Optimizing Architectural Layout Design via Mixed Integer Programming

KEATRUANGKAMALA Kamol[1] and SINAPIROMSARAN Krung[2]
[1] Faculty of Architecture, Rangsit University, Thailand
[2] Faculty of Science, Chulalongkorn University, Thailand

Keywords: layout design, linear programming, mixed integer programming, optimization

Abstract: For many decades, solving the optimal architectural layout design is unattainable for the reasonable problem sizes. Architects have to settle for acceptable layouts instead of the favourable optimal solution. With today technologies, various optimization techniques have been used to alleviate the optimal search according to diversified goals. This paper formulates the optimal architectural layout design as the multiobjective mixed integer programming model solved by the MIP solver. The main idea is to capture functional constraints, dimensional constraints and the objective function using only linear formulae with binary variables. Functional constraints are the connectivities, the unused grid cells, the fixed room location, the boundary and the fixed border location while dimension constraints are the non-intersecting, the overlapping, the length and the ratio constraints. The objective function is designed to minimize the absolute distance among rooms and maximize room spaces. Due to the nonlinearity of area computation, the linear approximation of width and height constraints have been utilized. Architects can control these different objectives within the model. By specifying the rigid restriction and the time limits, the problem can be solved within a reasonable amount of time.

1 INTRODUCTION

Finding the optimum architectural layout design is a challenging problem to many architects and researchers from the past decade. The problem involves both quantifiable and qualifiable goals, preferences, and constraints. Particularly, aesthetic and other subjective aspects of design are difficult to describe using the logical expression or the mathematical formulae. Many attempts have been used to deal with this problem such as the wall representation (Flemming 1978), non-linear programming (Imam and Mir 1989) and the evolutionary approach (Michalek and Papalambros 2002). There are various difficulties with each approach. The wall representation uses the special data structure to generate the linear programming subproblem which requires a specific algorithm. The nonlinear programming approach guarantees only local optimal. The evolutionary method can only guarantee the convergence with a long running time (Jo and Gero 1998).

We propose the use of the mixed integer programming model (MIP) (Linderoth and Savelsbergh 1999) to find the optimal architectural layout design. Our approach guarantee the global optimal solution if the MIP solver stops normally. Even though, the architectural space layout design problem is considered to be ill-defined (Yoon 1992). The automated architectural layout planning must deal with a large set of possible solutions (Scott and Donald 1999) that cannot be solved exhaustively for reasonably-sized problems. Therefore, we reduce the search space by allowing architect to specify additional reduction constraints such as the fixed room location, the unused grid cells, the fixed border location and the favorable choice of the nearest room to the top left corner. Furthermore, the requirement of rectangular boundary has not been imposed.

We implement the software that utilizes the graphic user interface (GUI) running on the Windows operating system. The GNU Linear Programming Kit (GLPK) solver is hooked up to automatically solve the given architectural layout model. Architects can request the drawing presentation of the global optimal solution or save it as the DXF format file to use with their CAD software. In the next section, we identify the model variables, parameters and constraints together with the objective functions. After the model has been completed, we tested our software with the simulated examples using only 4, 5, 6, 7, 8 and 10 rooms in the experiment section.

2 OPTIMIZATION OF GEOMETRY

The architectural layout design problem in this research is formulated based on the grid system. Thus, coordinates and dimension are used as the design variables of the problem (Li, Frazer and Tang 2000).

2.1 Design Variables and Parameters

For each room *i* in figure 1(a), four basic variables are formulated as the coordinated system and the origin point is placed on the top left corner.

x_i = X coordinate of the top left corner of the room *i*.

y_i = Y coordinate of the top left corner of the room *i*.

w_i = the horizontal width of the room *i*.

h_i = the vertical height of the room *i*.

