# The constraints satisfaction problem approach in the design of an architectural functional layout

Machi Zawidzki[1], Kazuyoshi Tateyama[1], Ikuko Nishikawa[2]

*[1]College of Science and Engineering, [2]College of Information Science and Engineering, Ritsumeikan University, Kusatsu, Japan*

Noji-Higashi 1-1-1, Kusatsu, Shiga, 525-8577 Japan, Phone: +81 080 6168 6660

*(Received xxxx; final version received xxxx)*

A design support system with a new strategy for finding the optimal functional configurations of rooms for architectural layouts is presented. A set of configurations satisfying given constraints is generated and ranked according to multiple objectives. Method can be applied to problems in architectural practice, urban or graphic design - wherever allocation of related geometrical elements of known shape is optimized. Although the methodology is shown using simplified examples- a single story residential building with two apartments each having two rooms- the results resemble realistic functional layouts. One example of a practical size problem of a layout of 3 apartments of total 20 rooms is demonstrated, where generated solution can be used as a base for a realistic architectural blueprint. The discretisation of design space is discussed, followed by application of backtrack search algorithm used for generating a set of potentially "good" room configurations. Next the solutions are classified by a machine learning method (FFN) as "proper" or "improper" according to the internal communication criteria. Examples of interactive ranking of the "proper" configurations according to multiple criteria and choosing "the best" ones are presented. The proposed framework is general and universal – the criteria, parameters and weights can be individually defined by a user and the search algorithm can be adjusted to a specific problem.

Keywords: architecture, design support, architectural layout optimization; multi-objective, discrete optimization, CSP

## 1. Introduction

### *1.1. What is architecture?*

According to some estimates, there are approximately two thousands definitions of the term "architecture" (Leśniakowska 1996). This alone shows that in this field there are some major difficulties with nomenclature and formalizations due to the enormous expressive power of the natural language, that can easily generate contradictions and paradoxes (Sowa 2008). Therefore the definitions used onward are somewhat arbitrary.

The language of architecture is in great majority visual- the most effective way of communication are diagrams, plans, perspective drawings, elevation drawings etc.

An architectural design is usually not a linear procedure, but a sequence of gradual transitions using functional sketches, sketches of the intended appearance, functional layouts, detailed drafting and approval process to the issue of the drawings and documents. There are usually many feedbacks and reconsiderations before the final blueprints are produced, and there is rarely common satisfaction with the results. This situation may indicate any combination of the following possibilities:

(1) In a general sense - the problems that architectural design deals with are difficult and the known heuristics are not sufficient.
(2) The problems are not addressed properly.
(3) Whether it is true or not, considering the architectural design as a multi objective optimization problem, an "ideal" solution was not found. The ideal solution in this sense is the point in the objective space where coordinates have maximal (minimal) values of the objective functions. Often such a solution does not exist. In such a case it is necessary to make a choice among the permissible solutions which then becomes the so-called admissible solution. In practice, this selection is another difficult problem, so the final decision is most often made arbitrarily.

## *1.2. What is a functional layout?*

The following definition of a functional layout is assumed:

*A functional layout is a graphic representation of relationships among spaces of a building. Although the level of accuracy greatly varies, the sizes and dimensions of spaces (for example rooms) need to be represented with proper proportions, so the elements of a functional layout roughly correspond to the final architectural plan.*

In most cases, the success of the final architectural design strongly depends on the proper solution of functional allocation of spaces considering their approximate sizes and relationships. Experience has shown also that the most effective strategy for the layout problem is the iterative approach (Liggett 2000).

As mentioned above, although the architectural design is a complex process that in general cannot be linearised, it can be divided into independent stages (figure 1).

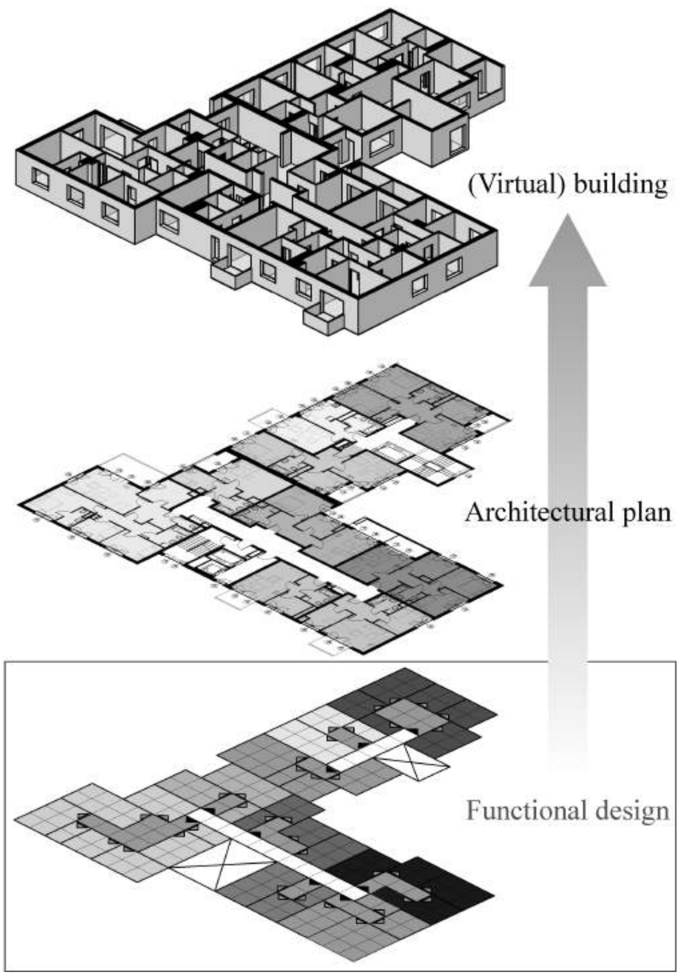


Figure 1. A scheme showing the stages of an architectural design. This paper focuses on the problems of the functional design.

In the past the problem of creating an optimal functional layout was approached by computer science methods. The resulting schemes - although correct - lacked certain organizational, aesthetic, or identifiable characteristics (Terzidis 2006). The results of the method presented, although simplified, have topologies and geometries of realistic architectural designs. The design of the functional layout of the spaces in the architectural design of a building and industrial facilities has received a lot of attention in the fields of Architecture and Engineering. However, research results from those fields are seldom used in the other (Del Río-Cidoncha 2007).

Two types of constraints limit the number of possible solutions: project requirements and external limitations. The project requirements provide information regarding the sizes and relationships among the spaces. As an example, consider a simple apartment with a master bedroom, second bedroom, living room, kitchen, bathroom and WC. Additional requirements are for the kitchen to be adjacent to the living room, the bathroom to be adjacent to the master bedroom, the living room to face south and none of the bedrooms to face north. The latter conditions are an extension of the traditional adjacency diagram often used in the architectural practice (Mitchell 1990). The newly introduced elements are orientation and negative edges (forbidden adjacencies). These are the requirements that must be met.

The external limitations are certain additional geometrical constraints due to plot size and shape, building code requirements, daylight conditions and, structural system requirements. These are the constraints that must not be violated and in the implemented algorithm they translated to so called “pruning functions”.

Although the presented framework can be applied to a variety of “layout” problems, further in the text the application is addressed specifically to single-story residential buildings.

If the geometry of the layout is given and only the facility arrangement is to be optimized, the problem can be reduced to so called “facility layout optimization”. Heuristic methods such as genetic algorithms (Tam 1992), simulated annealing (Sharpe 1986), “annealed neural network” (Yeh 2006) and tabu search strategies (including multi-searching tabu search strategy) (Liang 2008) have been successfully implemented for this kind of optimization.

However, in architectural practice such problems are in fact rare and almost always the purpose of the architectural layout optimization is to establish the layout geometry under several constraints.

## 2. The discretisation of the space: a unit grid.

Although theoretically rooms in a residential building can have seemingly any sizes (Michalek 2002), in fact almost always their sizes and proportions lie within a certain, surprisingly narrow range. For example the room proportions almost always lie within 1-2 squares. Such practical knowledge can greatly reduce the search space and exclude a vast amount of practically unacceptable solutions. The use of various grid systems has a long history in all sorts of design: architectural, urban, engineering, graphic, etc. Specifically in the field of layout optimization, grid is used in the quadratic assignment problem where the shape of a building is given on a coarse grid and space units are assigned to their corresponding locations while the space elements have relationships among each other. The relationships include topology and geometry and distinguish space layout planning from the classical linear assignment problem (Jo and Gero 1998). However, in the architectural functional layout problem usually additional constraints are given for the shapes of the spaces and the lot. Moreover, usually the outline of the building is not *a priori* given and designing the shape is the essence of architect's efforts. The studies of existing architectural layouts, specifically of multi-unit residential buildings lead to the following assumptions:

*Assumption 1: The functional layouts problems can be solved in a discrete space.*

All of the investigated cases of floor plans of residential buildings could be schematically transcribed on to a very coarse square grid maintaining exactly all the relationships among spaces and approximately the room sizes and proportions. Fine-grid architectural plans, where the grid is usually of 1 cm, can be discretised into a much coarser grid maintaining the topological relationships as shown in example in figure 2. The simplified graphical representation in the top right corner is a functional layout as defined above. The final architectural plan is far more complicated since it is subject to a number of additional constraints and requirements such as the construction system limiting the spans of the slabs, the placement of the load bearing members and the provision for services such as plumbing and HVAC. The type of the coarse grid, the sizes and proportions of spaces on the grid must be always determined according to the expert knowledge about certain type of building.

*Assumption 2: Any functional layout can be transformed into the architectural plan.*

In principle an architectural plan can be derived from a functional layout. There is a possibility that some functional layouts cannot be transformed into the architectural blueprints due to structural conflicts, but then some adjustments to the initial constraints in the algorithm must be made and process reiterated. All the dimensions of the functional layout, that uses a coarse single-unit grid must be interpreted and adjusted by a designer in order to become an architectural "blueprint".

*Assumption 3: For a given type of functional layout, the number of size variations of the given rooms is small.*

Although in architectural blueprints, the sizes of rooms on a 1 cm fine-grid vary greatly, after discretisation into the functional grid (which uses an arbitrary unit, in this case it is the minimum clearance for a corridor, that is between 1 and 1.5 m), the number of possible sizes and proportions of rooms decreases considerably (figure 2).

*Assumption 4: A combinatorial search is feasible in a functional design space.*

All these observations lead to an assumption that by providing sufficient initial information regarding the room configuration and by applying an appropriate coarse gird system, the number of solutions decreases enough to apply a combinatorial search method.

