A CONSTRAINT BASED GENERATIVE SYSTEM FOR FLOOR LAYOUTS

SIU-PAN LI, JOHN H. FRAZER, MING-XI TANG
School of Design
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
sdli@polyu.edu.hk, sdfrazer@polyu.edu.hk, sdtang@polyu.edu.hk

Abstract. This paper presents the current study of using a constraint based approach to solve floor layout problems. Nonlinear programming technique is used for the solution searching. This paper presents the authors' attempt to improve the nonlinear programming techniques for floor layout problems. Unlike most nonlinear programming systems, multiple optimized solutions can be provided with this system. The process of solving a layout problem, from constraint specification to solution searching, is described in detail. A case study is given in the last section before the conclusions to illustrate how the proposed model works.

1. Introduction

Other than building form, floor layout is another important consideration in architectural designs. Floor plans must conform to functional requirements, physical limitations and many other constraints, like circulation, evacuation, costs, building regulations, etc. Many attempts towards the automation of spatial layout planning have been reported using various techniques. Expert systems (Flemming, 1988), evolutionary approach (Hamamoto, *et al.*, 1999; Kochhar and Heragu, 1999; Rosenman, 1996; Hower, *et al.*, 1996, Jo and Gero, 1995) and constraint-based approach (Baykan and Fox, 1991; Lottaz, *et al.*, 1998; Medjdoub and Yannou, 1998; Meller, *et al.*, 1999) are some commonly used methods. Expert systems present many disadvantages: incompleteness, inconsistency and slow response (Medjdoub and Yannou, 1998) whist evoluationary and constraint-based approach have both their weaknesses and advantages (Flemming, *et al.*, 1992). Evolutionary approach has advantages in adaptation, capability in exploration and ability to handle problems without complete mathematical models. However, being a generate-and-test methodology, evolutionary approach has a slow response. Although a complete formal mathematical model is not required, the generation and selection methods must be chosen with care. Constraint based approach, on the contrary, is a direct search method which searches through the domains for the solutions according to the problem specific mathematical models. It is straightforward in model formulation, and fast in response.

Solutions for floor layout design using constraint based approach have been presented using techniques such as exhaustive enumeration, branch and bound methods, and other heuristic procedures (Imam and Mir, 1989). These heuristic methods usually limit the solution search in discrete space only, and thus approximation of design variables is needed. In response to this, a constraint based system on the "continuous" space is proposed.

This paper presents an on-going research that attempts to solve the layout planning problem using nonlinear programming. As the common practice exercised by many automatic layout planning systems, a 2-D placement problem of rectangular blocks within a rectangular boundary is considered. A rectangular block can represents a room, a corridor, a staircase, or an external void, etc. Position and size of each block is determined by four parameters: x-, y- coordinates of the centroid, length and width where the length and the width are the length of the block in horizontal and vertical directions respectively.

As numerous researchers see constraint specification and satisfaction as the key issues in the design process, this paper focuses on these two aspects. The process of solving a layout planning problem, from constraint specification to solutions searching, is described in detail.

2. Specification of Constraint Sets

Design constraints are functional and physical requirements for a design problem. This section introduces the process of converting design constraints into mathematical models. Two kinds of constraints of layout planning problems are considered: dimensional constraints and functional constraints. Dimensional constraints limit the size of each block whereas the functional constraints determine the placement of each block according to functional requirements.

2.1. VARIABLES OF PROBLEM

The floor layout problem, as described above, tackled in this system is characterized by the following variables

x_i = x-coordinate of the centroid of Block I
y_i = y-coordinate of the centroid of Block I
L_i = length of Block I in horizontal direction
W_i = length of Block I in vertical direction
L_{min} = x-coordinate of the lower left corner of the boundary
W_{min} = y-coordinate of the lower left corner of the boundary

L_{max} = x-coordinate of the upper right corner of the boundary
W_{max} = y-coordinate of the upper right corner of the boundary
P_{ij} = minimum length of contact between Block I and Block J.

