Multi-objective mixed-integer evolutionary algorithms for building spatial design

Blom, K. van der

Chapter 7

Mining Optimisation Data for Design Rules

Up to this point, a mixed-integer representation has been defined for the building spatial design problem in Chapter 3. Based on this representation, multi-objective evolutionary algorithms have been devised, along with problem specific operators in Chapters 4 and 5. Furthermore, the application of the hypervolume indicator gradient [40], to improve local search, was studied in Chapter 6, which resulted in a considerable amount of optimisation data.

Despite all this progress, the transfer of an optimisation result to a design expert is not merely a matter of stating "this solution is better than the previous one". For an optimised building spatial design to be used, the solution must be trusted by the design expert. To inspire such confidence in the optimised design, the optimised results should be made explainable. This can be achieved by learning heuristic design rules from the optimisation data. Given such rules, it becomes clear why the design is effective. Ideally, not only known rules that experts trust and understand are obtained, but also new insights. By combining known and new design rules it is possible for experts, and automated (e.g. co-evolutionary [23]) design systems, to improve their design process. These improved design processes can then be applied to similar problems, without another lengthy optimisation procedure. Learning these design rules, new and old, is the focus of RQ4, and as such of this chapter.

The process of learning innovative design rules from optimisation data was introduced in [33], and termed *innovization*. This concept has since been applied to a

variety of problems such as clutch brake design in [33], and truss design in [7]. Later, the learning process was interleaved with the optimisation process in [78], and further automated in [7, 31]. Furthermore, in [8] it was studied how an optimiser learns new concepts over time. Here it is investigated whether simple techniques used to verify optimisation results may also lead to innovative insights.

This work is a first step in applying innovization in building spatial design. The following contributions are made: Optimisation results are verified through data analysis of a subset of the 800 000 solutions found by multi-objective optimisation in Section 6.5. Handling a dataset of this size also results in new challenges. With this in mind, simple and computationally inexpensive analysis techniques are applied.

From here on, this chapter first introduces features to enable the discovery of heuristic design rules in Section 7.1. The preparation of the considered dataset is then described in Section 7.2. Section 7.3 evaluates the results from analysis of the data, and the implications that follow. Finally, Section 7.4 briefly summarises the study, and proposes possible directions for future work.

7.1 Features

The supercube representation introduced in Chapter 3 is a mixed-integer representation of the building spatial design problem, consisting of binary and positive real numbers. Raw data in this format is difficult to interpret in terms of building properties, making it difficult to learn directly from this data. To ease this process, this section introduces elementary features that allow building engineers to characterise a building spatial design. Such features are necessarily domain specific. However, the same process may be applied in other domains.

Given that the supercube representation is key to understanding the dataset and features, its essential components are briefly reintroduced in the following. Since it is used for building spatial design, the supercube representation considers a number of spaces that together form the building spatial design. Each space is defined as a cuboid (3D rectangle), such that the whole building consists of rectangular surfaces, like in Figure 7.1. Additional constraints ensure that the floors of all spaces are connected with the soil via other spaces, that is, in the given representation no floating or overhanging spaces may exist.

All considered features are listed in Table 7.1 with their definitions and explanations. Except for the last three, all other features are computed both for the building, and for individual spaces. Since the ordering of spaces is arbitrary, including values

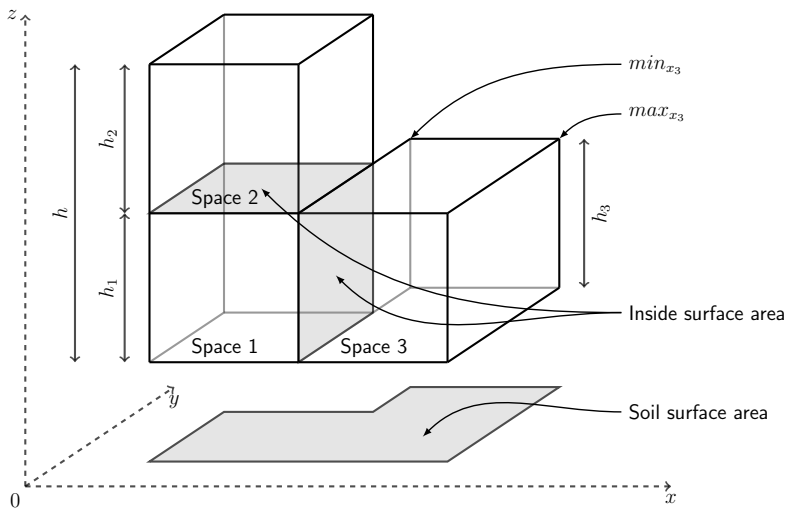


Figure 7.1: Example building spatial design, annotated with a selection of features.

