ISSN 1392–3730
JOURNAL OF CIVIL ENGINEERING AND MANAGEMENT
http:/www.jcem.vgtu.lt

2005, Vol XI, No 1, 13–21

OPTIMIZATION OF ARCHITECTURAL LAYOUT BY THE IMPROVED GENETIC ALGORITHM

Romualdas Baušys, Ina Pankrašovaitė

Dept of Graphical Systems, Vilnius Gediminas Technical University, Saulėtekio al. 11, LT-10223 Vilnius, Lithuania. E-mail: gsk@fm.vtu.lt

Received 12 July 2004; accepted 09 Nov 2004

Abstract. In this paper we consider architectural layout problem that seeks to determine the layout of Units based on lighting, heating, available sizes and other objectives and constraints. For a conceptual design of architectural layout we present an approach based on evolutionary search method known as the genetic algorithms (GAs). However, the rate of convergence of GAs is often not good enough at their current stage. For this reason, the improved genetic algorithm is proposed. We have analysed and compared the performance of standard and improved genetic algorithm for architectural layout problem solutions and presented the results of performance.

Keywords: architectural design, floorplanning, layout, conceptual design, optimization, genetic algorithms.

1. Introduction

Architectural conceptual design [1, 2] has been defined as the initial phase of design process during which the designer takes a specification for an object to be designed and generates many broad solutions on it where each solution should be sufficiently detailed. In order to generate new solutions the designer either uses novel components or combines known components in a novel way.

Conceptual design is one of the most critical because at this phase the bulk of the decisions, which determine outcome performance and cost are taken. At the end of conceptual design phase, 80 % of the costs of a designed object have been determined and yet most attempts to reduce costs occur. Furthermore, most errors in design are due to the use of a flawed conceptual design and the effects of poor conceptual design can never be rectified by good detailed design [3–6].

Architectural layout has been recognised as an important activity in architectural conceptual layout-planning by field practitioners and researchers. Architectural layout is concerned with finding feasible locations and dimensions for a set of interrelated components that meet all design requirements and maximise design quality [7].

First, the conceptual architectural layout problem is defined as the problem of 1) identifying the shape and size of the components (rooms or walls) to be laid out, 2) identifying constraints between components, and 3) determining the relative positions of these components that satisfy the constraints between them. Next, design objectives to optimise for are specified. This combinatorial problem is known to be NP-hard [8] and comes mainly through the freedom to place components on the plane. An attempt to perform topological optimization has been reported in [9].

There are different classes of layout problems that have been studied in literature: variations in the shape and size of components and the constraints between them. Layout components may have defined shape (square, rectangular etc) or a loose shape that is undesirable in layout stage. Furthermore, most existing algorithms handle rectangular components. Rectangular components may have: variable area with fixed aspect ratio, fixed area with (upper and lower bounded) continuously variable aspect ratio, fixed area with discrete allowed aspect ratio and variable area with (upper and lower bounded) continuously variable aspect ratio.

In the layout problem addressed here, the rooms have rectangular shape with variable size area, continuously variable aspect ratio and are placed by designer in order to avoid the NP-hard problem.

A number of methods have been developed for supporting aspects of conceptual design based on constraints. In some methods conceptual designs are represented as systems of algebraic constraints (equations) [10], others use interactive constraint processing with human designer [11].

Several approaches to implement conceptual design have been developed: case-based reasoning, knowledge-based approaches and adaptive methods.

Case-based reasoning (CBR) is a relatively new approach to decision automation [12] based on the premise

that new problems can be solved by modifying solutions to previously solved problems.

The knowledge-based approaches are based on a set of rules [15] or explanation-based reasoning [14] that provides the inference mechanism with the information needed to support the designer during the design process.

Adaptive search describes search and optimization techniques, such as genetic algorithms [15–17] and simulated annealing [18] that are heuristic in nature. In general, adaptive search techniques have been used as a basis for automated design applications. The goal of these applications has been improving the features of existing designs through the optimization of variables defined within the system. In order to enhance the search power, such techniques is often used in combination with others.

Several hybrid algorithms [7] using a combination of simulated annealing (SA) and gradient-based approaches, such as sequential quadratic programming (SQP) have been developed to find solutions for conceptual architectural layout problem. In this method, SA is used to search for a good starting point, and SQP is applied to find the local minimum near each starting point. In this way SA can search the space more globally with large moves, while SQP worries about the details. SQP can find several different local optima depending on where the start is chosen. Each point that SA selects is evaluated by locally optimizing it. Because of the local method nature, solutions are likely to be inferior local optima.

Some researchers have used just simulated annealing to search for an optimal solution [19]. The implementation of simulated annealing scheme relies on a layout representation where a new solution is generated and examined by perturbing the initial design state applying move operator. If a new design leads to an improvement in the objective function, this design is accepted and becomes the current state. If the new design leads to an inferior state, it may still be accepted with some probability (temperature). This method works with one solution at a time, to search for an optimal solution.

However, the preferable solution produced for the problem of this type has been found the genetic algorithm (GA) [20] as a robust global optimization approach. Rebaudengo and Sonza Reorda have obtained hybrid genetic algorithm [21] for layout problem, Gero and Kazakov have used the extension of genetic algorithms [22], which is based on the concepts of genetic engineering.

The main advantage of using a GA is that while other methods always process single solution in the search space, genetic algorithms maintain a population of potential solutions.

