of human-computer interaction. However, our approach allows more automation, non-orthogonal shapes, and optimization based on energy and structural performance of a building. The multiobjective optimization is accomplished by Pareto optimization using state-of-the-art optimization algorithms. The main optimization algorithm is a hybrid memetic multi-objective optimization algorithm [3] that is used to optimize layout choices, discrete variables as well as continuous variables (using local hypervolume gradient-based search [3]). Optimization is further explored by hybrid approaches that combine the algorithm with design process simulations [6]. The data generated during the optimization process can be interpreted by an explainability engine, which relates regions on the Pareto front to features of the building spacial design that are expressed in terms of the decision variables. A major downside of this system was that it was limited to orthogonal spatial designs, however, progress is made in allowing non-orthogonal BSD constrained to a collection of horizontal floor and vertical walls quad-hexahedrons [11]. Our vision in this new paper is to also represent more complex geometries of buildings in the BSO toolbox, namely,

BSDs with vertical walls but non-orthogonal floorplans or angles between walls. See Fig. 1 for examples of orthogonal designs, a realization of a design by the Dutch company 'De Twee Snoeken', and a non-orthogonal BSD.

To accomplish search spaces that comprise BSDs with more complex geometries we are going to propose the new *prism-net* representation. The *prism-net* search space accommodates all multi-floor building spatial designs with vertical and straight walls and horizontal floors and ceilings, and it can be integrated into evolutionary algorithms by augmenting it with mutation operators that will also be described in this paper. Importantly, the angles of corners of spaces are not restricted to right angles, allowing for more architectural freedom in the design and potentially a further improvement of the various design objectives. Together with the new *prism-net* representation we present a hierarchical mutation operator that encodes a scalable random modification of the building and is guided by the principles of mutation operator design as stated in Rudolph [18] for integer spaces and later refined in Droste and Wiesmann for metric spaces [10]. In brief, the principles are accessibility (every point should be accessible by a finite number of mutations from any other point), symmetry (reversibility), unbiasedness (maximum entropy), and scalability (of the mutation strength). They proved to lead to excellent results in evolutionary optimization when applied to non-standard search spaces such as integer vectors [18], binary decision diagrams [10] and (variable-dimensional) mixed-integer search spaces [16]. Our representation (the prism networks) consists of three levels - topological (triangulation of levels), categorical integer (assignment of prisms to spaces), and continuous (placement of corners or nodes of the triangulation). Constraints are introduced to express the concept of space in a building spatial design, and to accommodate practical needs, such as the avoidance of sharp corners, other geometrical preferences, and the connectivity of the building to the ground. Note that *this paper focuses on building representation for optimization, and not on benchmarking*, since there is no system with similar functionality to compare the results with.

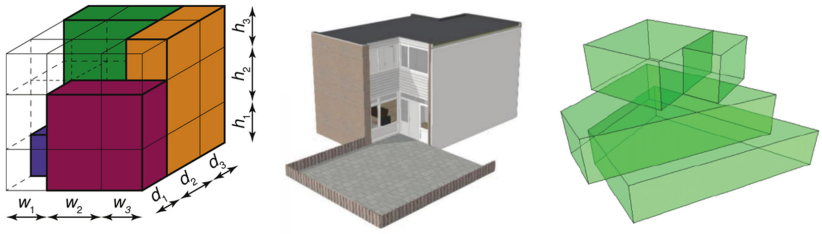

Fig. 1. Orthogonal BSDs (left) and a model of a building (middle), and BSDs based on quadrilateral floorplans with non-orthogonal elements (right).

The paper is structured as follows: In Sect. 2 we introduce the new search space representation for BSDs) and analyse its theoretical properties. In Sect. 3 we propose an hierarchical and scalable mutation operator for the BSDs that

generates neighboring solutions in the search space. In Sect. 4 we show how the new representation can be integrated into existing evolutionary multi-objective optimization algorithms. Also first, reproducible, Pareto optimization results with simple to encode performance indicators are presented. We conclude this work with an outline to future research steps needed to fully integrate the BSD representation into real-world computer aided design optimization environments, such as the *BSO toolbox* (Sect. 5).