Additionally, two basic parameters are width and height of the boundary area which are represented by *W* and *H*, respectively. We control the width and the height of the design variables using the lower and upper limits, $w_{min,i}$, $w_{max,i}$, $h_{min,i}$, $h_{max,i}$, T_{ij} and *R* where $w_{min,i}$ and $w_{max,i}$ are the minimal and the maximal width of room *i* while $h_{min,i}$ and $h_{max,i}$ are the minimal and maximal height of room *i*, respectively. T_{ij} is the minimal contact length between room *i* and *j* and *R* is the room ratio, see Figure 1(b).

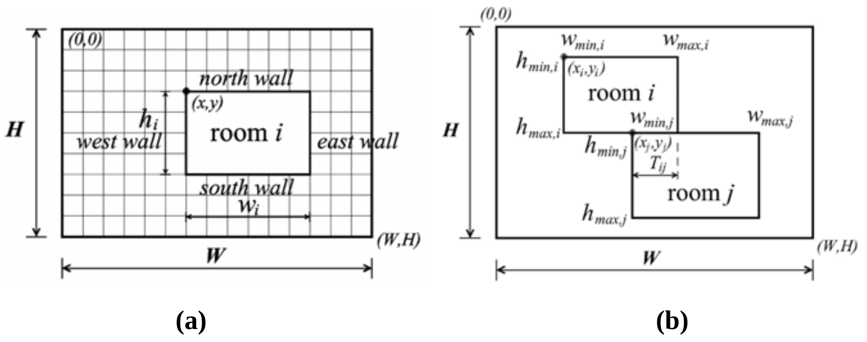


Figure 1 (a) Basic design variables and parameters and (b) Minimal and maximal dimension of room geometry

2.2 Objective of Problem

The multiobjective function of the optimization model in this research composes of minimizing the absolute room distance and maximizing room area. In order to calculate the room area, we use the linear approximation by maximizing the minimal width and height of each room. Moreover, each objective functions are controlled by different nonnegative weighted values. In order to generate multiple optimal distance and optimal area, architect can select the i^{th} room to be fixed at the top left corner coordinate. This function is shown below. For fixed i^{th},

$$\textit{minimize } (u_{i,1}(x_i + y_i) + u_{i,2}\Sigma \textit{ absolute distance} - u_{i,3}\Sigma \textit{ approximate area}) \quad (1)$$

where $u_{i,1}$ is the weight of the i^{th} room positioned to the nearest top left corner, $u_{i,2}$ is the weight of the total absolute distance and $u_{i,3}$ is the weight of the total approximated area. If the architect prefers the large area then the weighted sum of $u_{i,3}$ is set larger than $u_{i,2}$. Otherwise, if the architect prefers a short distance between rooms then $u_{i,2}$ is set larger than $u_{i,3}$.

2.3 Layout Design Constraints

In this research, two kinds of constraints are considered, the functional constraint and the dimensional constraint. The functional constraint determines the placement of all rooms while the dimensional constraint forces room dimensions.

2.3.1 Functional Constraints

Functional constraints determine the placement location of rooms according to the architectural requirements. They can be described as follows.

Connectivity constraint explains the relationship between different rooms (Medjodoub and Yannon 2000). To ensure room attachment for the purpose of the access way, we design two binary variables (p_{ij}, q_{ij}) which are illustrated as follows:

$$x_i + w_i \geq x_j - W*(p_{ij} + q_{ij}) \qquad \text{i to the left of j} \quad ,p_{ij} = 0, q_{ij} = 0 \qquad (2)$$

$$y_j + h_j \geq y_i - H*(1 + p_{ij} - q_{ij}) \qquad \text{i above j} \quad ,p_{ij} = 0, q_{ij} = 1 \qquad (3)$$

$$x_j + w_j \geq x_i - W*(1 - p_{ij} + q_{ij}) \qquad \text{i to the right of j} \quad ,p_{ij} = 1, q_{ij} = 0 \qquad (4)$$

$$y_i + h_i \geq y_j - H*(2 - p_{ij} - q_{ij}) \qquad \text{i below j} \quad ,p_{ij} = 1, q_{ij} = 1 \qquad (5)$$