Although GAs can solve complicated engineering optimization problems much more effectively than traditional optimization methods, their rate of convergence in solving the problems is still not so good for many realistic problems. In this paper, the evolutionary direction operator has been proposed to improve the existing standard genetic algorithm. The improved GA is analysed, studied empirically and compared, in feasibility and performance evaluation, with standard GA for architectural conceptual layout problem. The simulation results demonstrate the improvements of the new algorithm to standard GA for studied architectural layout problem.

2. Architectural layout problem

2.1. Problem formulation

In the paper we deal with traditional architectural layout problem that seeks to find the non-overlapping geometry and size of a group of interrelated components (Units). The genetic algorithms have to position the Units in a meaningful, feasible way that meets all design specifications and optimizes the layout.

A Unit is defined as a rectangular, orthogonal space allocated for a specific architectural function. Using this representation, rooms, hallways, doorways, and building boundary are all represented as combinations of orthogonal rectangular Units. Fig 1 shows a Unit represented as a centre point of rectangle *(x*, *y)*, and the perpendicular distance from that point to each of the four walls (*E*, *W*, *N*, *S*).

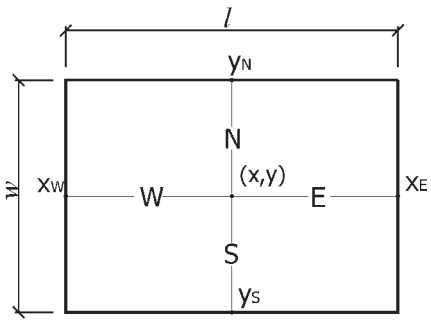


Fig 1. Representation of Unit

According to this representation design variables can be expressed as:

$$x_E = x + E\,, \tag{1}$$

$$x_W = x - W\,, \tag{2}$$

$$y_N = y + N\,, \tag{3}$$

$$y_S = y - S\,, \tag{4}$$

$$l = E + W\,, \tag{5}$$

$$w = N + S\,. \tag{6}$$

Such representation allows easily detect an overlapping area of the Units. Two Units, presented in Fig 2, are non-overlapping if one of the following inequalities is satisfied:

$$(x_{Wi} \geq x_{Ej}) \text{ OR } (x_{Wj} \geq x_{Ei}) \text{ OR } (y_{Si} \geq y_{Nj}) \text{ OR } (y_{Sj} \geq y_{Ni}). \quad (7)$$

A building boundary is a Unit that has other Units inside it. This Unit is not considered as living space. The building boundary is shown in Fig 3 by a thick line. Rooms are Units that serve for living activity. A hallway is a Unit with no physical walls that is not a living space. Hallways function as pathways. A doorway is a hallway that is constrained to geometrically intersect two Units.

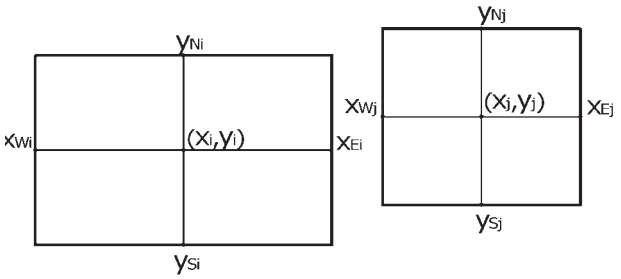


Fig 2. Two non-overlapping Units

Doorways are generally restricted to be small, and they are forced to intersect two other Units. They function to keep the two Units adjacent and connected, and to ensure that there is room for a door or opening between the rooms. The differentiation between living and non-living space is important only in optimization objectives that maximise the amount of space used for living relative to all other space.

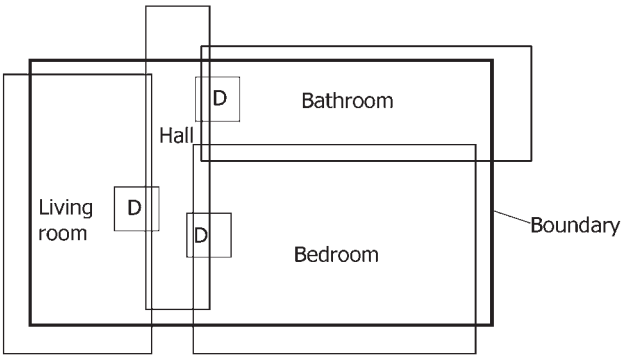


Fig 3. An example layout showing building boundary, four different types of Units and doorways

In Fig 3, the external rectangle represents the building boundary, the living room, bedroom, and bathroom are rooms, the hall is a hallway, and the three Units labelled "D" are doorways that define space between Units.

2.2. A theory of conceptual design

The model of conceptual design process can be regarded as a set of requirements that the designed object must satisfy. Fig 4 illustrates the model of conceptual design process.

It can be seen that conceptual design is an iterative process during which the designer generates a set of alternative solutions for an object based on a design specification. The design specification will contain a set of requirements that the object to be designed must satisfy. Based on these requirements, the designer begins, an iterative process of solution generation. During this process he will develop a solution that is intended to satisfy the design specification. This solution is then evaluated and compared against any solutions that have already been developed. Based on this comparison, the designer will choose to accept, improve or reject a particular solution. The process of solution generation will be repeated many times in order to ensure that a sufficiently large number of solutions have been considered.