2 Search Space Representation

Firstly, let us introduce the three-dimensional Cartesian coordinate system $\mathbb{R}^3$. Each point can be described through three coordinates $\mathbf{p} = (x, y, z)$ with origin $O = (0, 0, 0)$. The plane $(x, y, 0)$, $x \in \mathbb{R}$, $y \in \mathbb{R}$ will be denoted by xOy.

The *first assumption* we make is that all ceilings and floors are parallel to the plane xOy and all walls are parallel to the z-axis. Hence we can express the building layout through a set of two-dimensional projections onto the plane xOy for each level. The number of levels L and the heights of levels are given by the variables: $(h_0, h_1, \ldots, h_L)$. Representation implies that the building can be devided into levels, but at the same time a space can be located on several levels. Note, that these are building spatial designs with flat roofs and ceilings.

The *second assumption* is that we are given a 3-D cuboid (more specifically, an axis aligned 3-D orthogonal polyhedron) V in which the building is positioned. For clarity of presentation, we might for now consider that the cuboid has sides parallel to the axes and one of the vertices coincides with the origin: $V = \{(x, y, z) : \ x \in [0, x_V], y \in [0, y_V], z \in [0, z_V]\}$, where x_V, y_V and z_V are the predefined maximal width, depth and height of a building correspondingly. Subsequently, the entire specified volume will be partitioned into prism shaped cells, of which some will be selected (active cells) and define the building, whereas non-selected cells partition the space not part of the building. Cells in the interior of the building can be combined to *spaces*, i.e., compartments of the BSDs the points of which are not separated by walls. Each cell is a triangular prism fully located on one of the levels. We will denote the number of cells as N_{cells} and the set of cells as $C = \{c_i\}, i \in \{1, 2, \ldots, N_{cells}\}$. Because of the first assumption we made and the fact that we require the triangular prisms to be confined to two adjacent levels, it is possible to describe each cell c_i in the following way: $c_i = [(x_{1i}, y_{1i}), (x_{2i}, y_{2i}), (x_{3i}, y_{3i}), l_i, s_i]$, where (x_{ki}, y_{ki}), – coordinates of the vertices of a triangular prism, $k \in \{1, 2, 3\}$, $l_i \in \{1, 2, \ldots, L\}$ – level on which the cell c_i is located (L denotes total number of levels), and $s_i \in \{0, 1, \ldots, N_{spaces}\}$ – integer categorical variables referred to as 'colors', defined further in the text (N_{spaces} is the maximum number of spaces to be represented). The set of prisms should form a partition of V and the sets of prisms for a given level should form a partition of that level, and all prisms should have non-zero volume, meaning that the vectors (x_{1i}, y_{1i}), (x_{2i}, y_{2i}), and (x_{3i}, y_{3i}) are not allowed to be co-linear. It is desirable to be able to create building designs having an external shape other than the outer volume V (e.g, a box),

and spaces to have other shapes than triangular prisms. We introduce coloring scheme of triangular prisms to combine them into polygon-shaped spaces and hence building. Each cell c_i is associated with a non-negative integer variable $s_i \in \{0, \ldots, N_{colors}\}$. We will call the values of these integer variables *'colors'*, and they denote the space (an interior compartment that is not separated by walls) to which a cell belongs. The user-defined maximum for the number of spaces in the building is denoted by N_{colors}. If $s_i = 0$, then cell c_i is inactive, meaning that it does not belong to the building. If $s_i \in \{1, \ldots, N_{colors}\}$, then it is part of the building. If $s_i = s_j = n \neq 0$, $i \neq j$, $n \in \mathbb{N}$ then both cells c_i and c_j are parts of the same space s_n. See the example of one level in Fig. 2 (left). Here we see that only cells c_2, c_3, c_4, c_5, c_6, c_7, c_8, and c_{10} represent actual parts of the level. And cells c_2 and c_6 are combined into the space with color equal to 4 (red), cells c_5 and c_8 are combined into the space with color equal to 2 (yellow), cells c_7 and c_{10} are combined into the space with color equal to 1 (green), and cells c_3 and c_4 are combined into the space with color equal to 3 (purple). Other cells are technically present in the representation of this floor, but do not represent any part of the building.

Next we will define prism-nets as a data-structure and search space representation that is based on collections of triangular prisms of the aforementioned type and satisfy certain elementary constraints, given below. Here and further, "triangle" means the projection of triangular prisms (cells) on the xOy plane.