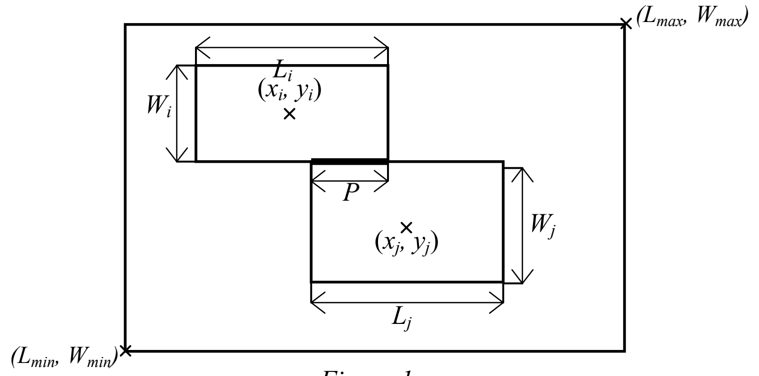

Figure 1.

2.2. DIMENSIONAL CONSTRAINTS

Owing to the implicit assumption that all element blocks must be placed inside a rectangular boundary without overlapping, the following constraints are imposed:

2.2.1. Non-overlapping Constraint

The *non-overlapping constraint* limits the placement of the blocks so that one block cannot overlap with the others. This is compulsory for all cases. In order to guarantee that two blocks are separated, their centers must be separated farther than half the sum of the lengths of their sides in any direction. These requirements can be expressed mathematically as:

$$\forall_{i<j} \left\{ \left|x_i - x_j\right| \geq (L_i + L_j)/2 \quad \text{OR} \quad \left|y_i - y_j\right| \geq \left(W_i + W_j\right)/2 \right\}$$

2.2.2. Boundary Constraints

The *boundary constraints* impose that all blocks must be placed inside a rectangular boundary. This is another compulsory condition. It can be enforced by ensuring all the corners of the blocks are placed within the range bounded by the lower left and upper right corners of the boundary. Hence they can be expressed mathematically as the following four simultaneous conditions:

$$\forall_i \ (x_i + L_i / 2 \leq L_{\max}) \qquad \forall_i \ (x_i - L_i / 2 \geq L_{\min})$$

$$\forall_i \ (y_i + W_i / 2 \le W_{\max}) \qquad \forall_i \ (y_i - W_i / 2 \ge W_{\min})$$

2.2.3. Minimum/Maximum Length Constraints

The *minimum/maximum length constraints* bound the sizes of all blocks so that the blocks can be limited to suitable dimensions. For instance, the minimum size constraints are usually set for most blocks to prevent them from being diminished to zero dimension. They can be written in the following form:

$$L_i \le L_{i\max} \quad \text{and} \quad L_i \ge L_{i\min} \quad \text{and} \quad W_i \le W_{i\max} \quad \text{and} \quad W_i \ge W_{i\min}$$

2.2.4. Size Constraints

In addition to limiting the length of each side, restricting the area and the aspect ratio of the blocks is another useful mean to control the blocks' dimensions.

2.3. FUNCTIONAL CONSTRAINTS

Functional constraints determine the placement of blocks according to the functional requirements of different components. They can be divided into four groups: *adjacency constraints*, *relative direction constraints*, *absolute direction constraints* and *absolute size/position constraints*.

2.3.1. Adjacency Constraints

Adjacency constraints specify the adjacency relationships between different component blocks. They are the most interesting and the most useful constraints since the concept of adjacency can be widely adopted in design problems. In a building layout problem, one may want to ensure some rooms are adjacent to some other rooms for the sake of evacuation or other reasons.

As all variables in this model are real numbers, it is necessary to define the term of "adjacency" explicitly. A parameter, *length of contact*, is thus introduced. Two blocks are said to be adjacent only if the two blocks are touching with each other with the length of contact not less than a preset value. This parameter may be very useful for floor layout problems. For example, the junction between a room and a corridor must be wide enough to accommodate a doorway.

$$\begin{cases} (L_i + L_j)/2 - |x_i - x_j| \ge P_{ij} \\ (W_i + W_j)/2 - |y_i - y_j| = 0 \end{cases} \quad \text{OR} \quad \begin{cases} (L_i + L_j)/2 - |x_i - x_j| = 0 \\ (W_i + W_j)/2 - |y_i - y_j| \ge P_{ij} \end{cases}$$

2.3.2. Relative Direction Constraints

The *relative direction constraints* specify the relative topological relationships between different blocks. Four types of relative position constraints are provided: *north*, *west*, *south* and *east*. As an example, "Block I is *relatively north to* Block J" implies that the y-value of I's bottom edge must not be less than that of J's top edge. And hence:

$$x_i - L_i / 2 \geq x_j - L_j / 2$$

More than one *relative direction constraints* can be applied to a pair of blocks but contradicting constraints, such as *north* and *south*, *west* and *east*, are prohibited.