for each of them in the feature set would be of little use. Therefore, statistics are taken over all spaces in a building for each feature. In particular, the minimum, maximum, mean, median, range, standard deviation and Gini index (average deviation from the mean) are considered. Since the last three features in Table 7.1 do not make sense for individual spaces (e.g. mean height of a space is equal to its height), they are only computed for the building as a whole.

Values for w, d, h are found by taking $max_* - min_*$, where $*$ corresponds to x, y, z respectively. In other words, they are simply the distance between the minimal and maximal coordinates of a given dimension. For example, the min_x and max_x of space 3 are marked in Figure 7.1. Note that these values are computed for the full design, as well as for individual spaces, as indicated for height in Figure 7.1 with h for the complete building spatial design, and h_1, h_2, h_3 for each individual space.

To differentiate between various surfaces, the following surface area definitions are used. First, to distinguish between different locations of the surfaces, a non-overlapping division is made between inside (*in_area*), outside (*out_area*), and soil (*soil_area*) surface area. Exterior surfaces are considered as outside, while interior surfaces are considered as inside. The ground floor which connects with the soil is excluded from the outside surface area, and taken as soil surface area. In Figure 7.1, examples of inside and soil surface area are highlighted (the rest is outside surface area). Second, to distinguish between walls and floors/ceilings, a division between

Feature	Definition	Explanation
vol	$w \times d \times h$	Volume of the space, or sum of spaces for the full design
short	$min(w, d)$	Shortest horizontal edge, indicator of span
long	$max(w, d)$	Longest horizontal edge, indicator of span
height	$max_z - min_z$	Height of the space or the full building spatial design
out	$sum(out\_area)$	Outside surface area, indicator of energy flow
in	$sum(in\_area)$	Inside surface area, indicator of energy flow
soil	$sum(soil\_area)$	Soil (ground floor) surface area, indicator of spread
horz	$sum(horz\_area)$	Horizontal surface area, indicator of total wall area
vert	$sum(vert\_area)$	Vertical surface area, indicator of floor and roof area
in_out	$in/(in + out)$	Ratio between inside- and outside surface area
out_vol	out/vol	Ratio between outside surface area and volume
long_short	$long/(long + short)$	Ratio between longest- and shortest horizontal edge
meanh	$sum(h \times roof\_area)/soil$	Mean height of the building
meanh_h	$meanh/height$	Ratio between the mean height and the height
height_soil	$height/soil$	Ratio between the height and the soil area

Table 7.1: Features, definitions, and explanations.

horizontal ($horz\_area$) and vertical ($vert\_area$) surface area is made. The horizontal surface area includes all floors and ceilings (so also the ground floor) while the vertical surface area consists of all walls, regardless of them being interior or exterior. Finally, the $roof\_area$ considered for $meanh$ is a part of the roof area in the building spatial design positioned at equal height.

Note that when considering a building as a whole, each surface is counted only once per considered distinction (e.g. horizontal/vertical). However, on the space level, surfaces are sometimes counted twice. That is, for two neighbouring spaces, both count their connecting surface as being part of, for instance, their horizontal surface areas. As a result, the sum of the surface areas of all spaces is not (necessarily) equal to the total surface area of the building.

In some cases, different features measure the same thing. For instance, the outside surface area of a building has an equal distribution (but not value) to the mean outside surface area of the spaces. Despite this, such features are kept to simplify data processing. In the analysis, only one representative should be used for these equivalent features, unless the differing values provide additional insights.