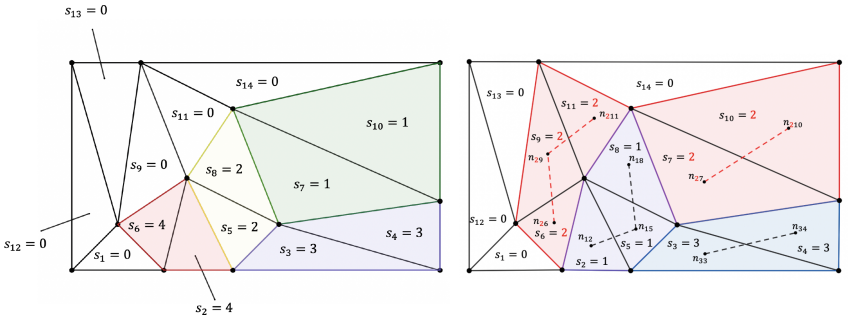

Fig. 2. Projection of one particular level to the plane xOy (left). Connectedness for the projection of one particular level to the plane xOy (right). (Color figure online)

Constraint 1: Non Overlap. Two cells on the same level should not overlap each other. It means that two triangles can intersect each other only in two cases: their intersection is a common vertex, or it is a common side of both triangles. Recall, that previously, we said that the space is partitioned by the prisms. However, this is not exactly true, because we allow the overlap to be of zero measure (that is overlap at the boundary). This way we can view prisms as closed sets.

Constraint 2: Complete Coverage. Every point in the volume V is covered by at least one cell.

Constraint 3: Connectivity of Spaces. As described earlier the color of a cell is a non-negative integer parameter which represents the space to which the cell belongs (positive values), or is zero for cells outside the building. It is necessary to prevent cases where cells with the same positive color are not connected to each other. Next, we consider separately two situations: when cells are on the same level and when they are not. To check if cells located on the same level with the same color are connected, we introduce a test based on the idea of a dual graph (see Fig. 2 (right)). The dual graph is constructed in the following way: nodes of the graph represent cells, and two nodes are connected if corresponding cells are connected (their intersection is either a face, a vertical edge, a vertex or a horizontal edge). We suggest to check if the dual graph is connected. The second condition is that the space has the same projection on every level, i.e. if we consider parts of the space belonging to the same level, or "layers" of the space, then all layers should have the same shape and location in 2-dimensional view. And moreover, a space should be located on adjacent levels.

Remark: Constraints 1–3 are intrinsic constraints, which means that they define constraints of the prism net representation. Constraint 1 and 2 guarantee that the projection of the prism net to the xOy plane is a triangulation for some set of nodes, which partitions the region V. Constraint 3 is intrinsic to the definition of spaces, making sure that spaces (regions of cells with the same color) are not separated by means of walls, floors or ceilings internally.

Definition 1. *(Properly colored) prism net: A* ***prism net*** *is a list of colored triangular prism cells* $c_i = [(x_{1i}, y_{1i}), (x_{2i}, y_{2i}), (x_{3i}, y_{3i}), l_i, s_i]$, $i = 1, \ldots, N_{cells}$ *positioned on level planes that are parallel to the ground-floor* ($z = 0$) *of a given outer cuboid* V, $l_i \in \{1, \ldots, N_{levels}\}$, *with colors* $s_i \in \{0, \ldots, N_{colors}\}$ *and contained in the cuboid* V. *The height of each triangular prism is the height difference of two consecutive level planes, of which* l_i *is the index of the lower of the two consecutive planes. In addition, in a prism-net also the 'partitioning' constraints* 1 *and* 2 *must be satisfied. – A* ***properly colored prism net (PCPN)*** *also satisfies constraint 3 (connectivity of spaces).*

Next we will define some further constraints on prism nets that turn out to be useful when implementing constraint checking or that are motivated by practical constraints for real buildings.

Constraint 4: Convex Polygon. Projections of spaces should form convex polygons. This constraint was added to keep the overall BSDs simple and to avoid costly constraint checking procedures. From a building engineering perspective, however, it is also possible to realize non-convex spaces, such as L-shaped spaces. In the current toolbox, only spaces can be handled with 4 corners. An L-shaped building can be exactly (or approximately) partitioned by triangular cells.