Fixed position constraint determines the room positioning in a space. In practical design, this constraint helps an architect to secure the room location in the design:

$$x_i = \text{fixed x coordinate}, \quad y_i = \text{fixed y coordinate} \qquad (6)$$

Unused grid cell constraint determines the area that is unusable. This constraint helps an architect designing various orthogonal boundary shapes. We use two binary variables (s_{ik}, t_{ik}) to identify the location of unused grid cell, k^{th}.

$$x_i \geq x_{u,k} + 1 - W*(s_{ik} + t_{ik}) \qquad \text{unused space to left of i} \qquad (7)$$

$$x_{u,k} \geq x_i + w_i - W*(1 + s_{ik} - t_{ik}) \qquad \text{unused space to right of i} \qquad (8)$$

$$y_i \geq y_{u,k} + 1 - H*(1 - s_{ik} + t_{ik}) \qquad \text{unused space to top of i} \qquad (9)$$

$$y_{u,k} \geq y_i + h_i - H*(2 - s_{ik} - t_{ik}) \qquad \text{unused space to bottom of i} \qquad (10)$$

where $x_{u,k}$ $y_{u,k}$ are unused positions in x,y coordinate of the unused k^{th} cell.

Boundary constraint forces a room to be inside a boundary:

$$x_i + w_i \leq W, \quad y_i + h_i \leq H \qquad \text{a room within the boundary} \qquad (11)$$

Fixed border constraint addresses the absolute placement of the room. This constraint is divided into four types: north, south, east and west. For example, a room is positioned to the “absolutely north” if its touch the top border:

$$y_i = 0, \; y_i + h_i = H \qquad \text{touch the north and the south border} \qquad (12)$$

$$x_i + w_i = W, \; x_i = 0 \qquad \text{touch the east and west border} \qquad (13)$$

2.3.2 Dimensional Constraints

Dimensional constraints determine the adjustment of room geometry according to the proportional requirements.

Non-intersecting constraint prevents two rooms from occupying the same space (Medjodoub and Yannon 2000). We use the same two binary variables p_{ij} and q_{ij} to protect room collision. These requirements can be illustrated as:

$$x_i + w_i \le x_j + W*(p_{ij} + q_{ij}) \quad \textit{i to the left of j} \quad ,p_{ij} = 0, q_{ij} = 0 \quad (14)$$

$$y_j + h_j \le y_i + H*(1 + p_{ij} - q_{ij}) \quad \textit{i above j} \quad ,p_{ij} = 0, q_{ij} = 1 \quad (15)$$

$$x_j + w_j \le x_i + W*(1 - p_{ij} + q_{ij}) \quad \textit{i to the right of j} \quad ,p_{ij} = 1, q_{ij} = 0 \quad (16)$$

$$y_i + h_i \le y_j + H*(2 - p_{ij} - q_{ij}) \quad \textit{i below j} \quad ,p_{ij} = 1, q_{ij} = 1 \quad (17)$$

Overlapping constraint forces the minimal contact length between two connected rooms. Two rooms are touching with each other with the minimal contact length defined by the value (T_{ij}). For example, the junction between a room *i* and room *j* must be wide enough to accommodate an access way:

$$0.5*(w_i + w_j) > T_{ij} + (x_j - x_i) - W*(p_{ij} + q_{ij}) \quad \textit{i to the left j} \quad (18)$$

$$0.5*(h_i + h_j) \ge T_{ij} + (y_j - y_i) - H*(2 - p_{ij} - q_{ij}) \quad \textit{i to the top of j} \quad (19)$$

$$0.5*(w_i + w_j) \ge T_{ij} + (x_i - x_j) - W*(1 - p_{ij} + q_{ij}) \quad \textit{i to the right of j} \quad (20)$$

$$0.5*(h_i + h_j) \ge T_{ij} + (y_i - y_j) - H*(1 + p_{ij} - q_{ij}) \quad \textit{i to the bottom of j} \quad (21)$$