Additionally, some features may result in distributions similar to each other. This is particularly common for the range, standard deviation, and Gini index. However, even small differences may make one of them more valuable in distinguishing between solution classes than the other. Since, a priori, it is not known which is more useful in which situation, all of them are included.

Finally, it is noted that undefined (NAN) values may appear in a few cases. Some spaces may be disconnected (meaning they do not share a wall with another space). As a result, it can occur in a building design that none of the spaces has a neighbour, from which it follows that their inside surface area is zero. In these cases, the Gini indices of the interior surface area, and of the ratio between inside and outside surface areas will be undefined and marked as NAN (the Gini index divides by the sum of the set of spaces, which is zero in this case). However, since these are very low quality solutions, they are not considered in the analysis in the rest of this chapter. This will become clear in the next section.

7.2 Data Preparation

In order to learn heuristic rules for building spatial design, the dataset from the optimisation experiments in Section 6.5 is used. The dataset is a Pareto front and an archive from a building design optimisation that aimed for a building spatial design consisting of three spaces, with a total volume of $300\,\mathrm{m}^3$ (cubic metre). Note that while these may seem like simple building spatial designs, they already require 9 continuous and 81 binary variables to encode with the supercube representation (Section 3.2), leading to a large search space. The optimisation runs resulted in a dataset of around 800 000 solutions. Here the data is prepared for analysis in the following five steps. First, classes are defined to enable the discovery of different qualities in different groups of solutions. Second, the nondominated (ND) set is identified. Third, the knee point solution is identified. Fourth, solutions are assigned labels to link them to a class, based on the previously identified ND set and knee point. Fifth, a procedure is described to equalise the number of solutions in each class for those analysis techniques that demand this. Note that all steps are defined such that they should at least be generalisable for two-dimensional convex Pareto fronts with a pronounced knee-shape.

To be able to learn from the features defined in the previous section, the data is split into different classes. This is accomplished based on objective values, rather than features. Classification based on objective values allows for the verification of the optimisation procedure: Do design experts agree that the designs with good objective values are indeed good? In addition, it is often a combination of features that indicate a certain quality in the building spatial design, making feature based classification more complex. Further, by classifying on known good qualities of a building spatial design, finding innovative design rules would become very unlikely. Here, four categories of solutions are considered: the knee point area (KP), good in the compliance objective

(F1), good in the heating/cooling energy objective (F2), and relatively low quality solutions (BD). The aim is to data-mine for heuristic design rules that make it possible to differentiate between all of these distinct classes. For problems with more objectives additional classes F* can be added as needed.

The classification considers two primary aspects: (1) It should clearly distinguish between the classes in the objective space, and (2) It should be computationally efficient to enable processing of the large dataset of circa 800 000 points. The computational efficiency should also allow the proposed methods to generalise to larger building spatial designs than those considered here.

Since the considered classes are defined based on the nondominated (ND) set and the knee point, these have to be identified first. For ND set computation the well-known log-linear time algorithm based on sorting is employed [65]. Based on the ND set, the knee point is derived as follows. First the objective values of the ND set are normalised to a $[0, 1]$ range, where outliers beyond 1.5 times the interquartile range are set to the appropriate boundary value. Next, the Euclidean distance to the origin $(0, 0)$ is computed for each normalised ND point. The point with the smallest distance is then taken as the knee point (indicated with 'kp' in Figure 7.2), which is a reasonable approximation for the given dataset.