Constraint 5: Spaces should be Connected to the Ground. The next considered constraint is connectivity to the ground. Figure 3 illustrates different types of connection of the spaces. For our example problem we allow all types of connection except for the connection between the red space and all other spaces. To formulate this constraint more strictly we need to introduce a dual graph $G = (N, E)$. The set of nodes N corresponds to the set of spaces. Two nodes n_i, $n_j \in N$ are connected by an edge $e_{ij} \in E$ if the intersection of two corresponding spaces is not an empty set. Black lines in Fig. 3 (left) represent edges of the graph G. Constraint 5 is considered violated if there is no path from any space to at least one space on the ground level.

Remark: Note that here we allow the connection of spaces via a single point. Although this solution is feasible it is expected to perform poorly for a structural objective function.

Constraint 6: No Cavities. In this representation we would like to exclude the possibility of cavities. An example of a cavity mentioned above is illustrated in Fig. 3 (right) (the cell between "blue", "pink" and "green" spaces is the cavity). To perform such a check, it is necessary to determine the cells with a color value of zero which have a side on the border of the building, and make sure that all other cells with zero color value are connected to them. To do it we need to introduce a dual graph $G_0 = (N_0, E_0)$. The set of nodes N_0 corresponds to the set of cells with zero color value. Two nodes n_i, $n_j \in N$ are connected by an edge $e_{ij} \in E_0$ if the intersection of two corresponding cells has two vertices, i.e. the whole side (See Fig. 6). Constraint 6 is met if there is a path from every cell with zero color value to at least one cell with zero color value located on the boundary of the building. **Remark:** If constraint 4 and constraint 6 are met then on each level all cells belonging to the same space are connected and there are no cavities present in the building structure, and therefore external boundaries on each level of each space is one of a closed chain of sides.

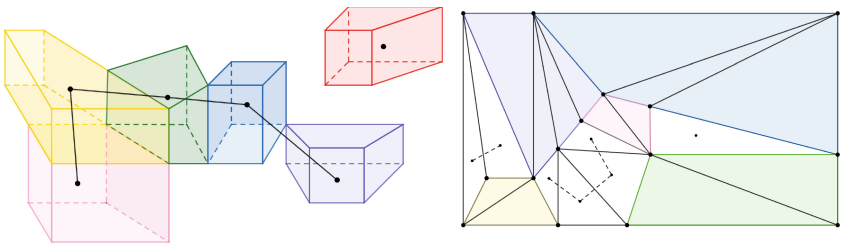

Fig. 3. Types of space connections (left). Allowed and not allowed location of cells with zero color value (right).

Constraint 7: Non Sharp Angles (optional). In addition we introduce a constraint that can be switched off by a user. This constraint avoids spaces

with angles less than some threshold. A user selects the minimal allowable angle and after that we calculate each angle in each space and check it against the threshold.

3 Mutation Operators

In order to formulate search algorithms, it is essential to define operators that generate neighbors of a given design. Next we introduce *hierarchical mutation operators* that can be used in (multi-objective) evolutionary optimization to create variations of a given design in the prism-network representation. It consists of the three following main operations. (1) Topological mutation of the cell partitioning, (2) Changing the discrete s-values ('colors'), (3) Changing the continuous coordinates of the vertices.

By **topological mutation** we will refer to a building layout transformation that changes the triangulation of a convex quadrilateral space without changing the boundaries of spaces. We need to define possible operations of mutation in such a way that no constraint becomes violated. There are three topological mutations suggested: diagonal 'flip' (change of diagonal), adding a vertex, and deleting a vertex. The probability of applying each of them is determined by mutation rate. Next, we will focus on each of them in more detail. *Diagonal flip* chooses randomly one of the convex quadrilaterals formed by two triangle cells belonging to the same space and the same level. The common edge of two cells is the diagonal of mentioned quadrilateral. The mutation is a changing of this diagonal to the other diagonal of the quadrilateral as shown in Fig. 4a).