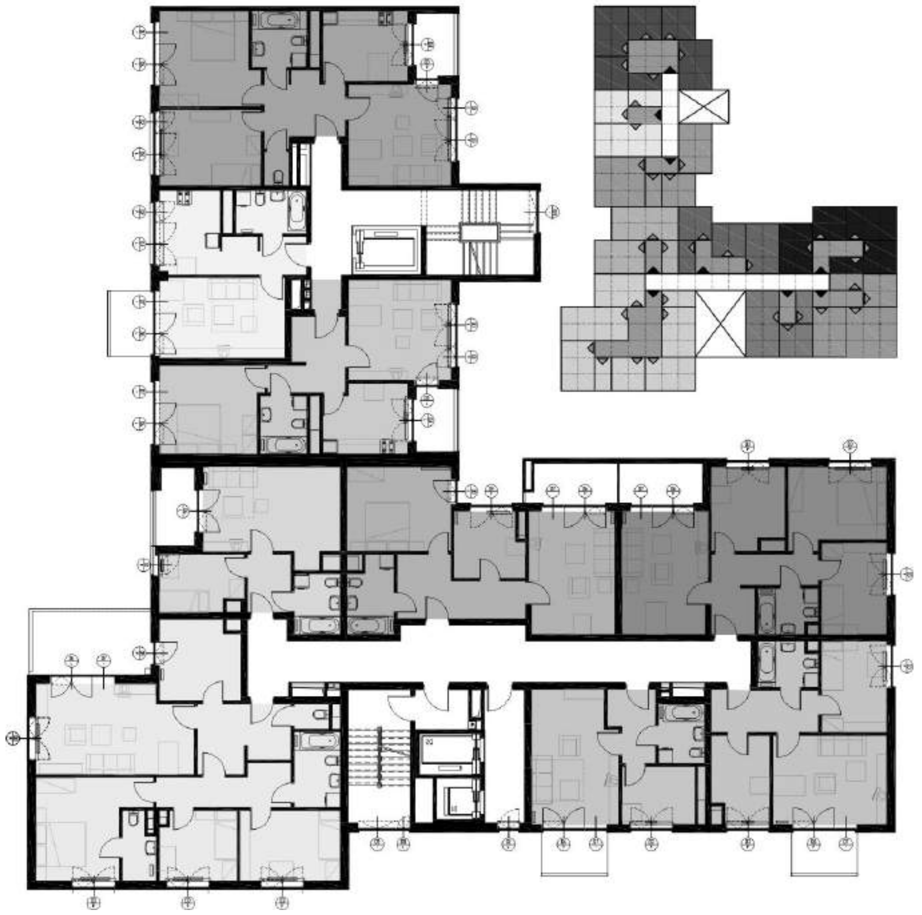


Figure 2. An example of a blueprint of one floor of a real residential building with nine apartments. In the top right corner the corresponding functional layout is shown.

Assigning facilities to given shapes on grid was first formulated by Armour and Buffa in 1963 as a "quadratic assignment problem" (QAP). They work resulted in a computer program CRAFT (Computerized Relative Allocation of Facilities Technique) (Buffa and Armour 1964) and later on, implementations of heuristic methods such as genetic algorithm were also successfully demonstrated (Liggett 2000, Jo and Gero 1998). However, space planning problems are generally more complex than the quadratic assignment formulation due to the imposition of activity area requirements. Since areas required by activities are not necessarily equal, it is not always feasible to match activities and locations on a one-to-one basis (Liggett 2000). In this paper, unlike the classic QAP, all rooms are rectangular, have different predefined sizes and the outline of the building is not given. Because of these conditions the use of genetic algorithm could not be implemented and seems very difficult if possible at all. However application of genetic algorithm for optimizing the topology of an architectural layout defined as finding of the best adjacencies between functional spaces among many possible ones under given constraints was demonstrated (Wong and Chan 2009).

## 3. The procedure

The procedure of ranking the room configurations according to the given criteria is a multi-step process:

(1) The input of all possible information including the desired geometrical properties of the rooms (number, sizes), daylight requirements, relationships between rooms (adjacencies, minimum and maximum distances between the given rooms ) and the geometrical constraints (sizes) for the apartments.

(2) The generation of the configurations (potential solutions). For this constraint satisfaction problem (CSP) a depth-first backtrack search algorithm is applied. Only complete configurations are valid so that every configuration must contain all the rooms and none of the constraints can be violated. The algorithm is very sensitive to the initial data. Therefore although the search is significantly constrained, the cordiality of the generated set of configurations sometimes is very large and sometimes is null. The procedure can be stopped after collecting any number of potential solutions, but ideally a full search should be performed in order to find the best one. From designer's perspective it is also desirable to have a choice among a number of solutions instead of getting a single mathematically optimal layout (Yeh 2009).

(3) Selection of the proper configurations from the potential solutions. Not all the initially generated configurations (potential solutions) are proper in the architectural sense. The classification criteria are explained further in text. Although the selection can be always made by a designer, in this paper a machine learning method was applied for automation of this classification problem.

(4) Final ranking of the proper functional layouts. Additional criteria can be applied for sorting of the solutions, for example: smallest perimeter of the whole layout, the size of the internal communication, a selected room being in a certain zone (for example at the north facade). Some of these criteria could be included in the initial input for the configuration generation, but in design practice it is more natural to consider a wider range of possible solutions and filter them out

according to additional criteria which are often variable. At this stage other kinds of knowledge can be used for ranking of the solutions. It is also possible that there are no solutions that meet both old and new requirements. However since it is an iterative process, adjustments to the initial input can be made and the whole process restarted.

## *3.1 Human knowledge and creativity*

The most natural way of expressing the knowledge about design requirements are fuzzy terms and although algebraic operations can be performed on fuzzy numbers after defuzzification process (Kosiński and Weigl 1998), quantification of the knowledge by assigning numerical values to the satisfaction levels is still and probably will remain a major problem.

Although the translation of knowledge into the algorithm is a separate problem which is not discussed in this paper, formalization of the architectural layout optimization as discrete combinatorial optimization problem required expert knowledge from the field of architecture. The embedded in the algorithm knowledge on residential building contains setting the type of and size of grid and rooms' sizes and proportions on the grid. Moreover the architectural knowledge is also included in the way how the rooms are positioned in the process of generating the layout solutions- reflecting daylight and communication requirements from the building code. It also possible to adjust the algorithm to various kinds of expert knowledge regarding issues like energy conservation, passive solar and daylighting, structural grid, HVAC, plumbing, security etc. by formulating geometrical constraints. Although the presented procedure can be applied to a variety of layouts, the algorithm should be always adjusted for a specific  problem, for example: residential, swimming pool, commercial, office building etc. The paper focuses on demonstrating that this design support system can be used by an architect for realistic tasks and gives concrete results for specific problems.

It should not be forgotten that designing is an creative act and the still remains a special human activity (Saridakis and Dentsoras 2008). Moreover in most times it is a pleasurable activity for the designer (Cross 1999). There are probably instances where a performance of a room configuration cannot be enumerated, or when a number of different room configurations will perform equally according to the given criteria. However the proposed framework is a practical tool that gives the designer a choice of solutions which meet designer's preferences and constraints which can be explicitly addressed and enumerated. Authors do not deny the creative role of a designer, therefore do not claim that the proposed algorithm will handle all the layout problems by its own. The successful implementation of a tool does not mean replacing the user, but giving one new possibilities.

### *3.2 The initial input data*

The initial data, provided by the user contain the information about the number and the sizes of the rooms and apartments. The information about a single room is a two-level list. The first sublist is the general and constant description of the room- the apartment index, the room index, the room index to which it is adjacent and the daylight requirement. The second sublist contains the information about the dimensions of the room (which may be exchanged by rotation of the room as explained further in text):

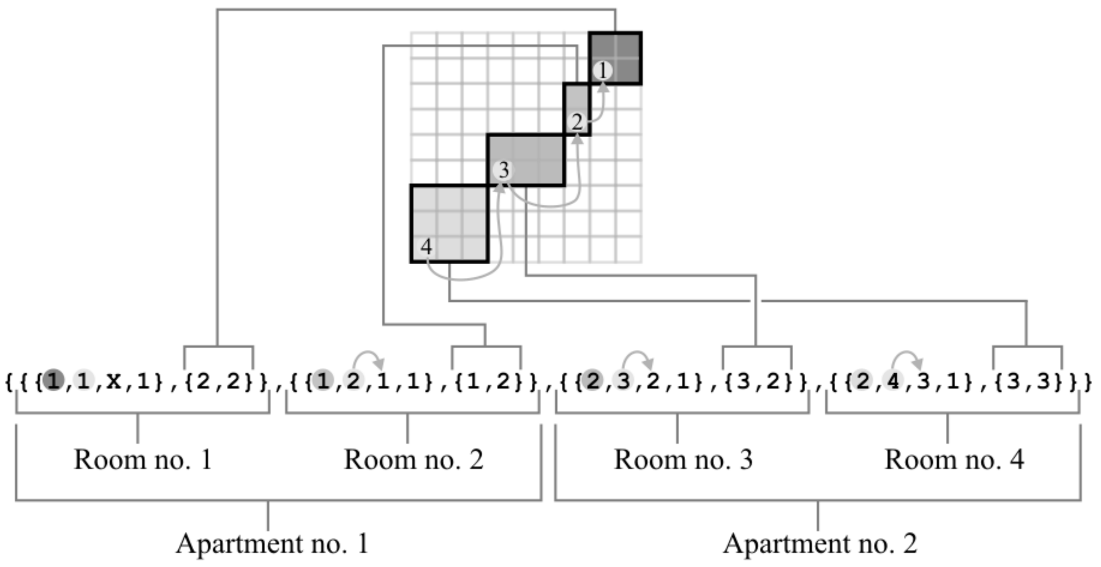
$$\{\{a_i, r_i, r_i^*, d_i\},\{w_i, l_i\}\}, \text{ where}$$

$a_i = 1,...,N_A$
$r_i = 1,...,N_{Ri}$
$r_i^* = 1,...,N_r -1$
$d_i = 0, 1$
$w_i$ – the width ($x$ dimension) of an $i^{th}$ room
$l_i$ – the length ($y$ dimension) of an $i^{th}$ room

$N_A$ – the total number of apartments
$N_{Ri}$ – the number of rooms in the $i^{th}$ apartment

In the process of the layout optimization the value of the second sublist may be reversed due to rotation of the room: $\{w_i, l_i\} \rightarrow \{l_i, w_i\}$.

A layout of an apartment or a whole building is uniquely represented as a list of such rooms. An example of a layout with two apartments each having two rooms is shown in figure 3.



```
{{{1,1,X,1},{2,2}},{{1,2,1,1},{1,2}},{{2,3,2,1},{3,2}},{{2,4,3,1},{3,3}}}
```

Figure 3. Input information for a simple layout with 1st apartment having rooms no. 1 and 2 and the 2nd apartment having rooms 3 and 4. The initial values contain information about the adjacency (indicated by arrows) and room sizes.

### *3.3. Further preparation of the input data: the rooms permutations*

The room configuration generating algorithm operates on a fixed list of rooms. The user gives more general information about rooms: sizes and proportions, therefore the algorithm must consider every permutation of the room sequence. The sequence of rooms' shapes within each apartment is permuted and combined with the other apartment as shown in figure 4.