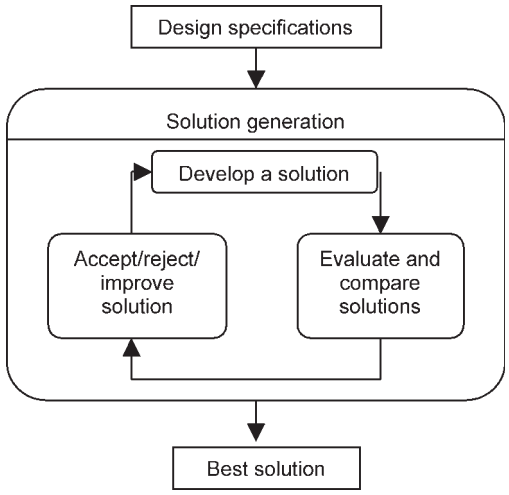


Fig 4. The model of the conceptual design process

Two categories of design requirements can be identified: functional requirements and physical requirements. The functional requirement is that the designed object can provide and physical requirements are that the designed object can satisfy.

The example of design specification for an architectural layout is presented in Fig 5.

Design specification
Functional Requirements: Minimise cost
Physical requirements: 1. the Units must be inside the main building boundary 2. two Units cannot occupy the same space 3. the doorways must be ensured 4. the windows must be ensured 5. the area of Units must be minimal 6. the ratio of Unit must be minimal 7. the window width cannot be larger than the wall width 8. the specific rooms must have minimal natural lighting

Fig 5. Design specification for architectural layout

In this paper, functional requirement to minimise cost can be considered as a set of objectives: heating cost objective, lighting cost objective, wasted space objective, doorways objective and hallway objective.

3. Mathematical model of architectural layout optimization problem

Based on the conceptual framework, five-objective and eight constraints non-linear programming was used.

The minimise heating cost objective estimates heating loss during cold months. The annual energy cost to heat the building is calculated as a function of the building boundary Unit shape, volume, surface area, and material as well as environment conditions. It is assumed that windows on all Units are constrained against external walls. The heating cost objective function is then:

$$f_1 = \Gamma_{heat} = \left(K_{gas} \times Q_{heat}\right) / \eta_{heater}, \tag{8}$$

where Γ_{heat} is the total heating cost, K_{gas} is the cost of gas per cubic metre, Q_{heat} is the annual heat lost and η_{heater} is the efficiency of the heater in Watts per cubic metre of gas. The entire procedure of heating loads can be found in [23].

The minimised lighting cost objective minimises the cost spent on lighting the building by encouraging natural lighting. The amount of natural lighting in room *i*, and the calculations of total required cost if all of this light is provided by electric lighting can be found in [23]. The total cost is:

$$f_2 = \Gamma_{light} = \Gamma_{elec} - \left(\sum_i \Theta_i\right)\beta_A 10^{-3}, \tag{9}$$

where Θ_i is the amount of natural lighting, *i* is the set of Units, Γ_{light} is the total lighting cost, Γ_{elec} is the maximum possible electricity cost and β_A is the number of hours of available light per months. The entire procedure of lighting cost can be found in [23].

The minimise wasted space objective minimises building space that is not living space. This could be space used for hallways or unallocated space inside the building boundary. The objective is formulated as:

$$f_3 = \left(l_1 w_1 - \sum_{k \in Rooms} l_k w_k\right), \tag{10}$$

where 1 indicates Unit 1 and l_1, w_1 is the length, width of Unit 1, which is assumed to be the building boundary Unit, l_k, w_k is the length, width of room and *k* is the number of rooms.

The minimised doorways objective brings connected Units together. Doorways may be constrained to be small to keep Units together. This objective can be used to bring Units together if possible, but allow them to be separated if necessary, providing that there is a doorway between them. The objective is formulated as:

$$f_4 = \left(\sum_{l \in Doorways} l_l w_l\right), \tag{11}$$

where l_l, w_l are the length, width of doorway and *l* is the number of doorways connecting Units together.

The minimised hallway objective is used to provide extra living space where possible. The objective is formulated as:

$$f_5 = \left(\sum_{m \in Hallways} l_m w_m\right), \tag{12}$$

where l_m, w_m are the length, width of hallway and *m* is the number of hallways.

These objectives can be combined into a single objective function using a weighted sum of the individual objective functions. The programming model can be expressed as follows:

minimise $$\sum_{n=1}^{5} w_n f_n(x), \tag{13}$$

subject to

$$y_{Ni} \le y_{Nj}, y_{Sj} \le y_{Si}, x_{Ei} \le x_{Ej}, x_{Wj} \le x_{Wi}, \tag{14}$$

$$\min\left(x_{Ej} - x_{Wi}, x_{Ei} - x_{Wj}, y_{Nj} - y_{Si}, y_{Ni} - y_{Sj}\right) \le 0, \tag{15}$$

$$\begin{aligned} &\min\{\max(d_i, d_j) - x_{Ej} + x_{Wi}, \\ &\max(d_i, d_j) - x_{Ei} + x_{Wj}, \\ &\max(d_i, d_j) - y_{Nj} + y_{Si}, \\ &\max(d_i, d_j) - y_{Ni} + y_{Sj}\} \le 0, \end{aligned} \tag{16}$$

$$\begin{aligned} &\min\{(x_{Ei} - x_{Ej})^2, (x_{Wi} - x_{Wj})^2, \\ &(y_{Si} - y_{Sj})^2, (y_{Ni} - y_{Nj})^2\} = 0, \end{aligned} \tag{17}$$