Adding a vertex splits into two cases: adding a vertex to an edge and adding a vertex to the interior of a cell. When adding a vertex occurs, first type of adding appears with probability 0.9 and the second one with probability 0.1. This ratio was picked empirically. For adding a vertex to an edge we randomly choose a side of a triangle. If two triangles have coinciding sides, we count them as one. Then we uniformly choose a point belonging to this side and add it as a vertex. And finally we split adjacent cells to avoid constraints violation. If the chosen side was on the boundary of two spaces (Fig. 4, c)) or if it was on the side inside a space (Fig. 4, e)), then we need to add two sides coming out of the selected vertex and corresponding cells. If the side was on the outer contour of the building structure (Fig. 4, d)), then we need to add only one side and corresponding cells. The second possible case is adding a vertex to the interior of a cell. We randomly choose a cell and uniformly select a point inside of it which becomes a vertex. And finally we add three sides and corresponding cells (see Fig. 4 b)).

Deleting a Vertex. Since we include the operation of adding a vertex then we also need a possibility of deleting a vertex so that the mutation operator is symmetrical. Firstly we determine the type of each vertex of the building structure. If the vertex is on the corner of the outer contour, then we cannot delete it. If the vertex is on the outer contour, but not on the corner, (Fig. 4, d) we allow

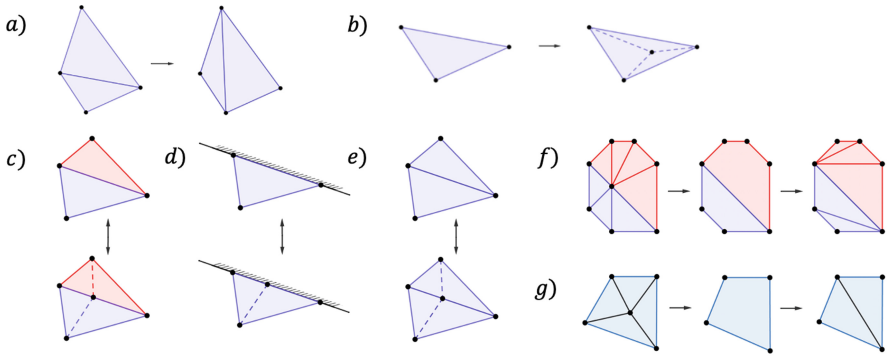

Fig. 4. Topological mutation: a) diagonal flip, b) adding a vertex to the interior of a cell, c)–g) adding a vertex to an edge/deleting a vertex.

to delete it only if it belongs exactly to one space (zero or non-zero space). If a vertex is in the building interior, then we allow to delete it only if it belongs to one (Fig. 4, e) or two (Fig. 4, c)) different spaces (zero or non-zero spaces). These conditions are justified by necessity of leaving only the unfolded angles after deleting a vertex. Secondly, we randomly select one of the vertices which are allowed to be deleted, we delete the vertex and combine the cells in a way that no constraint is violated as it is shown in Fig. 4 (c, d and e) for allowed cases. But sometimes additional triangulation might be needed. In Fig. 4 there are two cases when it is needed: f) with involvement of two spaces, g) in the interior of one space. We use standard Delaunay triangulation [8].

Next, we describe the **Discrete variable mutation** for the s-values (i.e. the 'colors'): Firstly we randomly choose the integer number from the set $\{0, \ldots, N_{spaces}\}$. If the chosen number is 0, then we randomly choose a cell with non-zero s-value and change it to 0. If the chosen number is not 0, we consider the space with s-value equal to the chosen number. From the set of cells belonging to other spaces and connected to the chosen space we pick a random number of cells and "color" them into the chosen color (s-value). After performing one of these two colorings, we check if all constraints are met. If one of them is violated, we skip this mutation. Finally, let us describe the **continuous parameter mutation**: The vertices of the triangulation can be moved to introduce topological changes on a particular level if the vertices are not on the corner of the building projection. If a randomly chosen vertex does not belong to the surface of polyhedron V, it is restricted to the space that is defined by the polygon formed by the triangles that are adjacent to the vertex. To move the vertex we randomly choose the angle from the half-open interval [0, 360), calculate the distance along the selected angle between the vertex and the polygon side, and use a truncated normal distribution with standard deviation equal to the obtained distance divided by 3 to generate the updated location for the vertex. If a randomly chosen vertex belongs to the surface of V, we identify the

segment of the border on the projection to which the vertex belongs, randomly choose one of two directions, and similarly move the vertex according to the truncated normal distribution.

