การเร่งกระบวนการหาผลเฉลยของการออกแบบการวางผังที่เหมาะที่สุด
โดยใช้ขั้นตอนวิธีทางพันธุกรรม

นางสาวฐิติยา เทพารส

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคณนา ภาควิชาคณิตศาสตร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2552

ACCELERATING THE SOLVING PROCESS OF OPTIMAL LAYOUT DESIGN USING GENETIC ALGORITHM

Miss Thitiya Theparod

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computational Science
Department of Mathematics
Faculty of Science
Chulalongkorn University
Academic Year 2009

Thesis Title ACCELERATING THE SOLVING PROCESS OF OPTIMAL LAYOUT DESIGN USING GENETIC ALGORITHM

By Miss Thitiya Theparod

Field of Study Computational Science

Thesis Advisor Assistant Professor Krung Sinapiromsaran, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

.... Dean of the Faculty of Science

(Professor Supot Hannongbua, Dr. rer. nat.)

THESIS COMMITTEE

.... Chairman

(Associate Professor Pornchai Satravaha, Ph.D.)

.... Thesis Advisor

(Assistant Professor Krung Sinapiromsaran, Ph.D.)

.... Examiner

(Assistant Professor Anusorn Chonwerayuth, Ph.D.)

.... External Examiner

(Kamol Keatruangkamala, Ph.D.)

ฐิติยา เทพารส : การเร่งกระบวนการหาผลเฉลยของการออกแบบการวางผังโดยใช้ขั้นตอนวิธีทางพันธุกรรม (Accelerating the solving process of optimal layout Design using genetic algorithm) อ. ที่ปรึกษา วิทยานิพนธ์หลัก : ผู้ช่วยศาสตราจารย์ ดร. กรุง สินอภิรมย์สราญ, 72หน้า.

การออกแบบการวางผังที่เหมาะที่สุด เป็นขั้นตอนของการออกแบบทางสถาปัตกรรมที่เกี่ยวข้องกับการหาตำแหน่ง และขนาดของห้องที่เหมาะที่สุดที่สอดคล้อง กับเงื่อนไขทางสถาปัตยกรรม ในบทความนี้เรา สร้างตัวแบบของปัญหาการออกแบบการวางผังในรูปของกำหนดการเชิงเส้นจำนวนเต็มผสม ตามจุดอ้างอิงของตัวแบบที่จุด ตรงกลางของห้อง และใช้หลักการของตัวแปรทวิภาค และฟังก์ชันเป้าหมายแบบหลายเป้าหมาย โดยอาศัยหลักการของการขยายและการจำกัดเขตในการหาผลเฉลยที่เหมาะที่สุด อย่างไรก็ตามแม้ว่า การออกแบบการวางผังในรูปของกำหนดการเชิงเส้นจำนวนเต็มผสมนั้น จะง่ายต่อการสร้างและปรับตัวแบบให้สอดคล้องกับความต้องการของผู้ออกแบบ แต่จำนวนห้องก็มีผลต่อเวลาในการหาผลเฉลยเป็นอย่างมาก

ด้วยเหตุนี้เราจึงทำการลดจำนวนรอบการหาผลเฉลยของตัวแบบลง โดยการเร่งกระบวนการหาผลเฉลยของตัวแบบกำหนดการเชิงเส้นจำนวนเต็มผสมในขั้นตอนของการขยายและจำกัดเขตให้ไปสู่คำตอบเร็วขึ้น โดยนำหลักการของขั้นตอนวิธีทางพันธุกรรมมาช่วยหาลำดับของตัวแปรในการขยายเพื่อเป็นการลดปริภูมิในการค้นหาในขั้นตอนการหาผลเฉลยลง ส่งผลทำให้เราสามารถลดจำนวนรอบของการหาผลเฉลยลงได้อย่างมีนัยสำคัญ

ภาควิชา...........คณิตศาสตร์.............. ลายมือชื่อนิสิต.........ฐิติยา.........

สาขาวิชา..........วิทยาการคณนา.......... ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก................

ปีการศึกษา............2552....................

5072264623 : MAJOR COMPUTATIONAL SCIENCE KEYWORDS : ARCHITECTURAL LAYOUT DESIGN / OPTIMIZATION PROBLEM/ MIXED INTEGER PROGRAMMING / GENETIC ALGORITHMS

THITIYA THEPAROD : ACCELERATING THE SOLVING PROCESS OF LAYOUT DESIGN USING GENETIC ALGORITHM.THESIS ADVISOR : ASSTSTANT PROFESSOR KRUNG SINAPIROMSARAN, Ph.D., 72 pp.