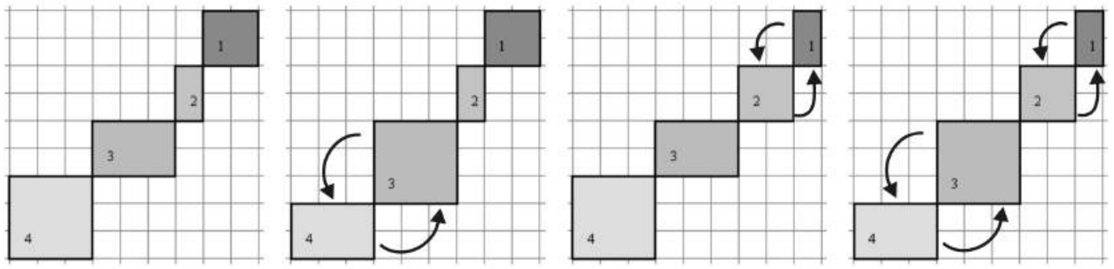


Figure 4. All permutations of the room shapes for the given initial data (the layout from the figure 3).

The total number of permutations in the apartment sequence can be expressed in the following way:

$$\frac{N_A!}{2}\prod_{i=1}^{N_A}(N_{Ri}!) \qquad (1)$$

where
$N_A$ – the total number of apartments, $N_A > 1$
$N_{Ri}$ – the number of rooms in the $i^{th}$ apartment.

The algorithm automatically rotates rectangular rooms in order to fit them in available space.

### *3.4. Finding potential solutions: the generation of the rooms configurations as the constraint satisfaction problem (CSP)*

Since the placement of the rooms is constrained to allowed locations, the design of a functional layout can be considered as a constraint satisfaction problem (CSP). For the layout optimization defined as partitioning of a rectangle a number of programs based on CSP have been implemented in the past: LOOS/ABLOOS (Flemming *et al*. 1992), SEED (Flemming *et al*. 1994), HeGeL (Akin *et al*. 1992) and WRIGHT (Baykan 1991).

Although the housing styles vary greatly, there are always certain constraints for room locations. The reasons why formalization of these constraints is usually the most challenging task are:

(1) At the communication level, transcribing expressed in natural language the expert knowledge of an architect and the client's requirements into numbers and

implementing it into the algorithm is often difficult, perhaps sometimes even impossible.

(2) Since the number of generated solutions is sensitive to the given initial constraints, setting the constraints too "loosely" results in astronomical numbers, while too "tightly" leads to no solutions at all.

(3) Setting the initial constraints too rigorously leads to omitting potentially very interesting and exceptional solutions which were not preconceived by the designer.

(4) Some of the constraints, at least to some degree are contradictory.

The architectural knowledge about the design of most residential buildings can be expressed in the following principles and became geometrical constraints for the rooms configurations, which have been implemented in the search algorithm:

- Two rooms are adjacent if they share at least one unit segment of the wall so a door can be placed in order to connect these rooms. Therefore a corner-to-corner neighbourhood is not considered as adjacency.
- The *main* rooms require daylight, therefore are located on the perimeter of the building (for example: living room, bedroom, study room). In this sense a building with atria has more than one perimeter.
- The *auxiliary* rooms such as bathrooms, toilets and closets can be located inside the building, that is without access to the daylight.
- The main rooms of all the apartments constitute a *chain* that is, one main room is adjacent to another *main* room and so forth. In this project gaps in the chain are not allowed (except between the very first and last rooms - to ensure that the main entrance can be realized).
- The size of the building is limited (and given as an input).
- All the rooms are rectangular. Although it is a major limitation for the architectural creativity, in fact such a statement is true for the great majority of the building stock. However, an approach for solving layout problems not limited to the orthogonal space was also investigated (Bier *et al*. 2008, Damski and Gero 1997).
- The sizes of all rooms are required as an input.
- The sizes of all apartments are limited and given as an input.
- Any distance constraints for any two rooms can be given as an input.
- The corridors are not predefined since their shape is usually not important. Nevertheless, they are crucial for the layout but usually should be as small as possible.

Further constraints for the algorithm such as lot, apartments and room sizes, relative location of the rooms, preferences in certain room locations etc. are given by the user. Most of the knowledge is formalized into the logic statements in the pruning functions which impose geometrical constraints of the placement of rooms in the layout generation process. Although fuzzy logic seems more natural for expressing spatial (architectural) relationships (Changa 2009), since the search space is large, for computational efficiency the bivalent logic was used in the algorithm. Most of the formalized architectural knowledge is specific to the type of building to be optimized.

### *3.5. The depth-first backtracking*

The number of all possible room configurations grows very fast with the number and sizes of the rooms. The number of all positions of rooms on a grid so that the sequence is given and the next room always adheres to the previous one (like in a chain as shown in figure 3) can be calculated with the formula below:

$$2^{k-1}\prod_{i=1}^{k-1}\left(x_i + x_{i+1} + y_i + y_{i+1}\right) \tag{2}$$

where $k$ = number of rooms, $i$= room index
$x_i$ = x-dimension of an $i^{th}$ room; $y_i$ = y-dimension of an $i^{th}$ room

Although corner-to-corner adjacency and overlapping of rooms are not accepted for a room layout, in this calculation they are also included. Moreover, every rectangular room can have two orientations as shown in figure 5.

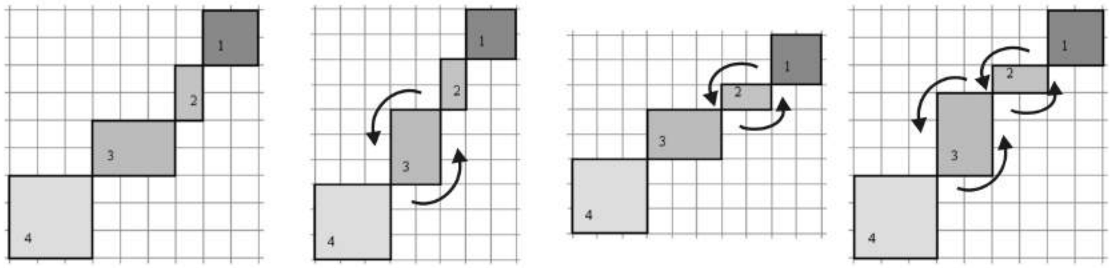


Figure 5. Each rectangular room has two possible orientations.

Therefore the number of possible configurations for a given sequence of rooms should be multiplied by $2^j$ where $j$ is the number of all rectangular rooms in the layout. For this extremely simple example the final number of all possible geometrical arrangements is:

$$\left(\frac{N_A!}{2}\prod_{i=1}^{N_A}(N_{Ri}!)\right)\left(2^{k-1}\prod_{i=1}^{k-1}\left(x_i+x_{i+1}+y_i+y_{i+1}\right)\right)2^j$$

$$\frac{2!}{2}2!\times 2!\times 2^3\times(2+1+2+2)(1+3+2+2)(3+3+2+3)\times 2^2=78848$$

This shows clearly that brute force search methods for problems of realistic size in this method is not rational. Although the search tree grows geometrically in width, the depth of the tree is fixed and defined by the number of rooms. Since the width of the search tree is much larger than the depth and only a room configuration containing all the rooms from the given list is valid- a depth-first backtracking search method was implemented. The algorithm adds the rooms from the input list one after another until all the rooms are placed successfully on a grid as shown in figure 6.

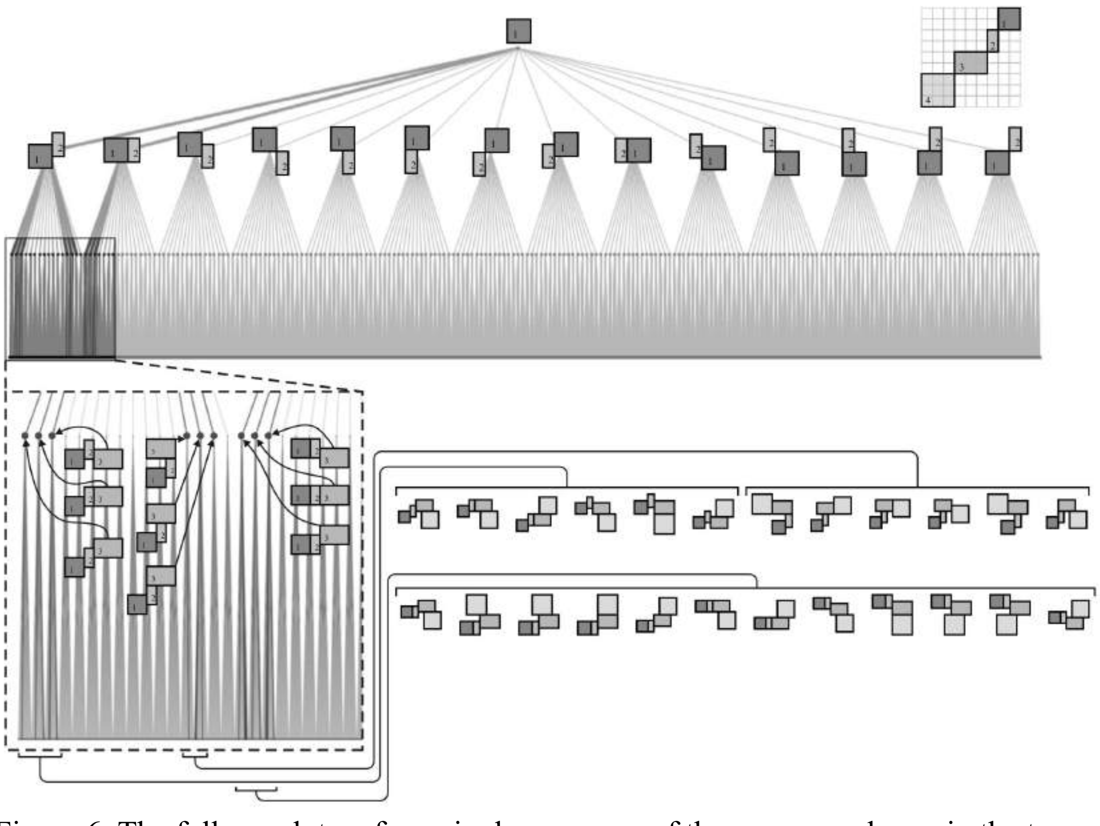


Figure 6. The full search tree for a single sequence of the rooms as shown in the top right corner has 4928 leaves. Dark grey line indicates paths to the 24 potential solutions which are also drawn in the bottom and from which the proper solutions are selected. On the left- magnified part of the graph with related room configurations. Evidently the depth-first search is the best strategy and in such approach the great majority of the nodes are not visited. It also shows that although it greatly depends on the constraints- the good configurations are in fact very rare and localized.

Backtracking was already implemented in the rectangle partitioning for the unequal area layout problem in HeGel (Akin *et al*. 1992) and WRIGHT (Baykan 1991) programs. In this project backtracking is implemented in the following way: if a room cannot be added due to the violation of any constraint or pruning function, this branch of the search is abandoned, the position of the previous room is altered (translation or in case of rectangular room- rotation) and the procedure continues.

The algorithm takes the list of initial input values as in figure 3, and additional parameters further constraining the search as shown and described in table 1.