2.3.3. Absolute Direction Constraints

Absolute direction constraints address the absolute direction of the component blocks. Similar to the relative ones, these constraints can be divided into four types: *north*, *west*, *south* and *east* meaning that the blocks are positioned on the north, west, south and east sides of the boundary respectively. For example, a block is said to be positioned "*absolutely north*" if its top edge is touching the top edge of the border, that is,

$$y_i + W_i / 2 = W_{\max}$$

2.3.4. Absolute Size/Position Constraints

Besides the directional constraints, *absolute size/position constraints* are some other useful criteria. These constraints can be used to place some fixed objects into the plans, to model unmovable obstacles, or to model irregular boundaries by placing external voids inside the boundaries.

3. Solution Searching

After converting all constraints to mathematical form, it is ready to search for the solutions. In this study a commercial nonlinear optimization tool, LINGO by LINDO Systems (Schrage, 1998), is employed. LINGO's nonlinear solver uses both successive linear programming (SLP) and generalized reduced gradient (GRG) algorithms. Just as all other nonlinear solvers, the global optimum is not guaranteed, but this does not affect the usefulness of this system. As not all knowledge can be transformed into mathematical forms, complete models are not possible. Besides searching for the optimal solutions, design tools need to provide facilities to explore feasible design alternatives. More knowledge about the requirements and definition can be acquired in the process of search (Archea, 1987).

Although mathematical programming approach towards layout planning problems have been attempted many times, only linear programming approach is commonly adopted by researchers (Lottaz, et al., 1998; Li and Hsieh, 1998; Meller, et al., 1999). Nonlinear constraints are approximated by successive linear constraints. In spite of the fact that this can help to find the global optimal solution and to speed up the solution searching process, the problem is usually over-simplified that the nature of the problem is changed. Solution solving using nonlinear programming is rare. It may be due to the fact that discontinuity of first derivatives, conditional constraints and misleading direction of steepest descent (Imam and Mir, 1989) are some common problems encountered by topological optimizations. Some measures are taken in this system to tackle two common functions in the floor layout problems: the *absolute* function and the *OR* operation.

3.1. ABSOLUTE FUNCTION

The following approximation of the *absolute* function was taken by the system:

$$F = |x| \approx \sqrt{x^2 + \tfrac{2}{n}} \qquad \text{where} \quad n >> 0$$

Obviously, F has a discontinuous first derivative at $x = 0$ for the *absolute* function. However, for the approximated function, the first derivative is 0 at $x = 0$ and the second derivative is $\sqrt{\tfrac{n}{2}}$, and thus its first derivative is no longer discontinuous.

3.2. *OR* OPERATION

As it is difficult to include conditional constraints in the simultaneous equations of a mathematical programming system, a convenient function was developed to handle the *OR* operations. The function returns a negative value when one of the parameters is negative.

$$f_{OR}(x,y) = \begin{cases} -xy & \text{if } x < 0 \text{ and } y < 0; \\ xy & \text{otherwise} \end{cases}$$

TABLE 1. Characteristics of the *OR* function.

x	y	f_{OR}
+	+	+
+	-	-
-	+	-
-	-	-

3.3. GENERATIVE CAPABILITY

The generative capability of this system is facilitated by the data initialization capability of LINGO which set different initial values for the search. Considering a simple non-linear function $x \sin 3x$ as an example, although there are several local optima over the range (0, 6) of the function, only one optimum can be found at a time. By giving different initial values to x, different optimal points can be reached. When 0.6 is assigned to x initially, LINGO will propagate to 0.68. If the starting value of x is 2.1, another local optimum at 2.66 can be found. By choosing different starting points for solution searching, the solver may converge to different local optima. It is envisaged that all local optimal points, including the global optimum, can thus be found.

A user-friendly interface is provided in this system for inputting constraints using functional diagrams. This interface is developed on a CAD system, MicroStation® (a trademark of Bentley Systems). After the designer input a functional diagram via the interface, the information of the problem is then transformed and stored in a relational database (implemented using Microsoft Access) automatically. This relational database can be linked to other CAD systems to provide more advanced features, such as the solution solver for this project and other data management. An external program was developed to retrieve the information from the relational database to convert to a LINGO model for solving. Results of the computation are then presented graphically on MicroStation as drawings so that the results can be manipulated or further modified without the need to input the results into a CAD system again.