Length constraint is a minimal or maximal length of the bounded size of each room. A certain room is limited to suitable dimensions between the horizontal range of $w_{min,i}$,$w_{max,i}$ and the vertical range of $h_{min,i}$,$h_{max,i}$, respectively:

$$w_{min,i} \le w_i \le w_{max,i} \,,\ h_{min,i} \le h_i \le h_{max,i} \quad \textit{range of width and height of room i} \ (22)$$

Ratio constraint restricts the length between horizontal and vertical dimension. This constraint prevents the long and narrow shape of a room. A binary variable (r_i) is used to select the constraint satisfying horizontal and vertical ratio:

$$w_i + r_i*(W+H) \ge R*h_i \,,\ h_i + (1 - r_i)*(W+H) \ge R*w_i \quad \textit{horizontal and vertical ratio} \quad (23)$$

3 SOFTWARE DEVELOPMENT

The software uses the non-commercial linear programming and mixed integer programming solver, GLPK (GNU Linear Programming Kit) that was developed by Moscow Aviation Institute (Russia). GLPK software can solve a large-scale linear programming problem and mixed integer linear programming based on the branch and bound technique.

We develop a user-friendly interface for this software to help architect identifying the layout requirement graphically. After the architects input a functional diagram via the interface as in Figure 2(a), the problem is then converted to a GNU MathProg model that is supported by GLPK. The result of the computation as in Figure 2(b) can be presented graphically as layout drawings, see Figure 2(c). Furthermore, the solution can be exported as DXF file format that is widely supported by CAD software, see Figure 2(d).

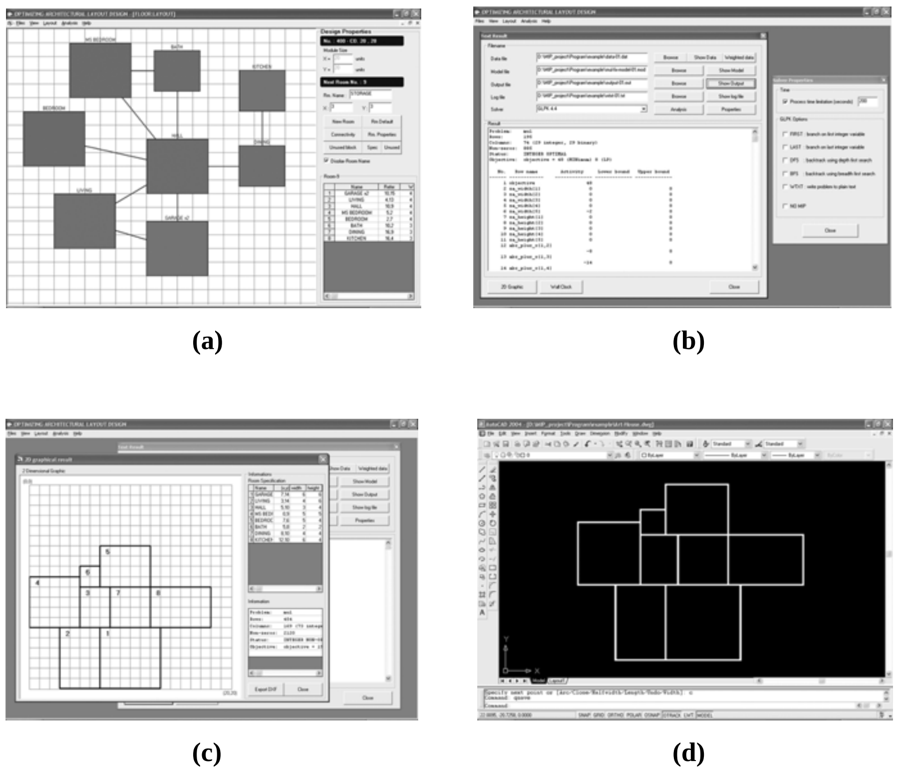


Figure 2 Software interface (a) for functional diagram, (b) and (c) for text and graphical output. (d) shows the export DXF file from CAD software