The layout design optimization is a complicated process of an architectural design which is concerned with finding feasible locations and size of rooms that meet design requirement and design preference. This paper formulates the optimal layout design as multi-objective mixed integer programming model using the binary variables and branch & bound technique to determine the best location and size of a group of interrelated rectangular rooms by placing a representative point at the center of the room. Although solving the layout problems using MIP model is easy to formulate and adapt for meeting architectural requirements, the number of iterations to find the optimal solution is still influenced by the number of rooms. For this reason, we decrease the number of iterations by accelerating branch and bound process. The genetic algorithm has been adopted to find a candidate sequence of branching variables which helps reducing the search tree. From the empirical test, we found that the iterations can be reduced significantly.

Department : ....Mathematics.... Student's Signature : ....

Field of Study : ....Computational Science.... Advisor's Signature : ....

Academic Year : ....2009...........

ACKNOWLEDGEMENTS

I would like to express my full gratitude to all those who made it possible to complete this thesis. I am deeply grateful to my thesis advisor Assistant Professor Dr. Krung Sinapiromsaran whose help with his simultating suggestions, superior encouragement and magnificent guidance helped me during the time of all my research and in the writing of this thesis. I furthermore would like to thank Associate Professor Dr. Pornchai Satravaha, Assistant Professor Dr. Anusorn Chonwerayuth and Dr. Kamol Keatruangkamala and my thesis committee; for their useful suggestions for improvement. My grateful thanks are also extended to my teachers and lecturers during my study.

Lastly but by no means least, I would like to give my special thanks to my family and friends whose encouragement enabled me to complete this thesis.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Motivation

Architecture is an art and science of designing and constructing buildings and other physical structures for human shelter (Arch.Thamer Al Jbarat). Designing building involves both interior and exterior design, which must address both the feasibility and the building cost, as well as the function and aesthetics for the dweller.

Interior design is a multi-faceted profession in which creative and technical solutions are applied within a structure to achieve a built interior environment. Interior design involves the floor planning, interior decoration,etc.

The layout design is a complex process of an interior architectural design. An architect starts by relationship diagram between rooms, spaces and other physical objects within the structure. Then he/she will focus on the size of rooms, and the distance among rooms according to given relationship. He/she normally studies and drafts many possible alternatives to select the best layout design before submitting the plan to be implemented. However, due to the combinatorial effects of layouts, the researchers must cope with thse problems as a combinatorial optimization.

A number of representations have been proposed for the interior achitectural design which are often grouped as the direct representation, the slicing structure, and non-slicing structure (Berntsson and Tang,2004). In 1986, Wong and Liu [1] presented an algorithm based on the slicing structure of an architecture layout. A slicing structure divides the floorplan by recursively cutting rectangles into finer rectangular objects using vertical and horizontal lines and then fit blocks into each segment (see figure 1.1.1).

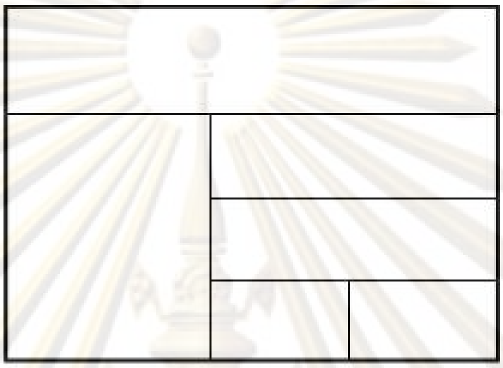

Figure 1.1.1: Slicing structure

A slicing floorplan can be represented by an oriented rooted binary tree, call slicing tree. Each parent node of the tree corresponds to a vertical and horizontal cut. Each leaf node represents a single segment. The system then computes the corresponding segment dimentions and their positions coordinates. The procedure starts with a random bipartitions and improve the partition iteratively by performing a sequence of vertex movements. If the resulting floorplan design is acceptable, then the design process is terminated. This technique helps reducing the complexity of the problem and search space, leading to shorter runtime.

Even though slicing floorplans are easy to used for optimization, this representation cannot handle non-slicing floorplan. In recent years, many researchers have been divising more layout representations to the topology using a directed acylic graph, such as B*-tree [2], Transitive Clusure Graph (TCG)[3], Twin Binary Sequence [4], and Corner Block List (CBL)[5][6].

For a non-slicing floorplan, researchers have proposed several representations.

In 1991, Onodera et al.[7] classify a topological relationship between two blocks into four classes and use the branch and bound method to solve it. The algorithm runtime is $O(2^{n(n+2)})$ which makes it impossible to handle a large problem at that time.