4. Case Study

A simple floor layout problem is illustrated in this section in order to show how the model works. Six component blocks are placed in an 8m × 10m boundary. They are an *entrance*, a *bathroom*, a *kitchen*, a *living room* and two *bedrooms*. These six components should cover the whole area of the boundary. They are subject to the dimensional requirements in Table II and the functional constraints listed below:

TABLE 2. Dimensional constraints.

Block	*Area Min*	*Area Max*	*Length Min*	*Length Max*	*Width Min*	*Width Max*	*Aspect Ratio*
Corridor (COR)	—	15	1	8	1	10	—
Living (LIV)	15	20	2	6	2	6	[0.5, 2]
Bedroom 1 (BED1)	12	18	1.7	6	1.7	6	[0.5, 2]
Bedroom 2 (BED2)	12	18	1.7	6	1.7	6	[0.5, 2]

Bathroom (BAT)	5	8	1.5	6	1.5	6	—
Kitchen (KIT)	5	8	1.5	6	1.5	6	—

- The *entrance* is on the north of the boundary.
- The *living room* is adjacent to the *entrance* with at least 1.5m contact length.
- The *bedrooms* are either adjacent to the *entrance* or adjacent to the *living room* with at least 1m in contact.
- The *kitchen* and the *bathroom* are adjacent to the *living room* with at least 1m contact length.
- In order to avoid the *bedrooms* and the *living room* from becoming narrow and long, the aspect ratios, in both directions, of these rooms are restricted to a maximum of 2.

Figure 2 shows the functional diagram of this problem. The objective of the problem is to maximize the living space, that is, the total area of the *living room* and the two *bedrooms*.

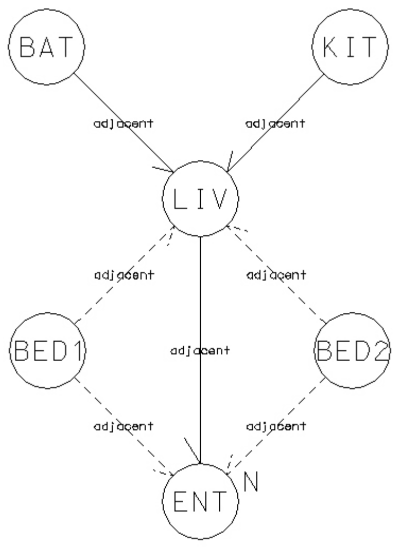

Figure 2. Functional diagram.

Figure 3 shows 10 solutions generated by the program in about 4 minutes. The objective values, shown on the top of each plan, indicate the total area of the living room and the two bedrooms in meter square. Obviously, the larger the objective function value, the better is the solution. From the diagram, it is observed that the program can generate different topological orders with all the constraints satisfied. All the solutions obtained are optimized with respect to each topological arrangement according to the objective function. From Table II, it can be found that the maximum total area of the *living room* and the *bedrooms* is 56 m^2. The system can find not only one optimal solution but several. Besides the optimal solutions, some other sub-optimal solutions are suggested. Although these sub-optimal ones are not the best solutions, they may give designers other inspiration.

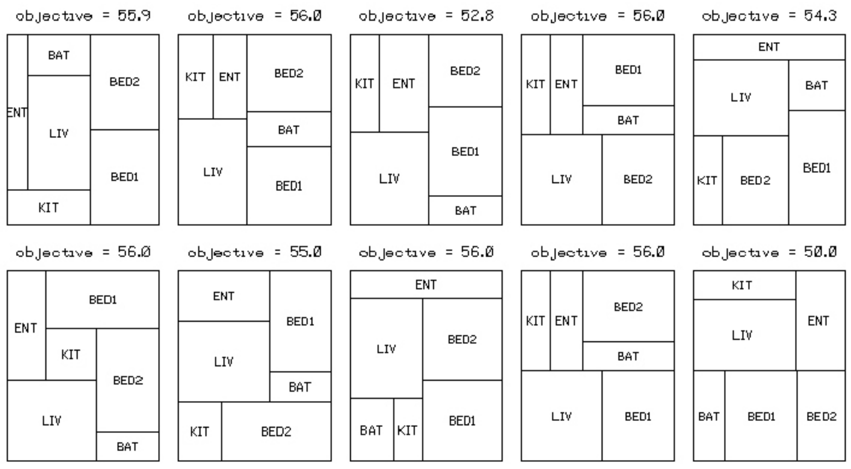

Figure 3.