4 EXPERIMENTS

This research presented the experimental results from the GLPK solver. All experiments were carried on a PC computer using Pentium 1 GHz and 256 MB of memories. We used 4, 5, 6, 7, 8 and 10 rooms for our experiments. Each run was performed using 15 configurations. More than 100 examples from six room sizes were tested.

4.1 Computation time of GLPK

In order to measure the performance, we collected computation time and global solutions. Our experiments are divided into three cases and described as follows.

Case 1 composes of 36 configurations with the equal room proportions. Each configuration contains 4 different connection patterns. Each pattern contains 6 experiments. The average time is reported in Figure 3. Case 2 composes of 36 configurations with different room proportions. We perform the same experiment as in the case a. Case 3 composes of 36 configurations with different of room width and height ratios. We perform the same experiment as in the case a.

The details of each pattern are as follows. Pattern A is the consecutive connection from the first room to the last room. Pattern B is the one room connected to the remaining rooms. Pattern C is the consecutive connection among odd-room numbers and the consecutive connection among even-room numbers and pattern D is the same as pattern C except one odd-room is connected to one even-rooms.

Figure 3 shows the faster computation times of the case 2 comparing with the case 1 about 50 percents. For the case 3, the room ratio shows no different computation time with respect to case 1.

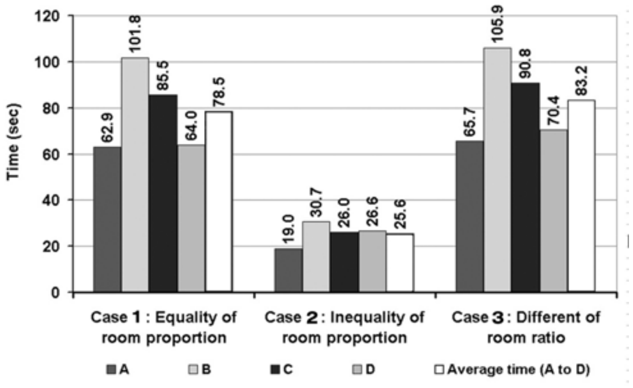
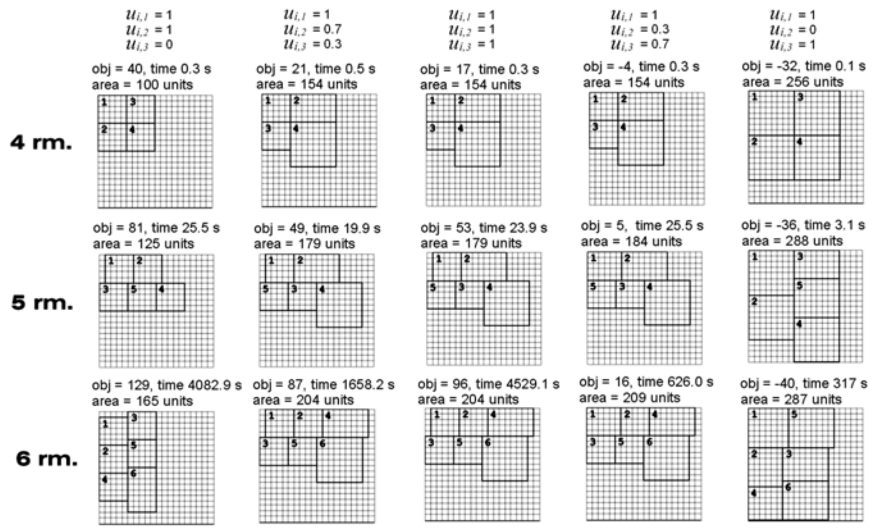