In 1995, sequence pair representation was proposed by Murata et al.[8]. They use two sets of permutations to present a geometric relationship of blocks. In the following year, 1996, Nakatake et al.[9] devised a bounded slicing grid approach. An $n \times n$ grid plane is used for the placement of n blocks (see figure 1.1.2). A block can have several choices to place, however the disadvantage of this representation is due to its redundancy.

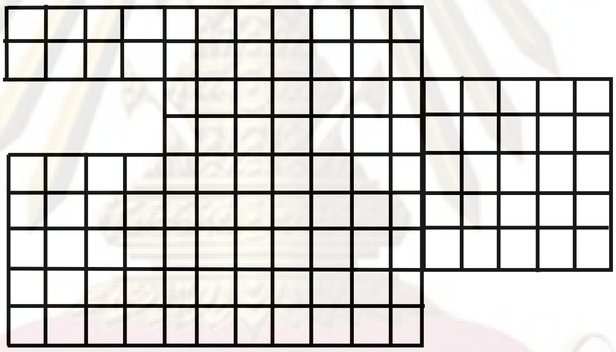

Figure 1.1.2: Grid structure

Another representation called the ordered tree (O-tree) representation was proposed in 1999 by Guo et al.[10]. There are two type of an O-tree consists of horizontal and veritical O-tree. In the O-tree, a node denotes a block, an edge denotes the horizontal or vertical related positions between blocks and the permutation determines their vertical relationship. The geometric constraint exists when we can draw a horizontal or vertical line between two blocks without passing through other blocks. See the example of horizontal O-tree in figure 1.1.3

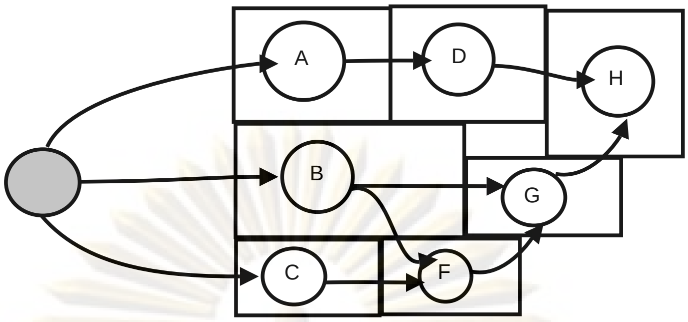

Figure 1.1.3: O-tree and placement

The O-tree is a simplified struture to present the geometric relation. The tree structure is well known in applied mathematics and computer science and the properties of tree are very straightforward and simple. The run time for transforming an O-tree to its representing placement is linear with respect to the number of blocks, i.e. O(n) [11].

In the designing process, qualitative and quantitative analyses are needed to be treated. Some qualitative measures such as noise and vibration disturbances must be handled appropriately whereas quantitative measures such as the cost of transporting products between department or the cost of commuting between rooms or the cost of laying the communication wiring must be handled mathmatically. The layout design optimization with both qualitative and quantitative is generally difficult to formulate using a mathematical programming model.

Attempts to automate the process of layout design problems started many decades ago. Pioneer work by Armour and Buffa in 1963 [12] presented a heuristic algorithm and simulation approach using a computer program to determine the suboptimum relative location patterns for facilities. The computer output obtains a block diagrammatic layout of facility area, where area does not need to be equal.

Another technique that was proposed in layout planning is a use of linear graphs for representing a floor plan [13]. Researchers have used several approaches to deal with this problem.

Drew J. van Camp, Michael W. Carter and Anthoni Vannelli [14] proposed a nonlinear programming technique (NLT) used to formulate the layout problem based on three constraints. Two constraints are based on a layout structure, that is, two departments may not overlap, and may not be located outside the boundary area. The last constraint is based on the dimension boundary of each department. The shape of every department must be rectangular and the area must be fixed, while the height and width are optimized using mathematical programming solver. To this end, three models used in NLT to approximate a layout problem were developed. The performance of NLT on several test problems were presented. Running time to solve these problems is acceptable for real-world problems comparing to some other algorithms.

Since a layout problem is known as NP-hard (Sahni and Gonzalez, 1976) and cannot be solved exhaustively for reasonably sized layout problems, many heuristic approaches have been proposed to avoid searching the design space exhaustively.

David M. Tate and Alice E. Smith [15] presented a heuristic search methodology called Genetic Algorithms (GA) for unequal area layout and showed how optimal solutions are affected by constraints on permitted department shapes, as specified by a maximum allowable aspect ratio for each department.