4 Integration to NSGA-II and SMS-EMOA

In order to test the new representation, we integrated it into two state-of-the-art evolutionary multi-objective optimization algorithms[1], the NSGA-II [9] and SMS-EMOA [1] algorithms, and performed multi-objective optimizations with two easy-to-reproduce example objective functions (see Github repository [17] for Python codes): (1) Minimize the external surface area, excluding the floor area (f_1). $f_1 := S_v + S_h \longrightarrow \min$, where S_v - surface area of all vertical external sides of the building, $S_h = \sum_{i=2}^{L}(s_i - s_{i-1})$, s_i – surface area of level i, L – number of levels. (2) Minimize the sum of deviations of space volumes from target predefined volumes. Here we specify the sizes of spaces and seek to minimize the absolute deviation from the prescribed sizes (f_2).$f_2 = \sum_{j=1}^{N_{spaces}} |V_j^a - V_j^d| \longrightarrow \min$, where V_j^a – actual volume of space j, V_j^d – predefined volume of space j.

These objective functions are motivated by resource efficient light-weight constructions. However, the ambition of the overall project is to state objectives that also include energy performance, which can for instance be measured using resistor networks, and structural performance, which can be computed using FEM simulations [6]. However, we would like to abstain in this paper from the details of simulation and are more interested in a problem that is reproducible and easy to understand for non-domain experts.

Three experiments were carried out with the described objectives with different values of the mutation rate. Each of the experiments contained 30 repeated runs. The NSGA-II and SMS-EMOA algorithm were tested. For all runs the same initial population (size: 10) was used. The initial population was set manually since randomized generation of building designs is to be done in future work. In the proposed experiment there are several invariants: the number of levels of the building is 2, the height of the building is 2 (1 for each floor), and the number of spaces is 3. The box V inside which the building is contained has dimensions $x_V = 5$, $y_V = 3$, and $z_V = 2$. Throughout the experiments, variables were limited by these values. The values of 100, 5 and 30 were chosen as the required space volumes for calculation of the second objective as V_1^d, V_2^d, and V_3^d correspondingly, and the value of 50 degrees was chosen as the minimum allowable angle of a space.

Since the values of the objectives differ significantly from each other, normalization is needed. The value of the surface area of no more than 75 was obtained experimentally, so it was decided to divide the absolute value of this objective

[1] Both algorithms feature parameterless selection in the bi-objective case. SMS-EMOA is highly competitive across a wide range of bi-objective problems [2]. NSGA-II is considered to be a commonly used algorithm for bi-objective optimization, whereas, for problems with more objectives, we would rather consider NSGA-III.

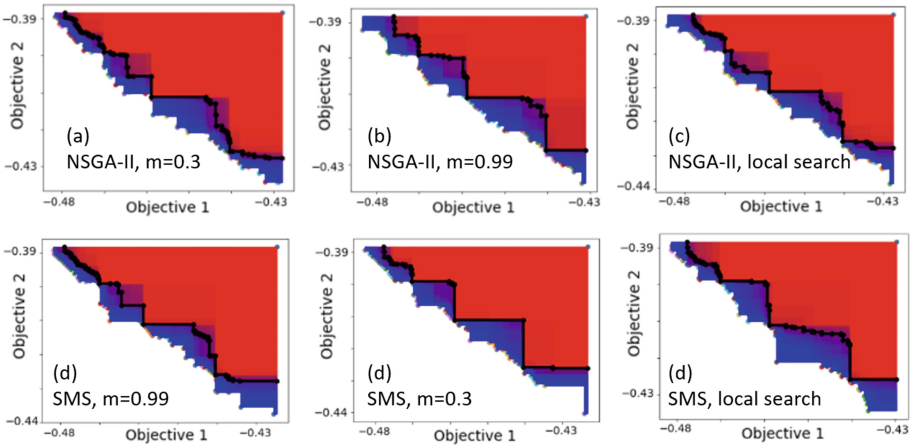

Fig. 5. Summary Pareto fronts (Empirical Attainment Levels [12]). Full red points are attained by all runs, full blue points by just one run, and for points with sliding shades of the color between blue and red are attained by 2 or more runs and less than 30 runs. The black curve marks the median attainment curve. (Color figure online)

by 150, and the value of the deviation from the specified volumes was divided by twice the sum of the required volumes. Thus, for the value of both objectives to lie in the interval $(0, 1)$ was achieved.