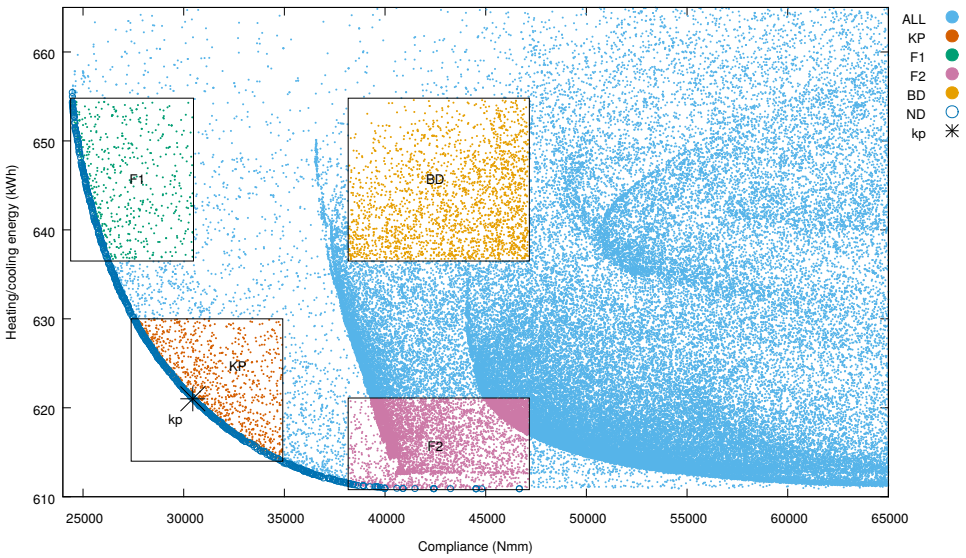


Figure 7.2: Division of data into different classes: All points (ALL), knee point area (KP), objective one (F1), objective two (F2), bad solutions (BD) included in the analysis; relative to the nondominated set (ND), and the knee point (kp). A subset of the full dataset is shown.

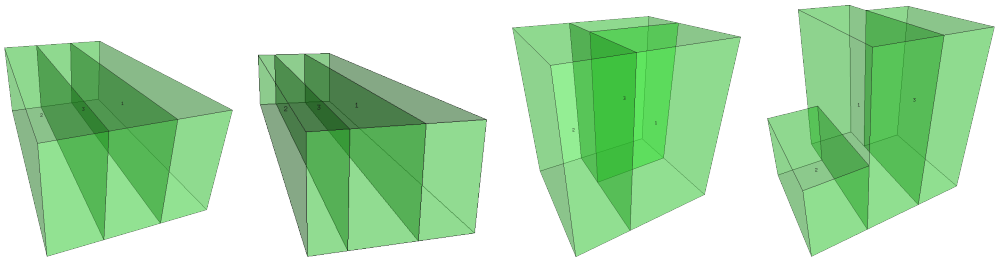
The data is then classified based on the knee point $p = (p_1, p_2)$, and the ND set. For this, the ND set is first reduced to the ND points that were not considered an outlier after normalisation, but the non-normalised values are used. In order to classify in a computationally efficient manner, each class is defined by a bounding box. These bounding boxes are found based on the length of the range of the ND set in objective one r_1, and objective two r_2. For the knee point area class (KP) the lower bound of the box is set to $(0, 0)$, while the upper bound is set to $(p_1 + r_1 \times 0.2, p_2 + r_2 \times 0.2)$. For class F1 a lower bound of $(p_1 + r_1 \times 0.35, 0)$, and an upper bound of $(p_1 + r_1 \times 0.75, p_2)$ are taken. Similarly, F2 is found with the bounds $(0, p_2 + r_2 \times 0.35)$, and $(p_1, p_2 + r_2 \times 0.75)$. Lastly, BD uses the bounds $(p_1 + r_1 \times 0.35, p_2 + r_2 \times 0.35)$, and $(p_1 + r_1 \times 0.75, p_2 + r_2 \times 0.75)$. Following this, points are assigned a label based on the box they are located in. Any remaining unlabelled points are excluded from the analysis. Note that the parameters $0.2, 0.35, 0.75$ are heuristic in their nature. It is possible to change them slightly, but overlap of the regions should be avoided.

The result of the classification process is visualised in Figure 7.2. Note that gaps are left between the different classes to improve the chances of being able to distinguish between them. If the classes would directly neighbour each other, points on the border are likely to have very similar features. This would impede learning what makes a solution perform well (or not) in one objective or the other. Future work could study how these points can be included in the analysis.

In Figure 7.3 a randomly selected example of a building spatial design is shown for each class. Although the examples for KP and F1 look similar, the design for F1 is far more elongated. This result can be expected, as the short spans (here coupled with elongated spaces) allow F1 designs to reduce the strain energy, at the cost of a larger surface area, which reduces thermal efficiency. The F2 design shows the reverse, with a much more compact design. Finally, the BD design is not very evenly arranged in the spatial sense, and shows relatively poor performance in both objectives.