Figure 3 An average summation of room sizes 4 to 10 with 90 run in the different connection patterns of A, B, C and D

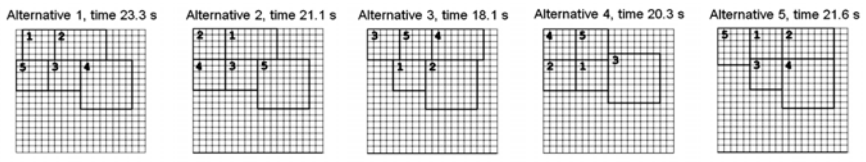
4.2 Layout Design with Multiobjectives

In practice, the balanced between quantifiable and unquantifiable aspects of architectural goals should be considered. In our model, we allow the architect to select his/her alternatives by using the weight values. $u_{i,1}$ is the weight of the room position, $u_{i,2}$ is the weight of the total room distance and $u_{i,3}$ in the objective function (see Figure 4). Furthermore, architect can select alternative global optimal solutions based on the room positioning to the top left corner which is shown in Figure 5.



$u_{i,1}$ = 1, $u_{i,2}$ = 1, $u_{i,3}$ = 0
$u_{i,1}$ = 1, $u_{i,2}$ = 0.7, $u_{i,3}$ = 0.3
$u_{i,1}$ = 1, $u_{i,2}$ = 1, $u_{i,3}$ = 1
$u_{i,1}$ = 1, $u_{i,2}$ = 0.3, $u_{i,3}$ = 0.7
$u_{i,1}$ = 1, $u_{i,2}$ = 0, $u_{i,3}$ = 1
4 rm.
obj = 40, time 0.3 s area = 100 units
obj = 21, time 0.5 s area = 154 units
obj = 17, time 0.3 s area = 154 units
obj = -4, time 0.3 s area = 154 units
obj = -32, time 0.1 s area = 256 units
5 rm.
obj = 81, time 25.5 s area = 125 units
obj = 49, time 19.9 s area = 179 units
obj = 53, time 23.9 s area = 179 units
obj = 5, time 25.5 s area = 184 units
obj = -36, time 3.1 s area = 288 units
6 rm.
obj = 129, time 4082.9 s area = 165 units
obj = 87, time 1658.2 s area = 204 units
obj = 96, time 4529.1 s area = 204 units
obj = 16, time 626.0 s area = 209 units
obj = -40, time 317 s area = 287 units

Figure 4 Shows the multiobjectives of 4, 5 and 6 rooms where $u_{i,1}$, $u_{i,2}$ and $u_{i,3}$ is the weight of the total room area for the fixed room *i*



Alternative 1, time 23.3 s
Alternative 2, time 21.1 s
Alternative 3, time 18.1 s
Alternative 4, time 20.3 s
Alternative 5, time 21.6 s

Figure 5 Five alternative global solutions of 5 room configurations

4.3 Comparison between MIP and Nonlinear Programming

In order to compare the efficient of the MIP and nonlinear model, we select the GLPK solver to solve MIP, the CPLEX solver to solve MIP and the DICOPT solver to solve MINLP (Mixed Integer Nonlinear Programming). Table 1 shows the average comparison with number of variables and objective values.

Table 1 The average comparison between MIP and Non-linear programming

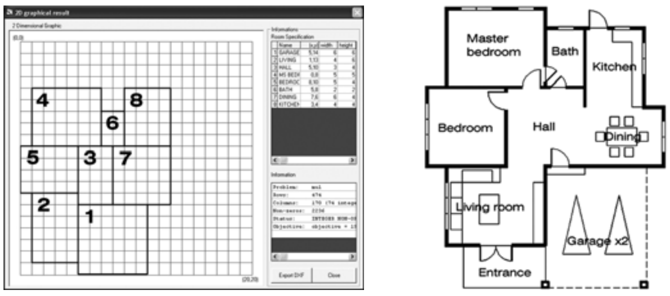
Room Sized	Time (sec.)			Variables (non-zero)			Objective value		
	MIP	MIP	MINLP	MIP	MIP	MINLP	MIP	MIP	MINLP
	GLPK	CPLEX	DICOPT	GLPK	CPLEX	DICOPT	GLPK	CPLEX	DICOPT
4 rm.	1.0	1.0	1.0	594	593	517	32	32	32
5 rm.	69.3	17.4	56.2	856	855	725	80	80	126
6 rm.	430	135	957	1152	1151	935	125	125	183