$$A_{\min i} - l_i w_i \le 0, \tag{18}$$

$$R_{\min i} l_i - w_i \le 0 \quad \text{and} \quad R_{\min i} w_i - l_i \le 0, \tag{19}$$

$$\omega_{Ni} \le l_i, \omega_{Si} \le l_i, \omega_{Ei} \le w_i, \omega_{Wi} \le w_i, \tag{20}$$

$$\varphi_{\min i} - \frac{\Theta_i}{A_i \theta_{reqi}} \le 0. \tag{21}$$

where *i*, *j* = 1, ..., *n*, two interrelated Units; $x_{Wi}, x_{Ei}, y_{Ni}, y_{Si} \in \Re$, is the Unit west, east, north and south wall location respectively; $\omega_{Wi}, \omega_{Ei}, \omega_{Ni}, \omega_{Si} \in \Re_+$, is the width of west, east, north and south window location respectively; d_i, d_j is the minimum size for a door or opening in Unit *i*, Unit *j*; $A_{\min i}$ is the minimum area of Unit *i*; A_i is the area of Unit *i*; $R_{\min i}$ is the minimum ratio of Unit *i*; θ_{reqi} is the required natural lighting per square metre; $\varphi_{\min i}$ is the minimum percentage of required lighting that is provided by natural light.

In expression (13), $f_n(x)$ is the nth objective function, wn is the weight (relative importance) of the nth objective function, and n is the total number of objective functions. Objective functions are measured in different units and after obtaining results, weights can be adjusted to compensate and guide the design to desired results.

The constraint group (14) forces Units inside the main building boundary. The constraint functions (15) are used to prevent intersection of two Units from occupying the same space. The constraint group (16) is used when Units are forced to intersect in order to ensure access (doorways). The constraints (17) are used to force a Unit to the edge of a building boundary because of a window or external door. The bound size constraint (18) includes constraint to bound the area of a Unit. Unit area is only reduced to improve objective functions, such as cost objective. The minimum ratio constraint group (19) can be used to maintain a desired aesthetic scheme or prevent long, narrow rooms that may not be usable. The feasible window constraints (20) ensure that the window width cannot be larger than the wall it is on. In addition, these constraints ensure feasible window size. The bound lighting constraint (21) is used to ensure minimum natural lighting for specific rooms. An estimation of the amount of daylight entering a Unit with windows is calculated using environmental and material information.

This non-linear, non-smooth mathematical formulation of design problem is undesirable for gradient-based calculations. So, for the solution of such a problem we apply genetic algorithm.

4. Architectural layout representation by GA

When designing a genetic algorithm for a given problem, choosing the representation (ie constructing the chromosome) is the first step. A common representation used in the genetic algorithms is the fixed length bit string. In case of more complex problems a more sophisticated representation leads to better results.

For example, the architectural layout consists of four rectangular components (Fig 6) with variable parameters. Each Unit requires six parameters to fully define their geometry. Layout is defined by a number of non-overlapping components.

A genotype for layout, represented in Fig 6, consists of a single chromosome of four blocks and each block consisting of eight genes:

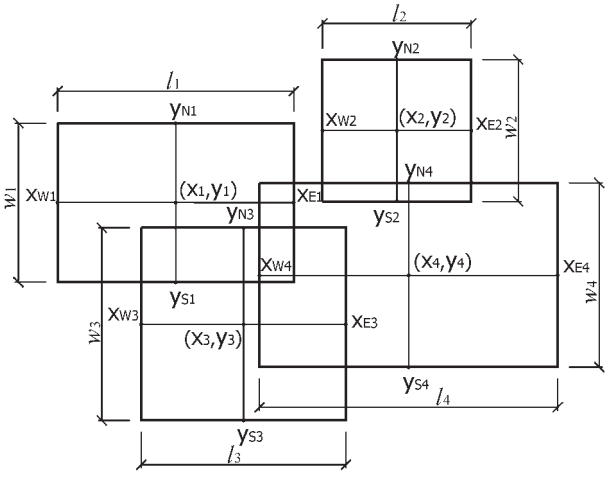


Fig 6. An example of architectural layout

$$Z = ((x_1, y_1, x_{E1}, x_{W1}, y_{N1}, y_{S1}, l_1, w_1), (x_2, y_2, x_{E2}, x_{W2}, y_{N2}, y_{S2}, l_2, w_2), (x_3, y_3, x_{E3}, x_{W3}, y_{N3}, y_{S3}, l_3, w_3), (x_4, y_4, x_{E4}, x_{W4}, y_{N4}, y_{S4}, l_4, w_4)). \quad (22)$$

This arrangement corresponds to the layout partitioning representation used to define the phenotype, with each block of genes being a coded Units shape and each gene being a coded parameter. In further calculations all variables are expressed as x_i variables.

In our case the problem consists of function optimization over real valued parameters. In such situations, real valued chromosome representation, in comparison with binary strings, can lead faster to the desired solution because we do not use transformation from real values to the genetic code.

5. Improved genetic algorithm

5.1. Evolutionary direction operator based genetic algorithm

Many researchers investigate how to increase the efficiency of genetic algorithm. Several tools have been implemented for searching the optimal layout [7], including a hybrid genetic algorithm GALLO [24], which exploits heuristic techniques, integrates them into the genetic approach and thus improves the performance of GA. GALLO procedure based on graph theory. A second hybrid search strategy was built on integration of global optimization methods such as genetic algorithms or simulated annealing with gradient-based algorithms [7]. However, this strategy is usually time consuming since it needs to evaluate gradients and can find locally optimal designs.