After processing the dataset[1] 70 088 of the 806 430 solutions are labelled, including 5978 KP, 3400 F1, 48 482 F2, and 12 228 BD solutions respectively. Given the mixed-integer nature of the representation, multiple discrete subspaces can be seen in Figure 7.2, indicated by the apparition of different curves in the point cloud. Since the dataset is not homogeneous, the resulting classes do not have an equal number of points. For some types of analysis, however, it is critical to have equally sized classes. In such situations, excess solutions are removed from the larger classes uniformly at random. In all other situations, all labelled data is used.

[1] The dataset is available under `http://moda.liacs.nl/index.php?page=code`



(a) KP. **(b)** F1. **(c)** F2. **(d)** BD.

Figure 7.3: Typical examples of the different classes.

7.3 Results

Two techniques are used for data analysis: Box plots and decision trees. Box plots give insight into the distribution of feature data for different solution classes. As such, it may be possible to identify features that allow for a clear distinction between two or more classes. Besides, the decision tree can provide information about distinguishing features, since it generates clear rules based on such features. Moreover, it gives confidence measures for the classification of solutions to different classes. Finally, by using the learned decision tree on new data, it is possible to validate whether those rules can indeed be used reliably.

7.3.1 Box Plots

To generate box plots all labelled data is used, with each feature normalised to a $[0, 1]$ range, without removing outliers. In the plots, each class is then visualised by an individual box, such that any differences become clearly visible.

In Figure 7.4 a subset of the features is shown that appears to allow for a significant amount of distinction between the different classes. Notice, for example, how the mean of the most extended horizontal edge (long.mean) enables differentiation between objective one (F1), and objective two (F2).

Surprisingly the soil surface area (soil.mean), and the horizontal surface area (not in the figure) both showed exactly the same distributions. This occurs because all buildings considered in the labelled dataset are single-storey buildings. For such single-storey buildings, the horizontal surface area is equal to the soil surface area plus the roof area. Since these two areas are equal, the horizontal surface area is exactly twice the soil surface area, which results in their equal distributions.

It appears then that, in general, single-storey buildings have a good performance

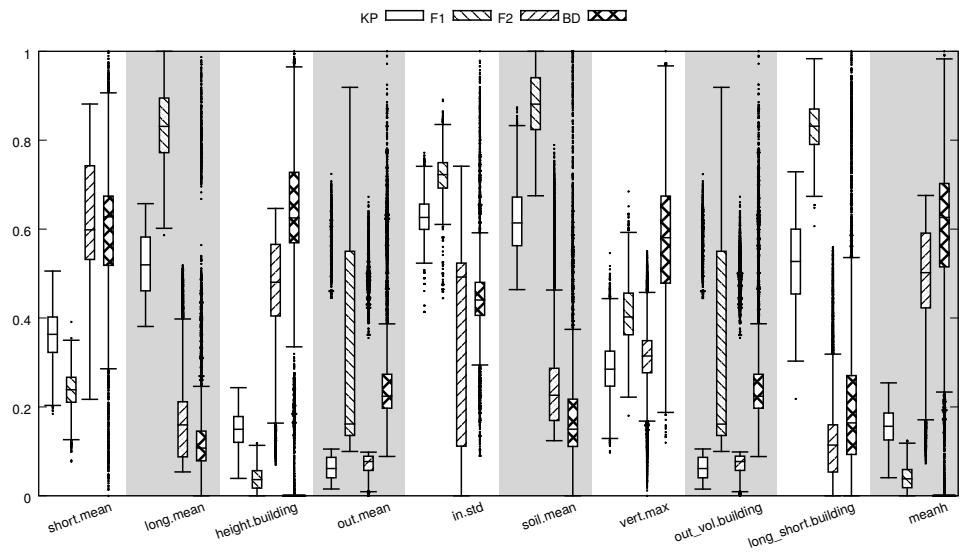


Figure 7.4: Boxplot of a selection of distinguishing features.

for the given objectives, even if they are not necessarily optimal. After all, the labelled solutions are all relatively close the Pareto front approximation. Naturally, this result may not generalise to designs with a larger number of spaces. This also indicates it may be interesting to include an even worse class of solutions in future analysis to see how things differ with even worse solutions. Additionally, a feature indicating the number of storeys of a building spatial design could be useful as well in this case. Even if just to identify this type of situation more easily.