Table 1. The complete initial information about a layout for the algorithm.

| A sublist of the initial data | Description |
|---|---|
| **`[{{1,1,X,1},{3,3}},{{1,2,1,1},{3,2}}, {{2,3,2,1},{1,2}},{{2,4,3,1},{2,2}}},`** | Information about all the rooms including the daylight accessibility and relationships to other rooms as in figure 2. |
| **`{7,6},`** | The lot size: 7 x 6 units |
| **`{{1,{6,5}},{2,{3,3}}},`** | The apartments' sizes:<br>Apt no. 1: 6 x 5 units<br>Apt no. 2: 3 x 3 units |
| `{{3,1,1,2},{4,1,1,2},{4,2,1,3}}]` | Additional three distance constraints: rm 3 in relation to rm 1 must not be closer than 1 unit and not farther than 2 units and so on. |

The solutions are generated in the form of list containing information for each room. Every room is uniquely encoded in a list of three sublists. The first one is the general description of the room provided as an input data- the apartment index, the room index, the room index to which it is adjacent and the daylight requirement. The second sublist gives the *x* and *y* dimension of the room which in case of rotation of a rectangular room will the reverse of the original input data. The third sublist are the coordinates of the room's bottom left corner as shown and explained in figure 7.

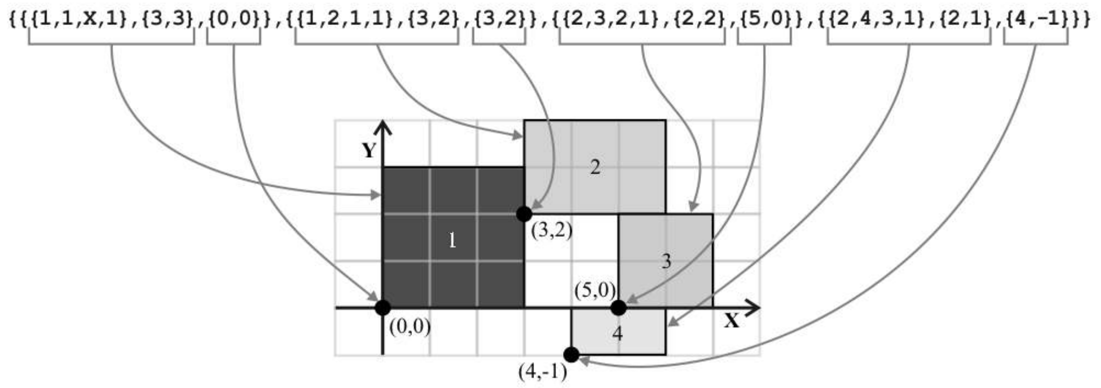

{{{1,1,X,1},{3,3},{0,0}},{{1,2,1,1},{3,2},{3,2}},{{2,3,2,1},{2,2},{5,0}},{{2,4,3,1},{2,1},{4,-1}}}
Y
X
1
2
3
4
(3,2)
(5,0)
(0,0)
(4,-1)

Figure 7. An example of a solution for a small layout and its graphic representation. The positions of the rooms are given by the coordinates of its bottom left corner.

The additional parameters such as plot size and the apartments' sizes further limit the search space. Adjusting them takes some trials and practice. In this example, if the constraints are limited only to the sizes of the whole layout and apartments, the number of generated configurations is very large and most of them have inefficient communication manifested as clustering of rooms as shown in figure 8.

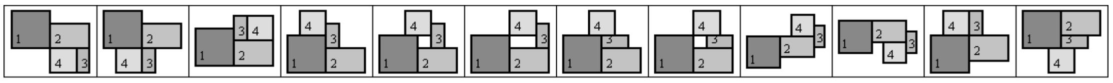


Figure 8. Uncontrolled distances among rooms lead to the generation of many useless configurations. None the examples shown can be used as a proper functional layout.

The communication of the layout means creation of sub-corridors to each room within the apartments and main corridor connecting all the sub-corridors. If it is impossible to create the communication system for the given layout or it becomes to large and/or complex- it is considered inefficient.

By further constraining of rooms' distances, for example- rooms 3 and 4 should always leave at least one unit of clearance from rooms 1 and 2, results in generation of much more useful configurations (figure 9).

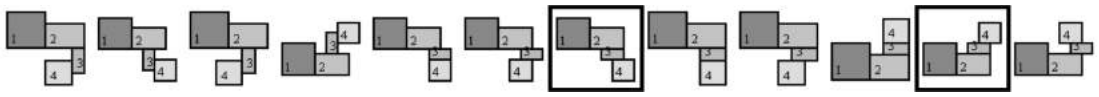


Figure 9. After further constraining of some distances, the generated configurations look much more useful. Configurations 7 and 11 are proper according to the classification based on the corridor creation (further in text).

In this example there are 695 unique solutions generated. It is important that the algorithm does not produce redundant configurations, that is any two solutions in which one - after rotation by π - is identical to the other one are considered not unique. Some examples of the generated configurations are shown in figure 10.

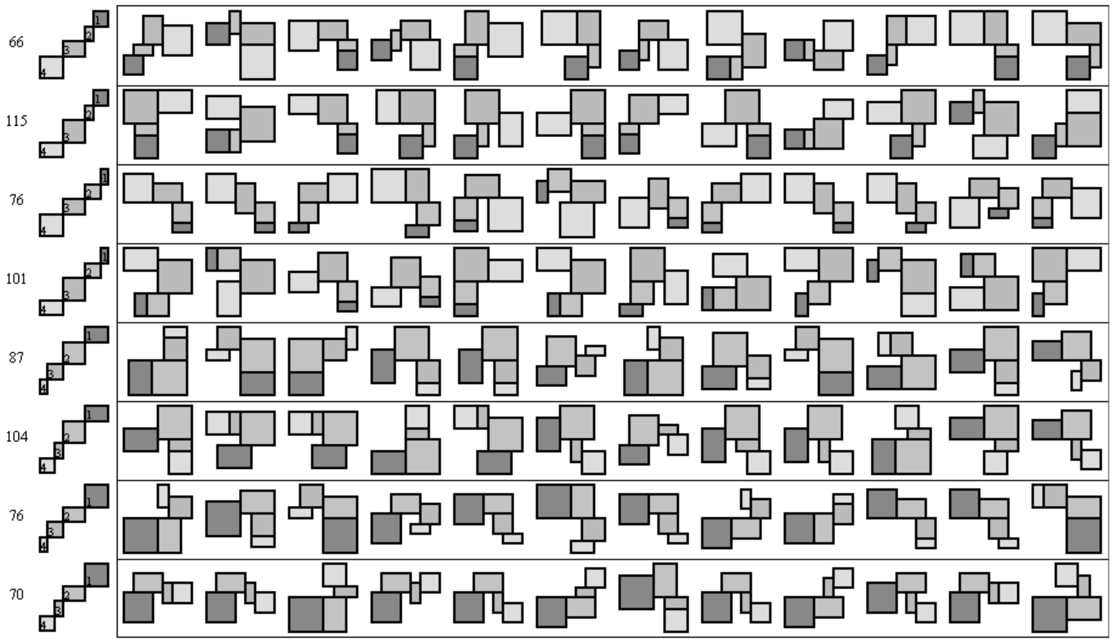


Figure 10. Samples of generated configurations (potential solutions); the number of potential solutions for the given permutation of the initial sequence of rooms is given on the left.

Out of a total of 695 generated configurations (potential solutions), 244 were proper (35%). The evaluation method is given in the next section.

### *3.6. Proper rooms configurations according to the communication criteria*

Unlike the rooms, the corridors are not predefined because their shape is not important as long as the communication works, in other words it is possible to enter all the rooms as shown in figure 11. If the shape is important, the corridor should be introduced as an additional room to the input data and specific adjacency relationships should be defined.

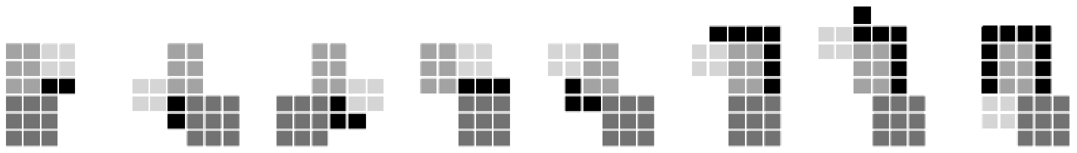


Figure 11. Some examples of corridors (indicated by black) communicating various configurations of an apartment with four rooms (indicated by different shades of grey).

The classification of the proper rooms configurations is a fairly complex process and rather difficult to formalize. Even for a trained person it takes a while to decide whether a given configuration is proper or not. For this project the classification was based on the following requirements:

- Every apartment must have an independent sub-corridor
- The sub-corridors of the apartments must be communicated by the main corridor
- All the communication must take not more than four units
- If possible, the main corridor should be larger than the sub-corridors.
- The size of the sub-corridor should be proportional to the size of the apartment.

An example of a proper room configuration that is having efficient communication among improper potential solutions is shown in figure 12.

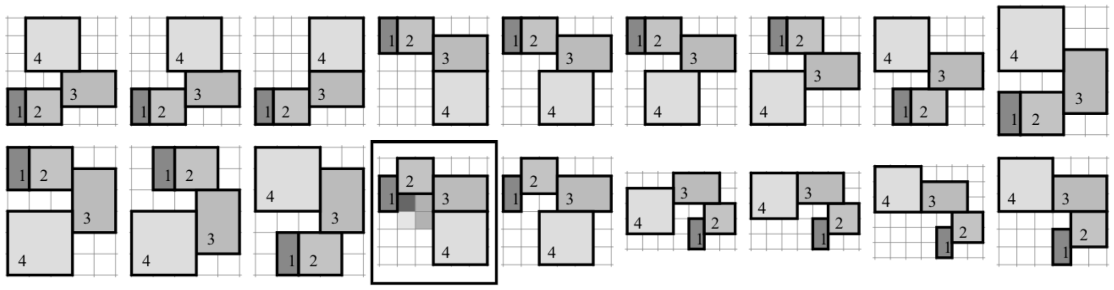


Figure 12. From all the configurations above (potential solutions) only the framed one allows the creation of the corridor system as described above. Two sub-corridors are connected by the main corridor; the number of units used for all communication is 4.

If both the numbers of considered configurations and number of rooms in the layouts are small, this kind of evaluation can be done simply by looking at every layout. Unfortunately in more realistic cases, the number of rooms is in the range of dozens, while number of solutions is in the thousands and (much) more.

The formalization of the communication criteria is rather difficult and its implementation as an additional  constraint during the generation process - seems problematic. Since a person evaluating a given  configuration uses simultaneously a lot of geometrical information which seems difficult to implement as a procedural algorithm, a machine learning method was applied for automation of this process.

## *3.7. The application of a neural network for the classification of rooms configurations.*

Although the number of solutions in the discussed example is not overwhelming and each of 695 configurations can be “manually” inspected, an application of a feed forward network (FFN) was investigated. The FFN was chosen for the following reasons:

- It is a classic supervised machine learning method.
- It is a common method for classification problems.
- It is always possible to determine whether a given configuration is proper or not.
- The evaluation criteria are difficult to formalize.
- Theoretically it is always possible to give sufficient training data. In this case over 15 x 15 = 225 training vectors were required- 622 were provided.
- All the classes are equally represented by the data
- All data samples are similar

The original encoding was compressed by removing any redundant information, such as room and apartment number etc. as shown in figure 13.