We think that the integration of different approaches is not the most perspective strategy and propose to improve the existing GA. In order to find the global optima efficiently, the implementation of evolutionary direction operator is the preferable strategy for solving the architectural layout problem under consideration. This operator does not require evaluation of gradients and thus is much less time-consuming than the gradient-based optimiser.

5.2. Evolutionary direction operator

The proposed evolutionary direction operator can be constructed as follows. Evolution from a present individual to an individual in the next generation is determined by two parental individuals of the present individual as well as their fitness. An individual is assumed to be represented by n values:

$$Z = (z_1, z_2, ..., z_n), \quad (23)$$

where each component value z_i represents a design parameter. The value of z_i is assumed ranging from $z_{i\min}$ to $z_{i\max}$, ie:

$$z_{i\min} \le z_i \le z_{i\max}, \quad i = 1, 2, \ldots, n. \quad (24)$$

With the use of the evolutionary direction operator the individuals of the next generation, ie children, are determined from the individuals at the present and the fitness of the present individual and parents. The child $(z_{C1}, z_{C2},..., z_{Cn})$ is expressed as follows:

$$z_{Ci} = \max[\min(z_i^*, z_{i\max}), z_{i\min}], \quad i = 1, 2, \ldots, n, \quad (25)$$

$$z_i^* = z_i + S \cdot sign(F - F_{P1})(z_i - z_{P1i}) + T \cdot sign (F - F_{P2})(z_i - z_{P2i}), \quad (26)$$

$$S,T \in [0,1].$$

Here $(z_{P11}, z_{P12},..., z_{P1n})$ is a parent 1 with fitness value F_{P1}, $(z_{P21}, z_{P22},..., z_{P2n})$ is a parent 2 with fitness value F_{P2} and $(z_1, z_2,..., z_n)$ is a present individual with fitness value F.

From the above definitions, it can be seen that two directions of the evolution have been obtained: one from parent 1 to the present and the other from parent 2 to the present. The evolution to the next generation is determined from these two directions by using equation (25). The contribution of the direction from parent 1 is expressed by the second term of the right-hand side of equation (26), whereas that of the direction from parent 2 is by the third term. The random parameters S and T give the magnitude of the contributions of the two directions to the evolution. The signs of the directions are selected depending on whether or not the present fitness is improved through the evolution from the parents.

For architectural layout problem, presented in section 4, two parent chromosomes were chosen for reproduction:

Parent 1:

$$Z_{P1} = ((x_{P11}, y_{P11}, x_{P1E1}, x_{P1W1}, y_{P1N1}, y_{P1S1}, l_{P11}, w_{P11}), (x_{P12}, y_{P12}, x_{P1E2}, x_{P1W2}, y_{P1N2}, y_{P1S2}, l_{P12}, w_{P12}), (x_{P13}, y_{P13}, x_{P1E3}, x_{P1W3}, y_{P1N3}, y_{P1S3}, l_{P13}, w_{P13}), (x_{P14}, y_{P14}, x_{P1E4}, x_{P1W4}, y_{P1N4}, y_{P1S4}, l_{P14}, w_{P14})) \rightarrow F_{P1}. \quad (27)$$

Parent 2:

$$Z_{P2} = ((x_{P21}, y_{P21}, x_{P2E1}, x_{P2W1}, y_{P2N1}, y_{P2S1}, l_{P21}, w_{P21}), (x_{P22}, y_{P22}, x_{P2E2}, x_{P2W2}, y_{P2N2}, y_{P2S2}, l_{P22}, w_{P22}), (x_{P23}, y_{P23}, x_{P2E3}, x_{P2W3}, y_{P2N3}, y_{P2S3}, l_{P23}, w_{P23}), (x_{P24}, y_{P24}, x_{P2E4}, x_{P2W4}, y_{P2N4}, y_{P2S4}, l_{P24}, w_{P24})) \rightarrow F_{P2}. \quad (28)$$

The child produced by evolutionary direction operator would be:

Child:

$$Z_C = ((x_{C1}, y_{C1}, x_{CE1}, x_{CW1}, y_{CN1}, y_{CS1}, l_{C1}, w_{C1}), (x_{C2}, y_{C2}, x_{CE2}, x_{CW2}, y_{CN2}, y_{CS2}, l_{C2}, w_{C2}), (x_{C3}, y_{C3}, x_{CE3}, x_{CW3}, y_{CN3}, y_{CS3}, l_{C3}, w_{C3}), (x_{C4}, y_{C4}, x_{CE4}, x_{CW4}, y_{CN4}, y_{CS4}, l_{C4}, w_{C4})) \rightarrow F_C. \quad (29)$$

Fig 7 shows how a child Z_C is determined given the properties of two parent chromosomes Z_{P1} and Z_{P2};

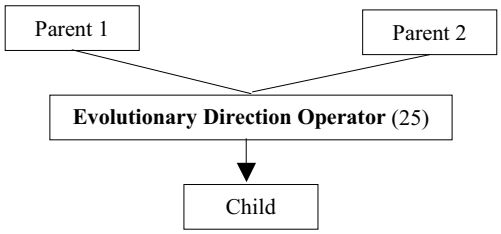


Fig 7. The child determined by evolutionary direction operator

The genotypes (27, 28, 29) corresponding phenotypes (layout designs) are represented in Fig 8.