7.3.2 Decision Trees

In order to use decision trees to their full potential, the data should be equally distributed among the classes. As such, this is carried out as described previously (Section 7.2). Since the smallest class contains 3400 solutions, the other classes are reduced to the same number of data points, resulting in a total of 13 600 solutions. This total is split into a training set of 10 200 solutions, and a test set of 3400 solutions by sampling uniformly at random. Note that as a result of random sampling, the representation of each class is not necessarily exactly equal in either of the training and test sets, but still sufficiently close. The training and test sets then consist of approximately 2550, respectively 850 solutions per class. Only labelled solutions are used, no normalisation

is applied, and no outliers of individual features are removed. In the future it may be of interest to do the same study with unlabelled solutions to see if the generated rules generalise.

Given the prepared dataset, the decision tree in Figure 7.5 was generated with the CART algorithm [27] implemented in the R package *rpart* [95] with default settings. In each node probabilities of belonging to each class are given (from left to right: BD, F1, F2, KP), as well as the percentage of the data concerned. From this figure, it can be found that the longest horizontal edge, the outer surface area, the ratio between the longest and shortest horizontal edge, and the ratio between the inner and outer surface area provide important information to distinguish between different classes of solutions.

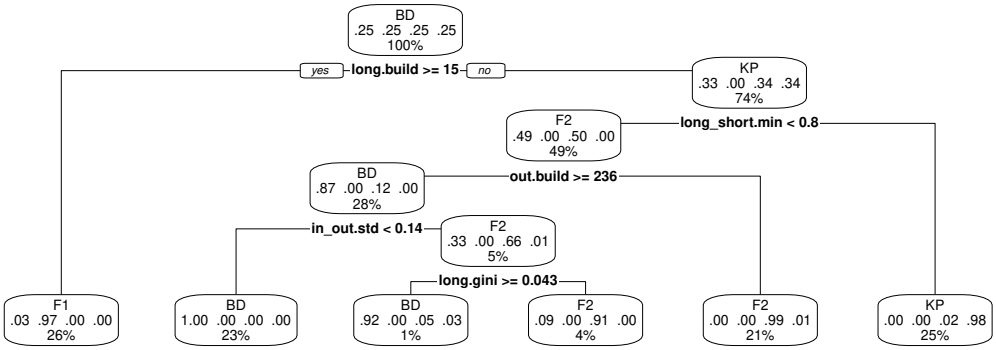


Figure 7.5: Decision tree based on data.

These rules indicate properties of a building that contribute to qualities present in different solution classes. The first split shows that relatively long buildings (long.build) are likely to be efficient in objective one (compliance). This split intuitively makes sense, since buildings that are more stretched out are likely to have short spans. Note that this is under the assumption that not just the building is stretched out, but the spaces as well (e.g. F1 in Figure 7.3).

In the other branch buildings are a bit more compact. Additionally, it can be seen that buildings where the minimal ratio of the spaces between the longest and shortest horizontal edge (long_short.min) is relatively high, are very likely to be solutions in the knee point area. This indicates that although the building as a whole is more compact, the individual spaces remain somewhat elongated to balance between the two objectives.

The primary split between low quality solutions and the second objective (energy)

is made based on the outer surface area of the entire building (out.build). Since a larger outer surface area is an indicator of a more significant loss of energy to the outside, this appears to be a sensible rule. Further, these rules provide clear pointers on how to navigate towards the PF. It may be possible to incorporate this in problem specific operators to speed up the optimisation process.

From the decision tree in Figure 7.5 it appears classification of solutions is possible with high precision. To validate this, the tree was used to classify the 3400 solutions in the test set. Table 7.2 shows the resulting predictions. All assignments were made with a confidence of at least 90 %, showing that it is possible to classify designs quite reliably. A particularly notable result is the classification of the majority of the solutions in the F2 and KP classes, which, for this dataset, is done with near perfect confidence. Not only does this provide confidence in the optimisation process, but these rules could even be useful during optimisation. By classifying new solutions based on these rules it may be possible to identify which solutions are more likely to perform well, such that expensive simulations might only be needed for those.