Each run of the NSGA-II algorithm was given a budget of 60 generations, within each the proposed mutation operator was used. The recombination operator was not used. Mutation rate in this case determined the probability of applying each of the mutations in the following sequence: topological mutation, discrete parameter mutation, continuous parameter mutation. If any of the mutations did not occur, the algorithm moved on to the next mutations in the list.

In the first of the experiments, the probability of using each of the mutations is 0.99, in the second with probability 0.3, and the third experiment can be considered as local search, in which the probability of topological and discrete mutations was 0.1, and continuous – 0.8. Firstly, points in objective space of all runs from 60 generations were combined and then sorted by means of Pareto dominance. Figure 5 (a)–(c) depicts the obtained Pareto non-dominated. We use *attainment curves* [12] in the plot (Attained by all runs (best), attained by half runs (median), attained by one run (worst)). The hyperparameter optimization is to be done in future work.

There are three examples of building designs presented in Fig. 6. Mutation rate 0.99 was set in order to obtain these designs. The knee point was chosen as the solution for which the objective 1 and objective 2 are closer to each other than for any other solutions among all 30 runs of the algorithm. Figure 6 also shows two extreme values: a design with a minimum value of objective 1 among all solutions and a design with a minimum value of objective 2. As you can see

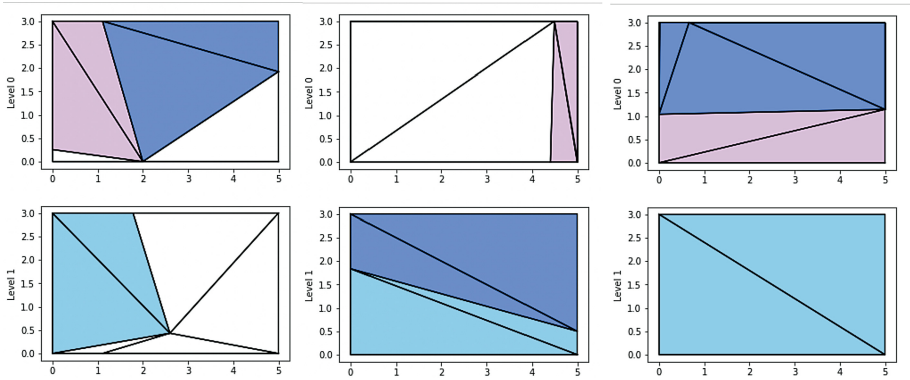

Fig. 6. Building design examples for NSGA-II with mutation rate 0.99: knee point and optimal solution in f_1 and in f_2.

from Fig. 6, optimization allows you to get extreme solutions corresponding to the objectives. Thus, for instance, to minimize the deviation of space volumes from the values of 100, 5 and 30 (for pink, blue and green spaces correspondingly), and for the maximum surface area, the algorithm outputs a design with a fully occupied volume V (Fig. 6, c)). And for minimal surface area the algorithm obtained a building design with a very small red space on level 0. None of the spaces disappeared, as the number of spaces was required to be constant. Three experiments were carried out with the described objectives with different values of the mutation rate. Each of the experiments contained 30 runs of the SMS-EMOA algorithm. Inputs used were completely the same as for NSGA-II algorithm. Obtained Pareto fronts are illustrated in Fig. 5 (d)–(f). Both algorithms produced almost linear Pareto fronts, however, Fig. 5 shows that the SMS-EMOA algorithm takes into account Objective 2 more than Objective 1 during optimization, unlike NSGA-II, where the solutions on Pareto fronts are distributed more evenly. Besides, for both algorithms Pareto fronts vary for different mutation rates. So, for the NSGA-II Pareto algorithm, the local search front is more sparse than for the mutation rate of 0.99 and 0.3, and for the SMS-EMOA algorithm, on the contrary, it is less sparse. The reasons for this behavior have yet to be understood. A promising idea is also to adapt mutation rates for the different mutation types by reinforcement learning [15].

5 Summary and Outlook

A new, non-orthogonal BSD representation was developed, equipped with a domain-specific hierarchical mutation operator, and integrated into MOEAs. The result forms an important step towards extending the design support systems (e.g. the BSO toolbox [4,7]). All data and a detailed description of algorithms are available in the GitHub repository [17].