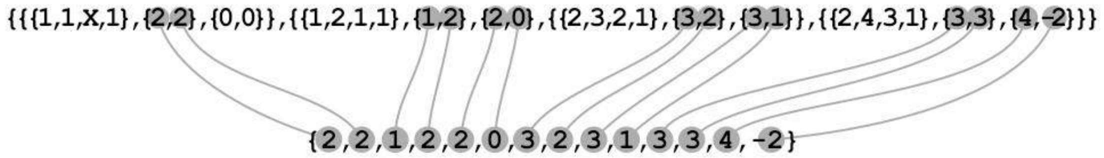

{{{1,1,X,1},{2,2},{0,0}},{{1,2,1,1},{1,2},{2,0},{{2,3,2,1},{3,2},{3,1}},{{2,4,3,1},{3,3},{4,-2}}}
{2,2,1,2,2,0,3,2,3,1,3,3,4,-2}

Figure 13. The original encoding of the room configurations compressed to a 14-dimensional vector for the FFN.

All the 695 potential solutions were transcribed into the 14 dimensional vectors as visualized in figure 14. The first part of them became the training and the second part- the validation data for the neural network.

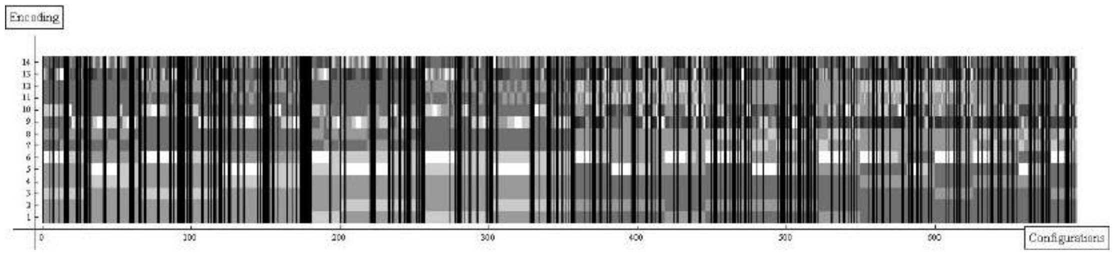


Figure 14. All 695 configurations (potential solutions) encoded in 14-dimension vectors; the shades of grey correspond to the vector's coordinates (min=white, max=dark grey). The proper configurations are marked black.

A 14 input/ 1 output feed forward network (FFN) with one hidden layer with 14 neurons and activation function of Sigmoid type was initialized. The training was done by the Levenberg-Marquardt algorithm with random initial weights in maximum 1000 iterations- the second terminating condition was when there was no change in 5 consecutive iterations of λ- which is the vector with one component describing the width of the basis function for each neuron. The training of the FNN in 10 trials expressed by root mean square deviation is shown in figure 15.

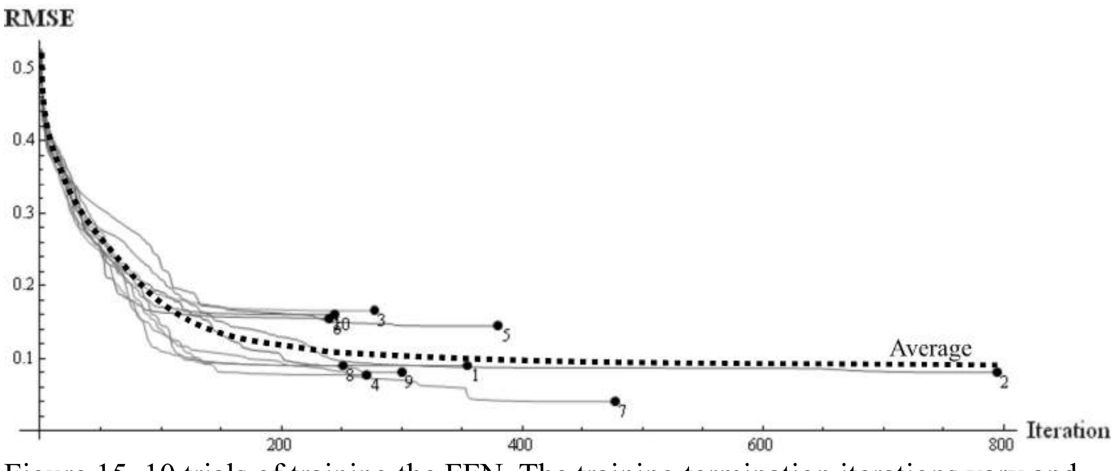


Figure 15. 10 trials of training the FFN. The training termination iterations vary and depend on the training progress. The interpolated average training curve is shown by the dotted line.

Performance of FNN in 10 training trials is compared in figure 16.

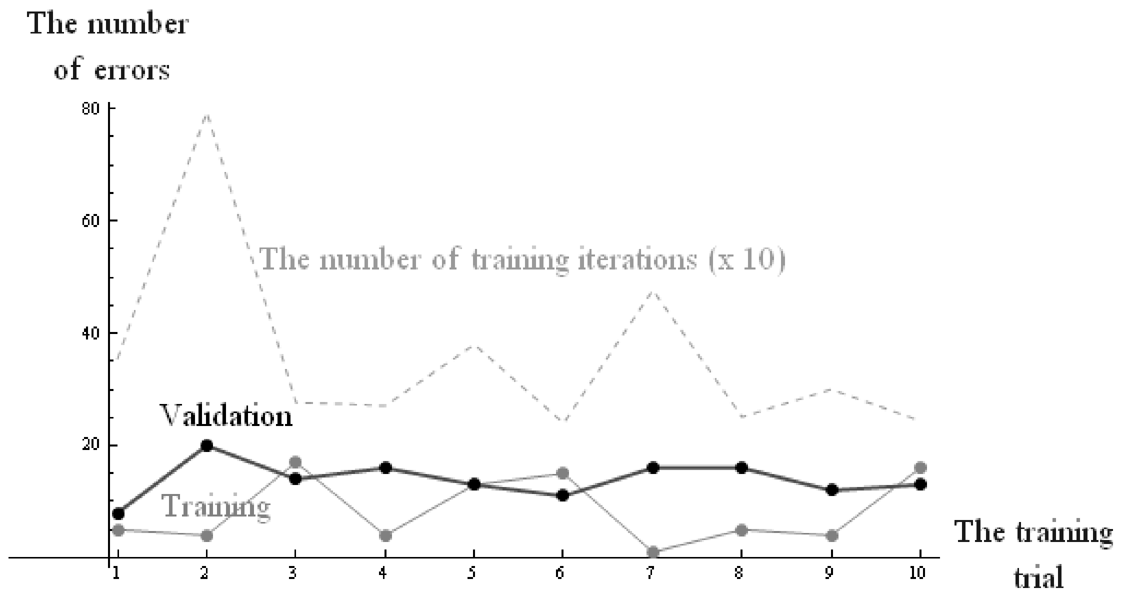


Figure 16. The errors in responses of the 10 trained networks. The errors for training data are shown in light grey; for validation data- in dark grey. The first trial network performed the best for the validation data (8 errors in 74 samples).

After applying a threshold function nearly 90% of validation samples were classified correctly by the best FFN of the 10 training trials. The detailed validation is shown in figure 17.

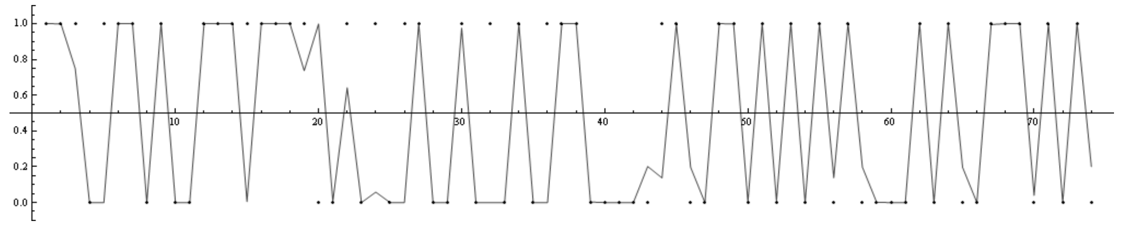


Figure 17. The validation of the best of the trained neural networks. The dots show the correct answers, the line shows the classification done by the neural network, after applying a threshold function there are 8 errors in 74 samples.

In these examples the shapes and sizes of corridors are in fact very limited, namely to a minimal (unit) square {1,1} and rectangles: {2,1} or {1,2}. Hence they could be included as a “room” for the configuration generation algorithm. Nevertheless, in more practical cases where the variety of shapes and sizes of corridors is much larger than those of rooms- using them as a special class of rooms seems impractical. It is crucial that the classification method is accurate- whether 90% accuracy is sufficient or not is arguable. Although misclassification of an improper configuration as a proper one can be verified later, much more disadvantageous is an error when a proper configuration is overlooked and excluded.

**4. Ranking the proper configurations according to additional criteria.**

The final selection of the proper configurations can be done by sorting them according to different parameters and values, for example: the length of the perimeter of a layout, the preference in the relative positions of given rooms, the size of the corridor, the geometrical complexity etc. In architectural practice however, usually not all of the client's requirements can be easily formalized and implemented. As an example, some configurations with various lengths of the layout perimeter are shown in figure 18. The smaller the perimeter, the more compact is the layout. In practice, building compactness, expressed by the area to volume ratio is a very important factor in energy conservation (Smeds 2007).

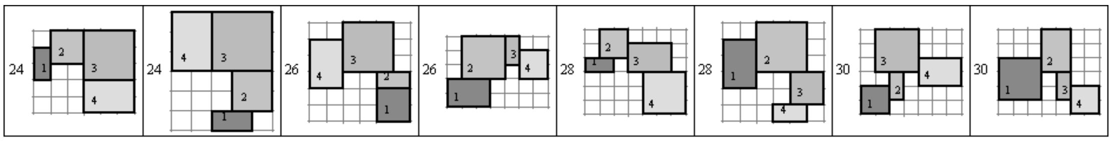


Figure 18. An example of configurations with various overall perimeters. The values are given left to each scheme. The corridors are not included in the calculations and therefore are not shown.

Another practical criterion that can be easily implemented for layout ranking is the preferred location of given rooms, for example room number 3 should be as close to the north-west (top left) corner as possible as shown in figure 19.

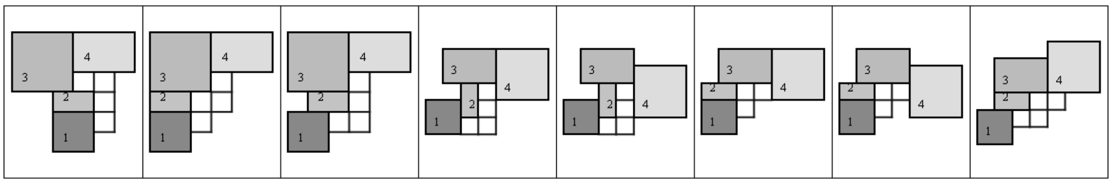


Figure 19. Examples of room configurations with room no. 3 located in the upper left corner of the layout.

One of the most intuitive criterion besides the overall perimeter of the building and certain rooms' locations, is the size of the corridors. Figure 20 shows some configurations with the smallest possible corridors (only the apartments' corridors are shown, the main corridor is excluded).