Prob.	.0000	.0008	.0046	.0048	.0051	.0162	.0276	.0345	.0483	.0926	.9074	.9241	.9655	.9784	.9949	.9952
BD	1758	0	0	0	728	0	54	0	0	0	0	0	0	860	0	0
F1	1649	860	0	0	0	0	0	0	0	0	0	0	891	0	0	0
F2	891	0	0	748	0	860	0	0	54	0	119	0	0	0	728	0
KP	728	0	860	0	0	0	0	891	0	119	0	54	0	0	0	748

Table 7.2: Decision tree results on the test set. Columns relate to the predicted probability of belonging to a specific class, whereas rows refer to classes. Each cell then contains the number of solutions that belong to a solution class, with a particular probability.

Based on first discussions with a design expert, it can be concluded that interesting heuristics are learned that accurately describe high quality building spatial designs. However, it seems to remain difficult to foresee the consequences of changes in feature values with respect to the objective values. In order to improve this, visual aids would be helpful. For instance, a slider controlling the weights of the structural and thermal objectives could be used to change the spatial design in real-time.

7.4 Conclusion

7.4.1 Summary

In this chapter optimisation data from previous experiments (Section 6.5) has been analysed to learn what makes a good building spatial design perform well with respect

to compliance and energy performance. This information could then be used to inform a design expert why a proposed new design should be considered. Moreover, if new design rules are learned, the expert can use them in the future.

To be able to analyse the optimisation data, this chapter included a process to go from optimisation data to practically analysable data. To this end, features have been defined that describe a building spatial design in a meaningful way for a design expert. Following that, the data was subdivided in multiple classes, representing solutions that are good in objective one (structural performance), objective two (energy performance), both objectives, or neither objective.

Data analysis has been performed through the use of box plots and decision trees. The box plots provided a clear overview of which features are likely to be useful in differentiating between the classes. Then, the decision tree produces specific rules to assign solutions to one of the previously defined classes. Further, these rules have been tested with new solutions (not used to produce the decision tree), which resulted in high precision ($\geq 96\%$) classification of solutions. Finally, from discussions with a design expert it was identified that the learned rules accurately describe what constitutes a good building spatial design.

With respect to RQ4 it can be said that it is indeed possible to learn valuable information from optimisation data, or to confirm existing empirical knowledge. Clearly, the optimised designs have been proven to conform with rules known by design experts, learned from the data.

7.4.2 Future Work

Besides generating insight, the design rules could also be useful in steering the multi-objective optimisation process. For future work, it would be interesting to investigate which moves in the optimisation process result in improvements. In other words, given an existing design, what changes to its features will, with high probability, result in an improved design. Furthermore, it may be possible to apply learned rules in co-evolutionary design processes [23]. Or, one could use the results from data mining to implement a mechanism to discard inferior solutions without the to use expensive simulations during optimisation.

The current work analyses data for a specific type of building. To generalise the conclusions, the same methods should be evaluated on a larger variety of building types. Given the computationally efficient nature of the used approach, it is probable that larger building spatial designs can be handled, however this must still be verified.

Additionally, currently only a subset of the optimisation data is labelled. As a result, it is unclear whether the learned rules generally allow the identification of, for instance, solutions that perform well in the compliance objective. It may be the case that some areas of the objective space, that have not been considered here, have similar characteristics in some features. This should be studied in the future. A challenge is how to do proper analysis with both sparse and dense areas in the objective space.

In the same line of including the currently unlabelled solutions into the analysis, it would be interesting to consider how the inclusion of solutions that lay between the currently considered classes would influence the ability to learn useful rules. Moreover, including solutions beyond the current worst class may provide new insights in how solutions change as they come closer to the Pareto front.

To improve the usefulness of the learned design rules for a design expert, it would be beneficial to have them more involved in the process. As such, an option might be introduced that allows them to use sliders to get a feel for the relation between different feature and objective values. The development of effective tools for these ideas is a challenging future direction towards a user-friendly workflow from problem specification to results and insights.