Thelma D. Maridou and Panos M. Pardalos revised (1997) a simulated annealing and genetic algorithms for solving facility layout problem approximately [16].

Russel D. Meller [17] reformulated Montreul's model by improving perimeter constraint, refining his binary variables and tightening the department area constraints. Optimum solution for facility layout problem (FLP) has been reported with a minimum of eight departments.

I- Cheng Yeh (2005) presented a new framework for a facility layout problem, named annealed neural network which arises from a combination of the Hopfield neural network and the simulated annealing [18]. The first is a representation model of a layout problem and the second is a search algorithm for finding the optimum or near optimum solutions. Annealed neural network exhibits the rapid convergence of the neural network, while preserving the solution quality afforded by simulated annealing.

Exact algorithm for solving this layout problem such as the branch and bound algorithm faces with large computational time. To alleviate this, the use of the genetic algorithm (GA) has become an increasingly popular tool in a computational optimization problem in recent years in order to deal with large-size instances. This is because GA offers several attractive features. GA is easy to understand and can be applied to many types of optimization problems with little or no modification, while other approaches have required substantial modification for using in building applications successfully. It is effective to solve complex problems that can not be solved by other optimization methods. GA is easy to be implemented. Other methods may offer the better performance but must be identified and configured properly. The main advantage of using a GA is that while other methods always process single solution, GA maintains a population of potential solutions.

Romualdas Bausys and Ina Pankrasovaite (2005) proposed improved GA for the architectural layout problem since the rate of convergence of standard genetic algorithm for solving a problem is not very effective for some realistic problems[19]. The performance of standard and improved genetic algorithm for architectural layout design has been analyzed and compared. The results demonstrate the improvement of their new algorithm.

In this thesis, we use the representation similar to the work from Bloch et.

al.[20] using a coordinated system. According to a nature of an architectural problem, there are several aspects to consider. Thus, this problem is not appropriated to be formulated as a single objective function. In this thesis, the multi-objective fuction has been used in order to meet several objective requirements. Our mathematical model has been constructed based on the work of Kamol Keatruangkamala and Krung Sinapiromsaran [21] using rectangular components. In [21], they use a point at the top left corner of the room to be set as a reference point. For our research, we use the reference point at the center of the room. Our thought is encouraged by observation that middle point is common for a symetrical room. Besides, research in this field often concentrates on the reference point at the center of the room such as the work of Michalek [22].

Because a computational time of a mathematical model has grown exponentially large, Kamol and Krung presented valid inequality to reduce their computational time and iteration [23]. From their experiments, the computational time and the number of iterations have been reduced leading to solve the larger problem size. However, the reduction is not consistant. As a problem grows, the capability of reduction is decreased because of the redundancy.

In this thesis, we formulate a layout problem as a Mixed Integer Programming model (Grorge, 1988, Linderoth et al., 1999, Russell et al., 1999) to determine the optimal solution using binary variables. We named this mathematical model as MIP. From our experiments, we found that a computational time for solving problems with more than seven rooms exceeds seven hours, which is far from satisfactory. To reduce the computational time significantly, Genetic Algorithm are proposed, called GALMIP. We use GA as a robustness learning methodology to find the order of branching variables in the branch and bound algorithm which is the process of solving MIP. By knowing a sequence of branching variables it helps reduce a search space by identifying the better path in the search tree.

1.2 The research objective

We expect to accelerate the branch and bound process of a layout design optimization via a special order set using a genetic algorithm for MIP model.

1.3 Overview of this thesis

The rest of this thesis is organized as follows.

In chapter II, we introduce backgrounds used in this thesis including linear programming, integer programming and genetic algorithm.

Chapter III presents an overview of the proposed model based on the MIP methodology. We also explain how to apply genetic algorithm to our problem in this chapter.

Finally, the results of our experiments are presented in chapter IV. We also discuss with some concluding remarks and suggestions in the last chapter.

CHAPTER II

BACKGROUND KNOWLEDGE

This chapter provides some important background knowledges that are used in this thesis. It consists of two main sections.

First, we introduce the mathematical programming background which is used to model a layout design problem. Then the branch and bound algorithm is applied to solve the Mixed Integer Programming model (MIP).

Secondly, we describe a genetic algorithm which is applied to the variable selection for the branch and bound algorithm.

2.1 Mathematical programming model

2.1.1 Layout design

A layout design problem is considered as a process of finding the best location and size of components. Each component is usually refered as an orthogonal regtangular *unit* for a specific architectural function, such as living space, storage space, facility and accesiblity space [24].