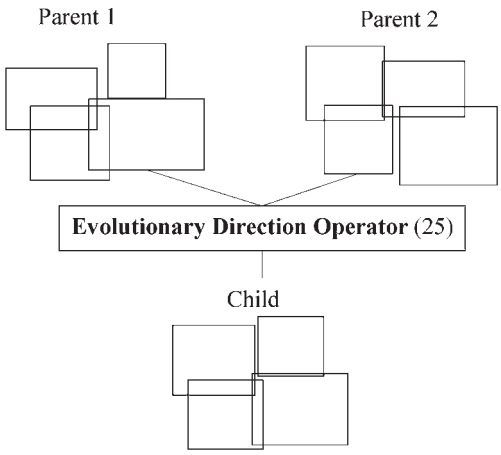


Fig 8. The phenotype (child) evolved by evolutionary direction operator

Fig 8 provides geometrical interpretation of the child generated by evolutionary direction operator using the properties of two parents.

6. Case study

In this section we study and compare the performance of standard genetic algorithm and evolutionary direction operator genetic algorithm [25, 26]. The initial concept of particular layout is sketched out (Fig 9).

The problem under consideration consists of three bedrooms, a bathroom, dining room and a living area with a hallway structure consisting of two parts. Connections are defined as shown in Fig 9 by adding doorways (a rectangular Unit marked 'D') between Units. Each Unit is initialised with its own constraints for a minimum area, minimum and maximum length and width. Unit constraints are represented in Table 1.

Default constraints such as constraints to force all Units inside the building bounds (14), to prohibit intersection (15) between all Units combinations, doorways constraints (16) that force intersection between the doorway and each of the connected Units such that the intersection overlap is large enough for a door, feasible win-

Table 1. Room constraints

Room	*Min area (sq m)*	*Min length (m)*	*Max length (m)*	*Min width (m)*	*Max width (m)*
Living Room	25,00	4,50	6,50	4,00	6,00
Dining Room	12,00	3,50	5,50	1,80	5,40
Bedroom 1	22,00	3,40	5,49	4,50	6,00
Bedroom 2	20,00	4,00	6,10	3,40	6,10
Bedroom 3	10,00	3,50	5,50	2,50	5,20
Hall 1	10,00	5,40	6,50	1,20	3,00
Hall 2	10,00	2,50	5,00	2,50	5,00
Bathroom	10,00	2,50	5,00	2,50	5,00