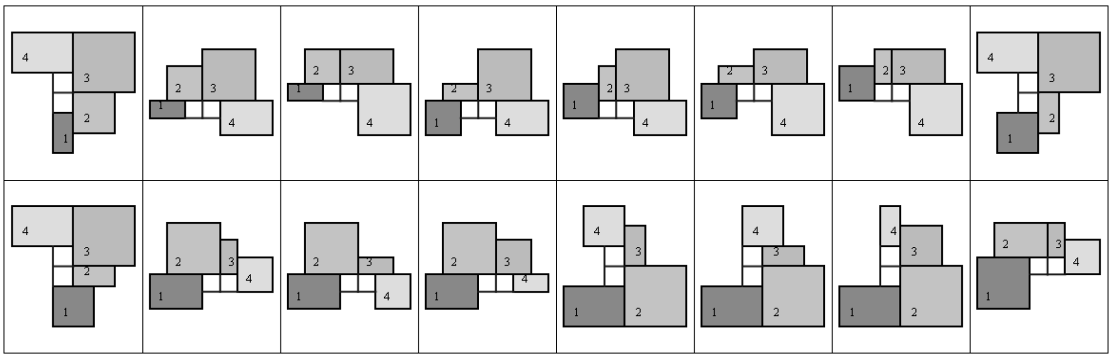


Figure 20. Examples of layouts with the smallest internal corridors (the main corridor is not shown but in all the cases can be done with a 2-unit rectangle).

Another highly intuitive criterion is the geometrical complexity of a layout which can be expressed by the total number of unique coordinates of all corners of each room. This means that if the corners of two rooms overlap, it is counted as one. The greater this number the more complex is the geometry of the layout (figure 21).

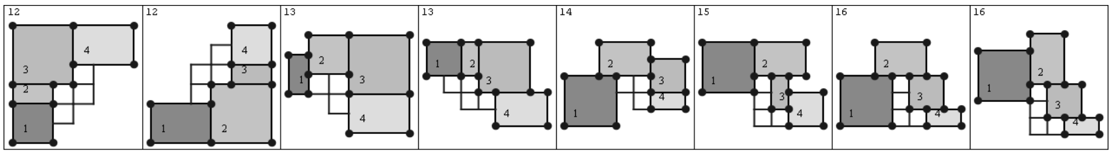


Figure 21. Some configurations with different geometrical complexity from the simplest to the most complex. The value describing the complexity of the rooms configuration is shown for each layout.

It also seems rational to include the corridors in the complexity calculation. The same configurations are shown with corridors in figure 22. The additional complexity of the corridor is expressed by the number of independent, that is not shared vertices. Usually low geometrical complexity of the rooms cause high complexity of the corridors and vice versa.

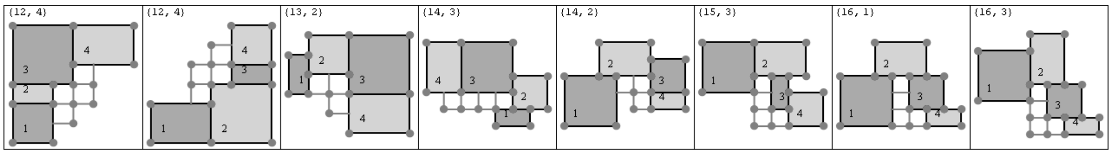


Figure 22. The same configurations as in figure 21 shown with the corridors' unique vertices. The main corridors are not included.

The combination of both values- the complexity of rooms and the complexity of the corridors gives a different sorting order. The values can be weighted for finer adjustments. Some examples of configurations where the sum of the room and corridor complexity equals 15 are shown in figure 23.

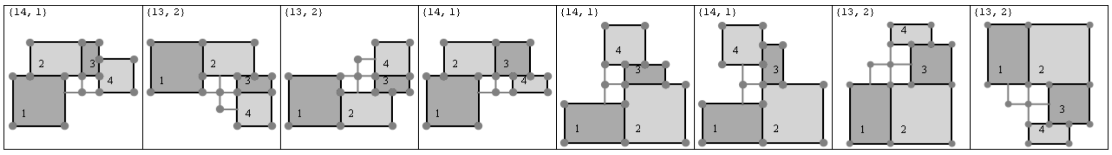


Figure 23. An example of configurations where the sum of the room and corridor complexity is the lowest and equals 15.

***4.1. Multi-objective optimization of a functional layout of rooms.***

The ranking of configurations according to multiple criteria can be considered as a multi-objective optimization problem:

$$\underset{x\in D}{Minimize}\,\mu(x) = \begin{pmatrix} \mu_1(x) \\ \mu_2(x) \\ \vdots \\ \mu_m(x) \end{pmatrix} \tag{3}$$

Where $m$ is the number of objectives; $D$ is the feasible region in the decision space; $\mu_1(x),...,\mu_m(x)$ are the coordinates of the image of $x$ in the objective space.

As an example, the following objectives representing the practical knowledge are subject to optimization (minimization):

(1) the overall geometrical complexity (including the corridors),
(2) the size of a corridor and
(3) the preference in the position of a given room- room no. 3 should be as far south as possible.

These parameters can be viewed as representing of the designer's expert knowledge or client's preference and can be visualized in the 3D space by assigning each of them to an axis (figure 24).

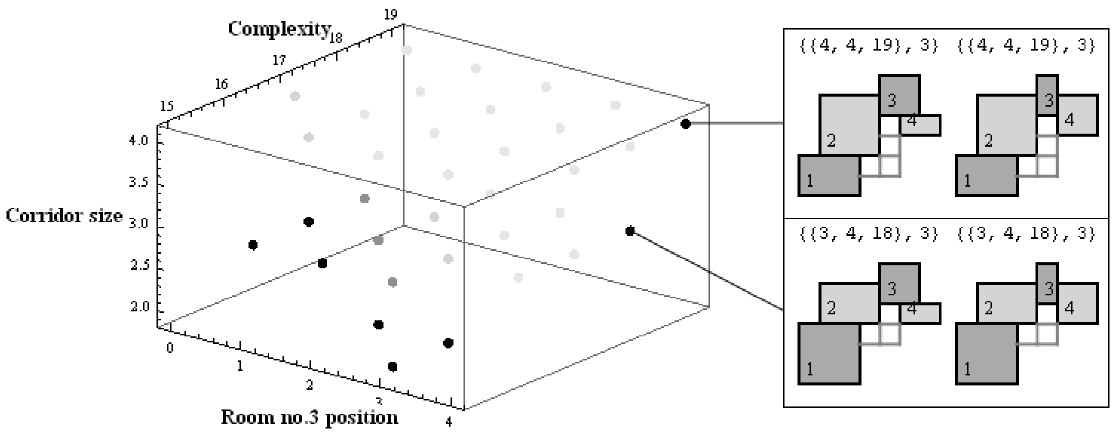


Figure 24. The points in 3D represent the objective function space of the 3-objective optimization of room configuration. One point can represent more than one configuration as shown on the right. Black points represent solutions for the minimization of all three objectives equally. The the brighter the dot, the smaller the value of the sum of the three objective functions.

In practice, the decision as to which configuration is better than another is rather difficult because only some criteria can be precisely defined, but usually not all of them and so far in practice, the final evaluation is most likely to be based on an arbitrary decision.

The simplest method is to combine all the objective functions by a weighted sum into a single aggregate objective function (AOF):

$$AOF = w_1F_1+w_2F_2+...+w_mF_m \quad (3)$$

Where $m$ is the number of objectives, $F_1...F_m$ are the objective functions and $w_1...w_m$ are the weights given to the objective functions.

The advantage of using this method is that the influence of each objective function can be adjusted by a weight. However when the number of objective rises, the manual setting of the weights becomes very difficult (Messac 2000). The AOF combines the size of the corridor ($2 \leq n \leq 4$), the distance of the room no. 3 from the southernmost edge of the layout ($0 < n < 4$) and the geometrical complexity of the configuration ($15 \leq n \leq 19$). The parameters were normalized. The histogram of optimization with all weights equal to 1 is shown in figure 25.

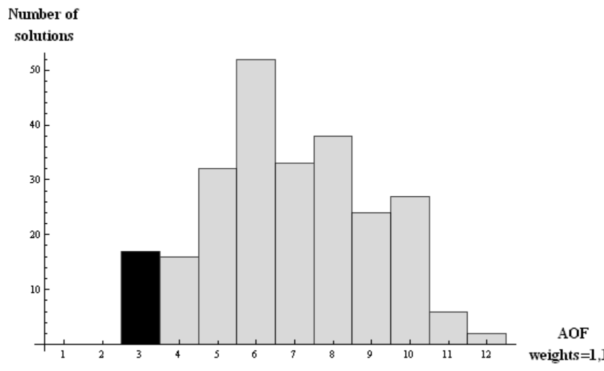


Figure 25. The black bar represents the configurations when AOF equals to 3 (1+1+1). There are 17 such "ideal" configurations for this AOF.

In real architectural problem, the weighting of the parameters is fairly difficult and requires a number of experiments. Moreover the evaluation of a solution is even more ambiguous due to the fact that the relative importance of the criteria is very difficult to estimate and it sometimes even changes during the decision making process.

In this example all feasible solutions in the objective function space can be directly visualized in a 3D space by assigning the three partial objective functions to three coordinates (figure 26).

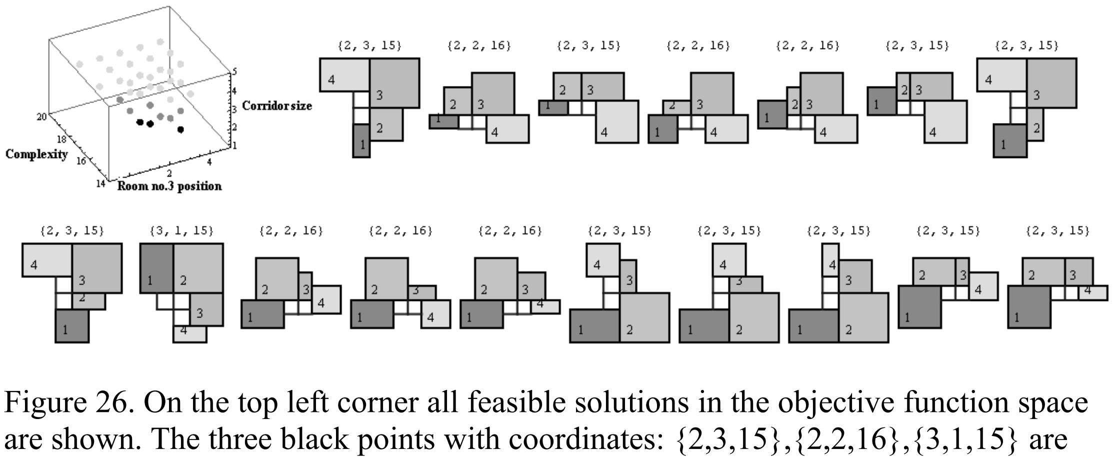


Figure 26. On the top left corner all feasible solutions in the objective function space are shown. The three black points with coordinates: {2,3,15},{2,2,16},{3,1,15} are located the closest to the origin of the coordinate system and represent the configurations of the minimal value of the AOF at weights:1,1,1. The values of all three parameters {Corridor size, room no. 3 from the southernmost edge, the complexity}are shown above each layout. The dark grey dots indicate the next solutions in this ranking.