References

1. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. Eur. J. Oper. Res. **181**(3), 1653–1669 (2007)
2. Bezerra, , L.C., Manuel López-Ibáñez, M., Stützle, T.: Automatic component-wise design of multiobjective evolutionary algorithms. IEEE Trans. Evolut. Comput. **20**(3), 403–417 (2016)
3. van der Blom, K., Boonstra, S., Wang, H., Hofmeyer, H., Emmerich, M.T.M.: Evaluating memetic building spatial design optimisation using hypervolume indicator gradient ascent. In: Trujillo, L., Schütze, O., Maldonado, Y., Valle, P. (eds.) NEO 2017. SCI, vol. 785, pp. 62–86. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-96104-0_3
4. Boonstra, S.: BSO toolbox (2018). https://doi.org/10.5281/zenodo.3823893
5. Boonstra, S., van der Blom, K., Hofmeyer, H., Emmerich, M.T.: Conceptual structural system layouts via design response grammars and evolutionary algorithms. Autom. Constr. **116**, 103009 (2020)
6. Boonstra, S., van der Blom, K., Hofmeyer, H., Emmerich, M.T.: Hybridization of an evolutionary algorithm and simulations of co-evolutionary design processes for early-stage building spatial design optimization. Autom. Constr. **124**, 103522 (2021)
7. Boonstra, S., van der Blom, K., Hofmeyer, H., Emmerich, M.T., van Schijndel, J., de Wilde, P.: Toolbox for super-structured and super-structure free multi-disciplinary building spatial design optimisation. Adv. Eng. Inform. **36**, 86–100 (2018)
8. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Delaunay triangulations: height interpolation. In: Computational Geometry, pp. 191–218. Springer, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-77974-2_9
9. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.A.M.T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evolut. Comput. **6**(2), 182–197 (2002)
10. Droste, S., Wiesmann, D.: Metric based evolutionary algorithms. In: Poli, R., Banzhaf, W., Langdon, W.B., Miller, J., Nordin, P., Fogarty, T.C. (eds.) Genetic Programming. EuroGP 2000. LNCS, vol. 1802, pp. 29–43. Springer, Berlin, Heidelberg (2000). https://doi.org/10.1007/978-3-540-46239-2_3
11. Ezendam, T., Ezendam, T.T., Hofmeyer, H.H., Boonstra, S.S., Pauwels, P.P.: Two geometry conformal methods for the use in a multi-disciplinary non-orthogonal building spatial design optimisation framework. M.Sc.-thesis Eindhoven University of Technology, Department of the Built Environment, Structural Design Group (2021)
12. Fonseca, C.M., Guerreiro, A.P., López-Ibáñez, M., Paquete, L.: On the computation of the empirical attainment function. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) EMO 2011. LNCS, vol. 6576, pp. 106–120. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19893-9_8
13. Grzesiak-Kopeć, K., Strug, B., Ślusarczyk, G.: Specification-driven evolution of floor plan design. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tusar, T. (eds.) Parallel Problem Solving from Nature–PPSN XVII. PPSN 2022. LNCS, vol. 13399, pp. 368–381. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-14721-0_26
14. Hajela, P., Lee, E., Lin, C.Y.: Genetic Algorithms in Structural Topology Optimization. In: Bendsoe, M.P., Soares, C.A.M. (eds.) Topology Design of Structures.

NATO ASI Series, vol. 227, pp. 117–133. Springer, Dordrecht (1993). https://doi.org/10.1007/978-94-011-1804-0_10
15. Igel, C., Kreutz, M.: Operator adaptation in evolutionary computation and its application to structure optimization of neural networks. Neurocomputing **55**(1–2), 347–361 (2003)
16. Li, R., et al.: Mixed integer evolution strategies for parameter optimization. Evolut. Comput. **21**(1), 29–64 (2013)
17. Pereverdieva, K.: Prism-Net Implementation (2022). https://doi.org/10.5281/zenodo.7430052
18. Rudolph, G.: An evolutionary algorithm for integer programming. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) PPSN 1994. LNCS, vol. 866, pp. 139–148. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58484-6_258
19. Schmidt, M., Lipson, H.: Distilling free-form natural laws from experimental data. Science **324**(5923), 81–85 (2009)
20. van der Eelke, H., Kruisselbrink, J., Aleman, A., Emmerich, M.T., Bender, A., IJzerman, A.P.: Evolutionary design of selective adenosine receptor ligands. J. Cheminform. **2**(1), 1 (2010)