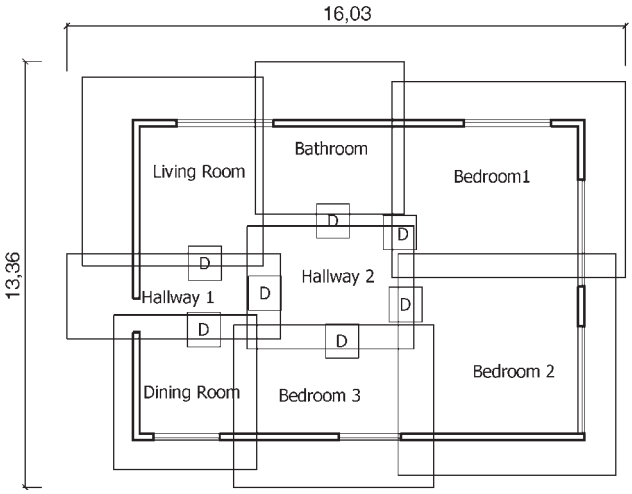


Fig 9. Initial layout

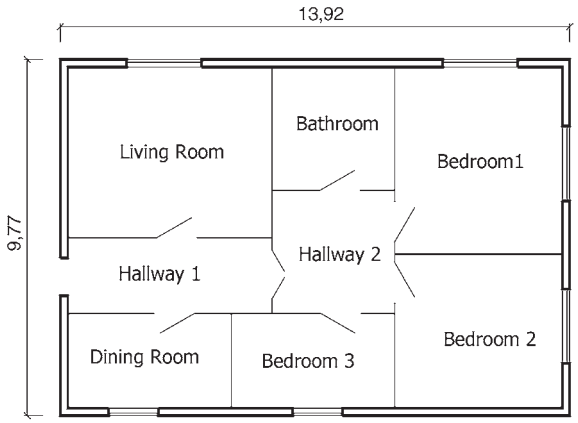


Fig 10. Optimal architectural layout design found by improved GA

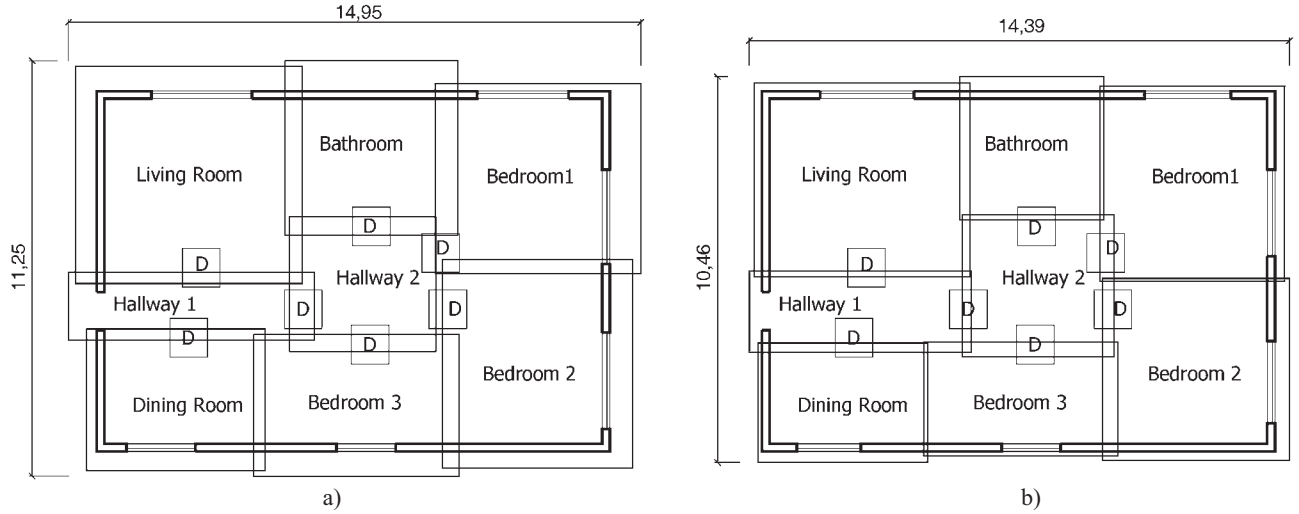
dow constraints (20) and constraints allowing to force a Unit to a particular wall (17), minimum area (18) and minimum ratio constraints (19) and others constraints are automatically added. Windows height can be fixed for each Unit, and all windows are allocated along external walls.

Next, design objectives are optimised. In this study, we have minimised annual heating cost, lighting cost, wasted space, doorways and hallways.

The experiments have been performed with 94 variables, 156 constraints, a population size of 940, generation number of 800, crossover probability Pc = 0,6, mutation probability Pm = 0,125 and evolutionary direction operator probability Ped = 0,6.

In this study, the GA-based optimization programme was run 5 times. The initial population of solutions is generated randomly with no feasible initial starting point. Selection evaluates candidate solutions, while crossover and mutation synthesise new solutions. Intermediate results of performance of improved genetic algorithm after 300 and 650 iterations are shown in Fig 11.

The optimal architectural layout design was found by improved GA and it is shown in Fig 10.



a) b)

Fig 11. Two intermediate architectural layouts generated by improved genetic algorithm a) after 300 iterations; b) after 650 iterations

The results of the performance of standard genetic algorithm and improved genetic algorithm are given in Table 2.

Table 2. Results of performance of standard GA and improved GA

Genetic algorithm	*Worst*	*Average*	*Best*	*Time, s*
Standard GA	742	660	609	188
Improved GA	708	600	550	156

In Table 2 the best solution is the optimal solution found by standard GA and improved GA after a computational time of 188 s and 156 s respectively. The average result can be evaluated as division of sum of objective function values by population size.

It is not difficult to observe that the improved genetic algorithm have better rate of convergence and can find better solution than standard GA in solving architectural layout optimization problem.

7. Conclusions

Though GA is superior to the traditional methods in solving the complex architectural conceptual layout design problem, the rate of convergence is still far from satisfactory. In this paper an improved GA is proposed, analysed in comparison with standard GA in the convergence performances. Applied genetic algorithm for present layout problem adopts a new evolutionary direction operator that allows some offspring to be generated referencing their parents evolving tendencies.

The experiment results demonstrate that the improved GA has better performance not only in the rate of convergence but also for the solution quality to solve the conceptual architectural layout problem.