The weight of the parameters of the AOF can be adjusted. An example is shown where the position of room no. 3 to the south has the highest priority (figure 27).

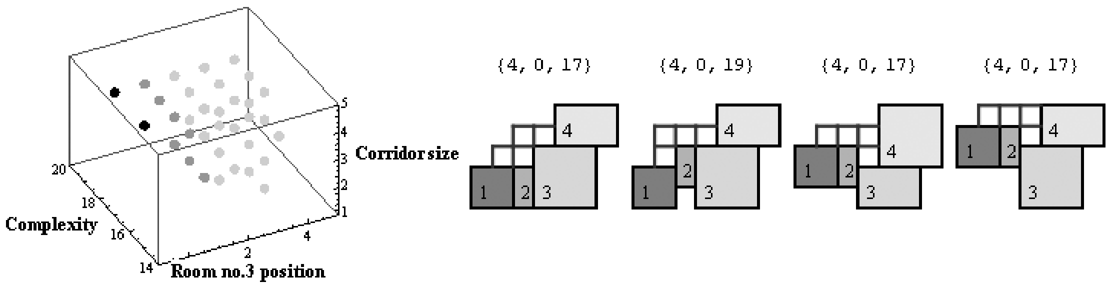


Figure 27. The position of the room no.3 to the south has preference over the rest of the criteria. The middle value of 0 indicates that the room no.3 is (one of) the southernmost rooms. The two black dots in the objective function space correspond to the four configurations shown on the right. The darker grey dots represent the next solutions in this sorting. Since the position of the room no.3 is the most important, the ranking of the solutions runs along this axis.

## 5. Application of the algorithm to a realistic layout problem

Although the concept was explained using simplified model, the current algorithm works efficiently also on moderate-size realistic problems.

The computational scale of this problem is also worth mentioning. For an example of a building with three apartments, having eight, six and six rooms respectively, using the formula (2) multiplied by $2^{15}$ since there are 15 rectangular rooms which have two possible orientations, the number of all possible room positions in the given sequence on the grid is:

274957638029688176640000 x $2^{15}$ (there are 15 rectangular rooms) ~ **$9.01\times10^{27}$**

Despite the fact that this is 100 times more than the size the observable universe expressed in meters, after approximately 25 minutes on a single core 1.86 GHz computer- 24 potential solutions were generated as shown in figure 28.

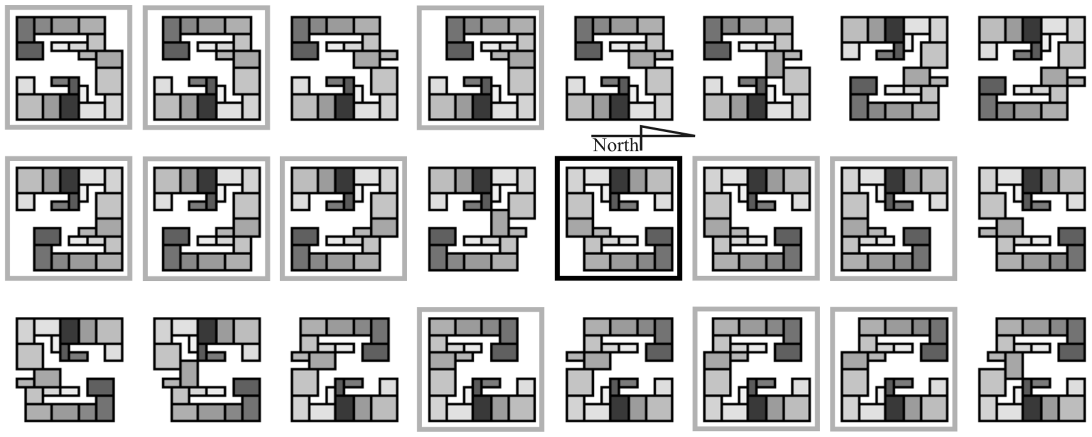
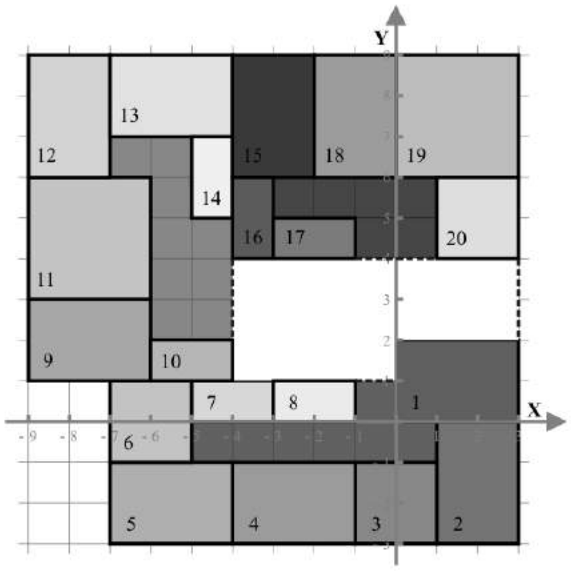


Figure 28. Although all these 24 solutions meet given very restrictive initial criteria, half of them have inefficient communication. 12 proper configurations are framed. The finally chosen one is framed in black.

In this case the initial data gave very constrained search space, which resulted in generation of only 24 configurations as shown in figure 28. The selection of the 12 proper configurations was done by authors and the final one was chosen according to the following criteria:

(1) The largest apartment to be on the East,
(2) the smallest apartment to be on the North,
(3) the north facade to be simple (low geometrical complexity).

The finally selected configuration is shown in figure 29.



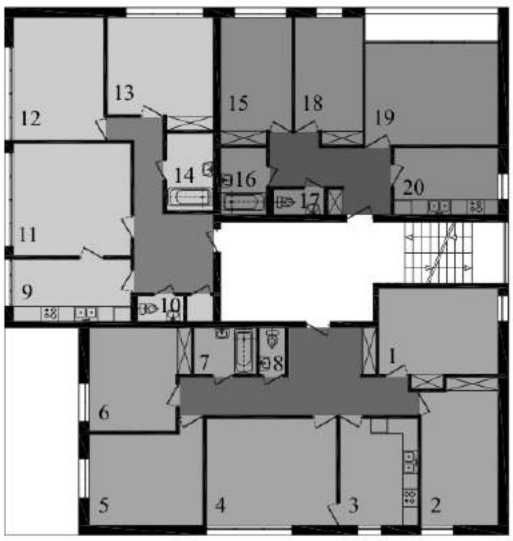


Figure 29. The solution of the functional layout optimization (on the left) and the architectural plan created on its basis (on the right).

This functional layout was adjusted by an architect according to several practical issues and was used as a floor plan for a project of a multi-family house as shown in figure 30.

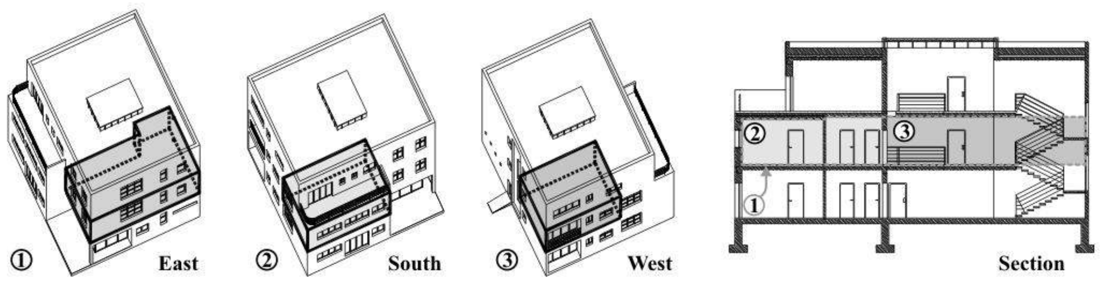


Figure 30. Section and three isometric views of the designed residential building with locations of the apartments. Dotted line in the section indicates the position of the apartment no.1 which is in front of the section plane.

As mentioned before, the CSP search algorithm is sensitive to the constraints, therefore setting them is crucial and rather difficult. Despite the fact that at the current stage, the process of setting the parameters for the algorithm is difficult and generating optimal solutions is rather time consuming- the results are finally acceptable from the architect's perspective.

## 6. Predicting the size of the search space

Since the search space tends to grow astronomically as shown in the chapter 3.4, it would be very desirable to estimate its size prior to the computation.

The numbers calculated by the formula (2) are discouragingly large and their practical usage is questionable since it allows overlapping of rooms and does not consider the constraints which greatly reduce the search space. However for this algorithm there is actually no straightforward way of predicting either if the solution set is non-empty or what is its cardinality, besides running an experiment. The preliminary evaluation whether the  constraints are too tight - so there is no solution - can be done by checking if the adjacency matrix of the relationships gives a planar graph (figure 31).

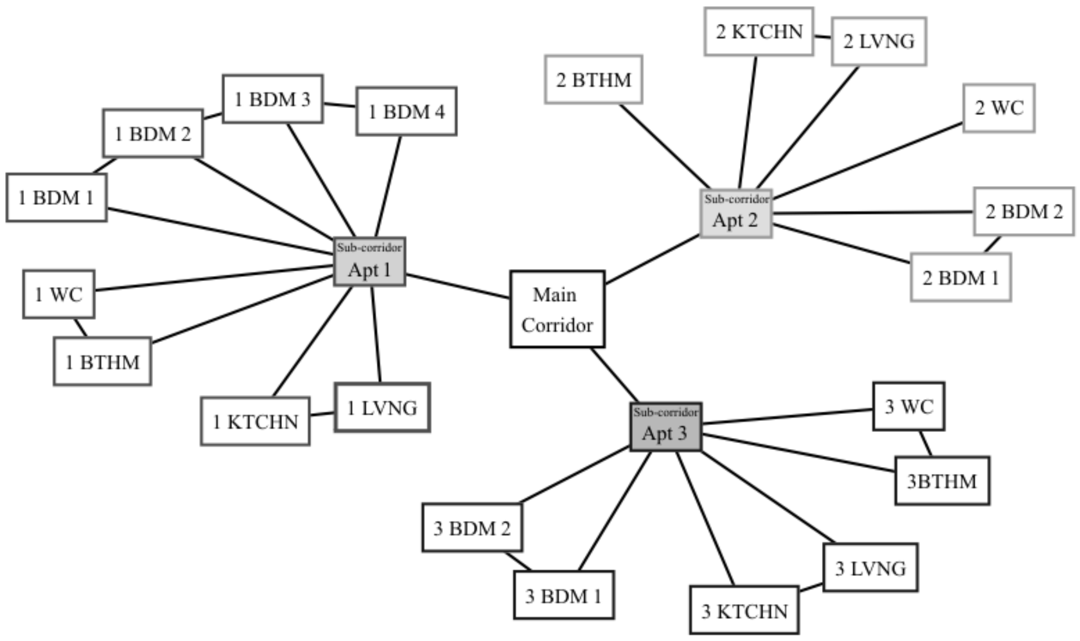


Figure 31. A graph representation of the relationships given for an example of a realistic layout problem has a planar embedding, therefore the solution may exist.

Planarity is a necessary but insufficient condition for a solution to exist. Although the planarity can be checked by available algorithms, in case of non-planarity two problems occur - the crossing minimization on almost planar graphs (Hliněný and Salazar 2007) as well as the decision on which edge (relationship) should be removed.