5. Conclusions

In this paper, a constraint-based system for layout planning is presented. With the nonlinear programming approach, this system can produce fast response. Unlike other nonlinear programming systems which can only suggest single optimal solutions, multiple optimized solutions for a problem can be proposed. Each proposed solution is optimized according to the objective function for each configuration. As realized by many researchers that it is difficult to define all constraints before searching the solutions (Maher, 1990; Gross, *et al.*, 1987), listing different alternatives may help designers to explore the problem. Moreover, as designers are the final judges to decide which solution is the best, it is envisaged that other knowledge that cannot be formulated can also be incorporated in the solutions.

Despite its advantages, this approach has its weakness in solution solving when the number of blocks is large (more than 10). Although the non-smooth *absolute* function was replaced by a smooth function in this system, the user-defined *OR* function created a new non-smooth function. Some discrete variables in the system also introduced difficulties in solution searching. It is expected that the ability of the solution solving will be doubled when a smooth approximation of the *OR* function is created in the near future. More realistic design problems can be handled then.

Due to the similarity between the floor layout problem and other design problems, such as the placement problem in electrical engineering, the facility allocation problem and plant layout design in industrial engineering, the techniques developed in this system can be applied to other fields of design.

Acknowledgments

This project is made possible by funding from the Areas of Strategic Development of the Hong Kong Polytechnic University. It is a part of an on-going research project in developing an Integrated System for Rapid Product Design and Development.

References

Archea, J.: 1987, PUZZLE-MAKING: What architects do when no one is looking, *in* Y.E. Kalay (ed), *Computability of Design*, Wiley-Interscience, New York.

Baykan, C. and Fox, M.: 1991, Constraint Satisfaction Techniques for Spatial Planning, *in* P.J.W. ten Hagen and P.J. Veerkamp (eds), *Intelligent CAD Systems III: Practical experience and evaluation*, Springer-Verlag, Berlin.

Flemming, U.: 1988, A generative expert system for the design of building layouts, *Artificial Intelligence in Engineering: Design,* Elsiever, Amsterdam.

Gross, M., Ervin, S., Anderson, J. and Fleisher, A.: 1987, Designing with constraints, *in* Y.E. Kalay (ed), *Computability of Design*, Wiley-Interscience, New York.

Hamamoto, S., Yih, Y. and Salvendy, G.: 1999, Development and validation of genetic algorithm-based facility layout – a case study in the pharmaceutical industry, *International Journal of Production Research*, **37**(4), 749-768.

Hower, W., Rosendahl, M. and Köstner, D.: 1996, Evolutionary layout design, in J.S. Gero and F. Sudweeks (eds), Artificial Intelligence in Design '96, 663-680.

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

Jo, J.H. and Gero, J.S.: 1995, Space layout planning using an evolutionary approach, *in* M. Tan and R. Teh (eds), *The Global Design Studio, proceedings of the sixth International Conference on Computer Aided Architectural Design Futures*, Singapore, 189-203.

Kochhar, J.S. and Heragu, S.S.: 1999, Facility layout design in a changing environment, *International Journal of Production Research*, **37**(11), 2429-2446.

Li, H.L. and Hsieh, M.C.: 1998, Solving facility-layout optimization problems with clustering constraints, *Environment and Planning B.: Planning and Design*, **25**, 299-308.

Lottaz, C., Stalker, R. and Smith, I.: 1998, Constraint solving and preference activation for interactive design, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **12**, 13-27.

Maher, M.L.: 1990, Process models of design synthesis, *AI Magazine*, Winter 1990, Menlo Park, California.

Medjdoub, B. and Yannou, B.: 1998, Topological enumeration heuristics in constraint-based space layout planning, *in* J.S. Gero and F. Sudweeks (eds), *Artificial Intelligence in Design '98, Kluwer Academic, the Netherlands*, 271-290.

Meller, R.D., Narayanan, V. and Vance, P.H.: 1999, Optimal facility layout design, *Operations Research Letters*, **23**, 117-127.

Schrage, L.: 1998, *Optimization Modeling with LINGO*, LINDO Systems Inc., Chicago.

Rosenman, M.A.: 1996, The generation of form using an evolutionary approach, *in* J.S. Gero and F. Sudweeks (eds), *Artificial Intelligence in Design '96*, 643-662.