References

1. Hsu, W.; Liu, B. Conceptual design: issue and challenges. *Computer-aided design*, 2000, Vol 32, No 14, p. 849–850.
2. O'Sullivan, B. Constraint-aided conceptual design. PhD thesis, Dept of Computer Science, University College Cork, Ireland, 1999.
3. Zavadskas, E. K.; Kaklauskas, A.; Kvederytė, N. Multivariant Design and Multiple Criteria Analysis of a Building Life Cycle. *Informatica*, 2001, Vol 12, No 1, p. 169–188.
4. Zavadskas, E. K.; Antuchevičienė, J. Evaluation of buildings redevelopment alternatives with an emphasis on the multipartite sustainability. *International Journal of Strategic Property Management*, 2004, Vol 8, No 2, p. 121–128.
5. Zavadskas, E. K.; Kaklauskas, A.; Gulbinas, A. Multiple criteria decision support web-based system for building refurbishment. *Journal of Civil Engineering and Management*, 2004, Vol 10, No 1, p. 77–85.
6. Vilutienė, T.; Zavadskas, E. K. The application of multicriteria analysis to decision support for the facility management of a residential district. *Journal of Civil Engineering and Management*, 2003, Vol 9, No 4, p. 241–252.
7. Michalek, J. J.; Choudhary, R.; Papalambros, P. Y. Architectural layout design optimization. *Engineering optimization*, 2002, Vol 34, No 5, p. 461–484.
8. Garey, M. R.; Johnson, D. S. Computers and intractability: A guide to the theory of NP-completeness. New York: W. H. Freeman and Company, 1979.
9. Szuba, J.; Borkowski, A. Graph transformations in architectural design. *Computer assisted mechanics and engineering sciences* (CAMES), 2003, No 10, p. 93–109.
10. Buckley, M. J.; Fertig, K. W.; Smith, D. E. Design sheet: An environment for facilitating flexible trade studies during conceptual design. In AIAA 92-1191 aerospace design conference, February 1992, Irvine, California.
11. Bowen, J.; Bahler, D. Frames, quantification, perspectives and negotiation in constraint networks in life-cycle engineering. *International journal for artificial intelligence in engineering*, 1992, Vol 7, p. 199–226.
12. Iivonen, H.; Riitahuhta, A. Case-based reasoning in conceptual design. In: Proceedings of the tenth CIM-Europe annual conference, Copenhagen, Denmark, Oct 1994, Vol 5, p. 307–314.
13. Sabouni, A. R.; Al-Mourad, O. M. Qualitative knowledge-based approach for preliminary design. *Artificial Intelligence and engineering*, 1997, Vol 11, p. 143–154.
14. Sun, K.; Faltings, B. Supporting creative mechanical design. In: Artificial intelligence in design, 1994, p. 39–56.
15. Cvetkovic, D.; Parmee, I. C. Genetic algorithms based systems for conceptual engineering design, ICED 99, Munich, August 24–26, 1999.
16. Lin, J. J. Constructing an intelligent conceptual design system using genetic algorithm. *Journal of materials processing technology*, Vol 140, 2003, p. 95–99.
17. Miles, J. C.; Sisk, G. M.; Moore, C. J. The conceptual design of commercial buildings using a genetic algorithm. *Computers and structures*, 2001, Vol 79, p. 1583–1592.
18. Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. Optimization by simulated annealing. *Science*, 1983, Vol 220, No 4598, p. 671–679.
19. Cagan, J.; Degentesh, D.; Su Yin S. A simulated annealing-based algorithm using hierarchical models for general three-dimensional component layout. *Computer-aided design*, 1997, Vol 30, No 10, p. 781–790.
20. Goldberg, D. E. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.
21. Rebaudengo, M.; Sonza Reorda, M. Flooplan area optimization using genetic algorithms. In: Proceedings Fourth Great Lakes Symposium on VLSI, March 1995, p. 22–25.
22. Gero, J. S.; Kazakov, V. Evolving building blocks for genetic algorithms using genetic engineering. In: Proceedings of the IEEE conference on evolutionary computing, 1995, p. 340–345.
23. Construction technical regulatory law (Statybos techninis reglamentas), STR 2.09.04:2002. Vilnius, 2002 (in Lithuanian).
24. Rebaudengo, M.; Sonza Reorda, M. GALLO: A genetic algorithm for floorplan area optimization. *Computer-aided design*, 1996, Vol 15, No 8, p. 943–951.
25. Pankrašovaitė, I.; Baušys, R. Analysis of genetic algorithms for floorplanning problem. In 8*th* international conference on mathematical modelling and analysis, Abstracts, May 28–31, 2003, Trakai, Lithuania, p. 9.
26. Pankrašovaitė, I.; Baušys, R. Analysis of improved genetic algorithm for optimization of architectural layout. In: Proceedings of conference on Information Technologies, 2004 01 28–29, Kaunas, Lithuania, p. 78–82 (in Lithuanian).

ARCHITEKTŪRINIO PATALPŲ IŠDĖSTYMO UŽDAVINIO OPTIMIZAVIMAS TAIKANT PAGERINTĄ GENETINĮ ALGORITMĄ

R. Baušys, I. Pankrašovaitė

Santrauka

Architektūrinio patalpų išdėstymo tikslas – rasti kiekvienos patalpos parametrus, formą bei patalpų konfigūraciją, kurie minimizuotų bendrą plotą ir tenkintų visus apribojimus. Įvertinti ir kiti architektūriniai kriterijai, tokie kaip apšvietimo ir šildymo sąnaudų mažinimas.

Toks uždavinys priklauso kombinatorinių uždavinių klasei, kuriems spręsti dažniausiai taikomi genetiniai algoritmai. Genetiniai algoritmai, skirtingai negu kiti optimizavimo metodai, pasižymi globaliu ieškojimo būdu ir gebėjimu operuoti visa sprendinių populiacija. Tačiau genetinių algoritmų efektyvumas ir konvergavimo greitis nėra pakankami. Siekiant pagerinti genetinį algoritmą, siūloma taikyti evoliucinį krypties operatorių, kuris nesiremia informacija apie gradientų skaičiavimus. Taip sumažinamos laiko sąnaudos, reikalingos optimaliam patalpų išdėstymo uždavinio sprendiniui rasti, ir pagerėja optimalaus sprendinio kokybė.

Raktažodžiai: architektūrinis patalpų išdėstymas, kambarių išdėstymas, konceptualus projektavimas, optimizavimas, genetiniai algoritmai.

Romualdas BAUŠYS. Prof Dr Habil. Head of Dept of Graphical Systems, Vilnius Gediminas Technical University, Lithuania. Professor, 2001: Doctor Habil (technological sciences), 2000; Assoc Prof, 1996; Doctor (technical sciences), 1989. Research interests: computational mechanics, spatial decision support technologies, image processing technologies, analysis and design of engineering information systems.

Ina PANKRAŠOVAITĖ. MSc, PhD student in the Dept of Graphical Systems, Vilnius Gediminas Technical University, Lithuania. Research interests: application of genetic algorithms in computer-aided design.