Graph theory was already implemented for layout optimization of a rectangular plan in a commercial program FactoryOPT (Montreuil *et al*. 1993). However in the presented approach the outline of the building is not limited to a rectangle and there are many other geometrical constraints, thus the solutions (rooms configurations) can not be derived directly from relationships given by a graph. The application of graph theory for the initial validation whether the constraints are reasonable and for the approximation of the size of the feasible search space is currently under investigation (figure 32).

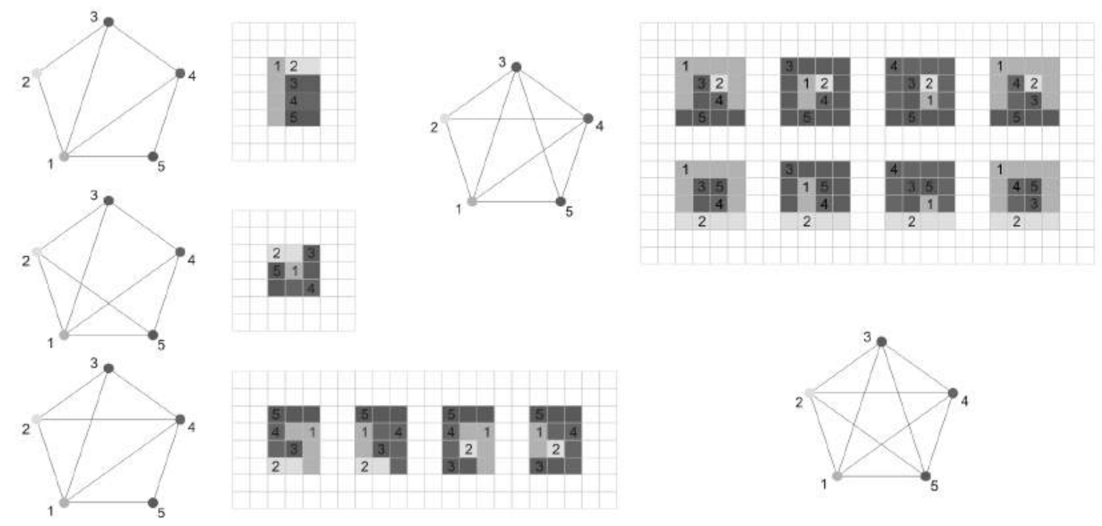


Figure 32. Some examples of room adjacencies represented by a graph and all smallest possible functional layout realizations on a square grid. “Holes” in the functional plan are not allowed (a hole should become an extra room of with “hole” function); obviously a non-planar ($K_5$) graph cannot be realized as a functional layout, which by definition is represented in a 2D plane.

The figure 30 shows an attempt to connect the abstract notion of a graph with a concrete geometry of an architectural layout. Building layouts have very rigid topological structure and notion of “outside” and “inside” as oppose to the dimensionless graphs which usually have a variety of embeddings. Finding the smallest possible functional realizations for a given relationships may help to predict the feasibility of the search, the size of the search space or be a starting point for the heuristic optimization.

## 7. Conclusions and future work

(1) The architectural design is a creative act, always to a certain degree arbitrary and intuitive. The purpose of this project is to provide a method which as objectively and as flexibly as possible ranks all the proper configurations according to the criteria that can be formalized by the designer.
(2) The proposed framework is general and the aims, criteria, parameters, weights etc., can be customized.
(3) The algorithm should become more coherent and improved for performance, flexibility and interactivity.
(4) The method of classifying the proper configurations should be improved. Perhaps the use of machine learning methods is not necessary or can be trained to give 100% accuracy.
(5) The adjustment of the decision variables should be improved. Presently it is done by a simple trial-and-error.

(6) In this project all the relationships among the elements are limited to the 2D horizontal plane. Some problems in reality though include the 3rd dimension (Bier *et al*. 2008). Nevertheless, although the physical space is 3D, due to the anthropological nature of the architectural design the space in the most of the problems is strongly anisotropic. Changing the vertical coordinate (for example moving) is much more difficult and rare than any of the horizontal ones. Although in any multi-story building there are elements of vertical transportation such as stairs, elevators, piping, etc., there is no need to generalize the problems to 3D. The step-down to the 2.5D space seems rational and practical. For example although a statement: "the kitchen is adjacent to the living room" can be realized by adjacency both on the horizontal plane and by placing one room over another- for occupants these are certainly two different kinds of adjacencies, especially for the elders. Nevertheless extending the problem to more than 2D is necessary.

(7) Implementing a solution space prediction algorithm is also necessary. As mentioned above it would be practical to know prior to the computation if there will at least be one solution produced for the given problem.

(8) A study on grid systems for various types of buildings is under consideration.

(9) A study on applying different types of knowledge in the algorithm is under consideration.

(10) An integration of the algorithm with a CAD system is under consideration.

(11) An implementation of fuzzy logic in the formulation of the layout parameters such as rooms' and apartments' sizes, the outline of a lot, adjacency relationships etc. and for the configuration classification algorithm (Hsu *et al*. 1999) is under consideration. At the present state the optimization is done in a few distinctive steps. Perhaps application of the fuzzy logic could unify or at least simplify the procedure (Xiong and Rao 2004).

Machi Zawidzki
Currently attending a Doctoral Program under a MEXT Scholarship in Integrated Science and Engineering Major at Ritsumeikan University, Japan.
Doctoral course at the Institute of Fundamental Technological Research, Polish Academy of Sciences, Warsaw, Department of Eco-building Engineering.
Graduated form the Faculty of Architecture, Warsaw University of Technology.

Kazuyoshi Tateyama
Professor in the Department of Architecture and Urban Design, College of Science and Engineering at Ritsumeikan University.
Doctoral degree in Engineering from Kyoto University.
Graduated from Department of Civil Engineering, Kyoto University.

Ikuko Nishikawa
Professor in the Department of Human and Computer Intelligence, College of Information Science and Engineering at Ritsumeikan University.
Assistant Professor in the Graduate School of Science and Technology, Kobe University.
Doctoral degree in Science from Kyoto University.
Graduated from Department of Physics, Faculty of Science, Kyoto University.

## References

Akin, O., Dave, B., Pithavadian, S., 1992. Heuristic generation of layouts (HeGeL): based on a paradigm for problem structuring. *Environment and Planning B: Planning and Design,* 19, 33–59.

Baykan, C., 1991. *Formulating spatial layout as a disjunctive constraint satisfaction problem.* Thesis (PhD). Carnegie Mellon University.

Bier, H. et al., 2008. Prototypes for Automated Architectural 3D-Layout. *Virtual Systems and Multimedia in Lecture Notes in Computer Science*, 4820, 203-214.

Buffa, E., Armour, G., 1964. Allocating facilities with CRAFT. *Harvard Business Rev*., 42 (2), 136–159.

Changa, P.T., 2009. Applying fuzzy weighted average approach to evaluate office layouts with Feng-Shui consideration. *Mathematical and Computer Modelling,* 50, 1514-1537.

Cross, N., 1999. Natural intelligence in design. *Design Studies*, 20, 25–39.

Damski, J.C., Gero, J.S., 1997. An evolutionary approach to generating constraint-based space layout topologies. *In* R. Junge, ed. *CAADFutures 1997*, Dordrecht: Kluwer, 855-864.

Del Río-Cidoncha, M.G., Iglesias, J.E., Martínez-Palacios, J., 2007. A comparison of floorplan design strategies in architecture and engineering. *Automation in Construction,* 16, 559–568.

Flemming, U., Baykan, C., Coyne, R., 1992. Hierarchical generate-and-test versus constraint-directed search. *In*: J. Gero, ed. *Proceedings of the Artificial Intelligence in Design Conference'92*, Dordrecht: Kluwer, 817–838.

Flemming, U., Coyne, R., Fenves, S., Garrett, J., Woodbury R. 1994. SEED- software environment to support the early phases in building design. *IKM94 International conference on the application of computer science and mathematics in architecture and civil engineering*, Germany, Weimar, 5–10.

Hliněný, P., Salazar, G., 2007 . On the crossing number of almost planar graphs. *Lecture Notes in Computer Science*, Berlin / Heidelberg: Springer, 162-173.

Hsu, C.H., Jiang, B.C., Lee, E.S., 1999. Fuzzy neural network modeling for product development. *Mathematical and Computer Modeling,* 29, 71–81.

Jo, J.H., Gero, J.S. 1998. Space layout planning using an evolutionary approach. *Artificial Intelligence in Engineering*, 12, 149-162.

Kosiński, W., Weigl, M., 1998. General mapping approximation problems solving by neural networks and fuzzy inference systems. *Systems Analysis Modelling Simulation,* 30 (1), 11–28.

Leśniakowska, M., 1996. *Co to jest Architektura.* Warsaw: Agencja Kanon, 95.

Liang, L.Y., Chao, W.C., 2008. The strategies of tabu search technique for facility layout optimization. *Automation in Construction*, 17 (6), 657-669.

Liggett, R.S., 2000. Automated facilities layout: past, present and future. *Automation in Construction,* 9, 197–215.

Messac, A., 2000. Aggregate Objective Functions and Pareto Frontiers: Required Relationships and Practical Implications. *Optimization and Engineering*, 1, 171–188.

Michalek, J.J., 2002. Architectural layout design optimization. *Engineering Optimization*, 34 (5), 461-484.

Mitchell, W.J., 1990. *The Logic of Architecture: Design, Computation, and Cognition*. The MIT Press.

Montreuil, B., Venkatadri, U., Ratliff, H.D., 1993. Generating a layout from a design skeleton, *IIE Transactions*, 25 (1), 3-15.

Saridakis, K.M., Dentsoras, A.J., 2008. Soft computing in engineering design – A review. *Advanced Engineering Informatics,* 22, 202–221.

Sharpe, R., Marksjo, B., 1986. Solution of the facilities layout problem by simulated annealing. *Computers, Environment and Urban Systems,* 11 (4), 147–154.

Smeds, J., 2007. Enhanced energy conservation in houses through high performance design. *Energy and Buildings*, 39 (3), 273-278.

Sowa, J. F., 2008. Conceptual Graphs. *In*: F. van Harmelen, V. Lifschitz, B. Porter eds. *Handbook of Knowledge Representation*, Elsevier, 213-237.

Steuer, R.E., 1986. Multiple Criteria Optimization: *Theory, Computations, and Application*. New York: John Wiley & Sons, Inc.

Tam, K., 1992. Genetic algorithms, function optimization and facility layout design. *European Journal of Operational Research,* 63, 322–346.

Terzidis, K., 2006. *Algorithmic Architecture*. Architectural Press, 44.

Wong, S.S.Y., Chan, K.C.C., 2009. EvoArch An evolutionary algorithm for architectural layout design. *Computer-Aided Design,* 41, 649-667.

Xiong, Y., Rao, S.S., 2004. Fuzzy nonlinear programming for mixed-discrete design optimization through hybrid genetic algorithm. *Fuzzy Sets and Systems,* 146, 167–186.

Yeh, I-C., 2006. Architectural layout optimization using annealed neural network. *Automation in Construction,* 15, 531-539.