4.4 Practical Case Study

A simple practical floor plan was solved by the architect and by our software with the time limit of 1000 seconds. The problem composes of eight rooms in 20x20 m. The detail parameters are shown in Table 2 and the result is shown in Figure 6.

Table 2 Room Dimensional Constraints (Scale in Meter)

Room Name	Min width	Max width	Min height	Max height	Ratio	Connect	Fixed wall
1. Garage x2	5	6	6	6	0.5	2,3	South
2. Living	4	6	5	6	0.5	1,3	None
3. Hall	3	6	3	6	0.5	1,2,4,5,6,7	None
4. Ms Bedroom	5	6	5	6	0.5	3,6	None
5. Bedroom	4	5	4	5	0.5	3	None
6. Bath	2	3	2	3	0.5	3,4	None
7. Dining	5	6	5	6	0.5	3,8	None
8. Kitchen	4	6	4	6	0.5	7	None



(a) (b)

Figure 6 The comparison shows the similar result between the MIP and the architect where (a) is solved by MIP and (b) is designed by the architect

5 CONCLUSION

From our experiment, the MIP and nonlinear programming solver perform comparatively well with respect to the same architectural designs. For a larger problem size, the fixed position constraints, unused grid cells and the fixed boundary help reduce the computation time considerably. This shows the feasibility of using the MIP model under the constrained space. Moreover, we can select the multiple global solutions by changing the room positioning to the top left corner in the objective function. Our software can be used to generate the optimal architectural layout design alternatives according to architect's preferences. For larger architectural layout design problem, the commercial and parallel MIP solvers are required. Our current use of the MIP solver without the domain expert causes the software to perform very slowly for 10 – 20 room configurations. The machine learning methodology should be adopted to speed up the computational time while the Pareto scheme and Flemming wall's representation should be considered to reduce the possibly search space.

REFERENCES

Flemming, U. 1978. Representation and generation of rectangular dissections. *Annual ACM IEEE Design Automation Conference* 15: 138-144.

Imam, M.H., and M. Mir. 1989. Nonlinear programming approach to automated topology optimization. *Computer-aided Design* 21(2):107-115.

Jo, J.H., and J.S. Gero. 1998. Space layout planning using an evolutionary approach. *Artificial Intelligence in Engineering* 12(3): 149-162.

Li, S.P., J.H. Frazer, and M.X. Tang. 2000. A constraint based generative system for floor layouts. In *CAADRIA 2000* [conference proceedings], eds. Ben-Kiang Tan, Milton Tan and Yunn-Chii Wong*:* 441-450. Singapore: CASA.

Linderoth, J., and M.W.P. Savelsbergh. 1999. A computational study of search strategies for mixed integer programming. *INFORMS J. on Computing* 11: 173-187.

Medjodoub, B., and B. Yannon. 2000. Separating Topology and Geometry in space planning. *Computer-Aided Design* 32: 39-61.

Michalek, J., and P.Y. Papalambros. 2002. Interactive layout design optimization. *Engineering Optimization* 34(5): 461-184.

Scott, A.A., and D.H. House. 1999. Making Design Come Alive: Using Physically Based Modelling Techniques in Space Layout Planning. In *Computers in Building: Proceedings of the CAADfutures '99 Conference*, eds. G. Augenbroe, and Ch. Eastman: 245-262. Dordrecht: Kluwer.