Components are grouped into several categories based on their functions, which are *room*, *boundary*, *hallway*, and *access way*[24]. A room is refered to a space which is considered by a designer for a living area. A boundary is a space that encloses other components inside. A hallway is a space that functions as a connector between components. Access way is a small area that connects two components together, usually one is a hallway and another is a room. Figure 2.1.1 shows an example of a simple layout whose composes of a bedroom, a bathroom, a living room, a kitchen and a hall. The letter "a"represents access-ways between two components.

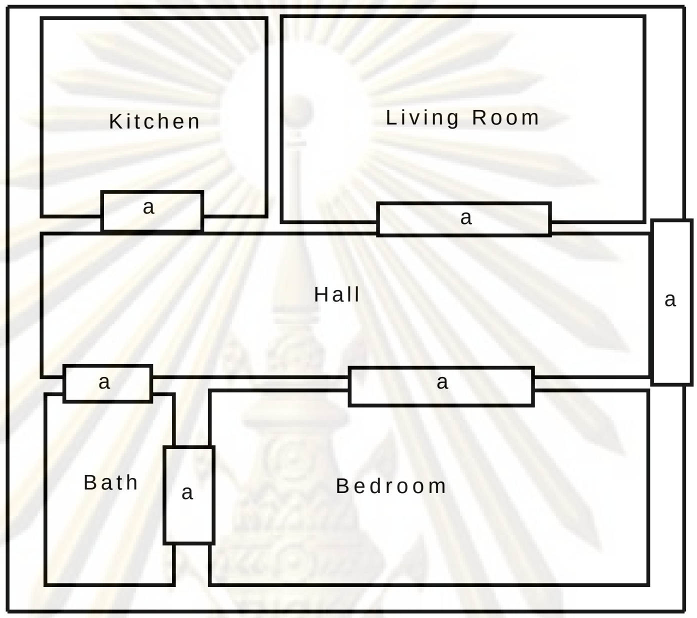

Figure 2.1.1: An example of a simple layout

From figure 2.1.1 we can see that a valid layout design does not overlap internal components and the total area is the sum of areas from all components excluding access ways. This representation is easy to formulate using a mathematical model. A popular technique for an optimization problems is to use a linear programming which is described in the next section.

2.1.2 Linear Programming

In this section We will describe an algebraic representation and its assumption, which are neccessary for formulating a linear programming model.

- **Definition**

Linear programming model is a mathematical programming model widely used for solving optimization problems in several aspects of business including production and management, which are needed to optimize the objective function subject to certain constraints. The mathematical expressions for the objective function and constraints are linear.

- **Assumptions of the linear programming model**

To model using a linear relation, the problem must satisfisfy.

1. *Proportionality* - It requires that the value of each term in the linear programming problem is strictly proportional to the value of the variable in the term. We can also say that the slope of objective and constraints function is constant.

2. *Additivity* - It concerns with the relationship among decision variables. Terms in the objective function and constraint equations must be additive.

3. *Divisability* - The decision variables are real-valued which means they can take on fractional values and therefore they are continuous. However, fractional values are not suitable to represent some quantities, such as the number of workers. So in this case, it is appropriate to use an integer variable.

4. *Certainty* - Parameters in the model are required to be known before they are solved.

Next we give the general mathematical programming formula of the linear programming problem.

- **Standard form**

A standard form of a linear programming problem is to find a vector $(x_1, x_2, ..., x_j, ..., x_n)$ which minimizes the linear objective function

$$c_1x_1 + c_2x_2 + ... + c_jx_j + ... + c_nx_n \tag{II.1}$$

subject to the nonnegativity constraints

$$x_j \geq 0 \qquad\qquad j = 1, 2, \ldots, n$$

and linear constraints

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + ... + a_{1j}x_j + ... + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + ... + a_{2j}x_j + ... + a_{2n}x_n &= b_2 \\
\ldots\ldots\ldots\ldots\ldots \\
a_{i1}x_1 + a_{i2}x_2 + ... + a_{ij}x_j + ... + a_{in}x_n &= b_i \qquad (\text{II.2}) \\
\ldots\ldots\ldots\ldots\ldots \\
a_{m1}x_1 + a_{m2}x_2 + ... + a_{mj}x_j + ... + a_{mn}x_n &= b_m
\end{aligned}$$

where a_{ij}, b_i, and c_j are given constants. We shall always assume that the equation in II.2 has been multiplied by -1 where necessary to make $b_i \geq 0$.

There are several notations in common use.

1. Minimize $\sum_{1}^{n} c_jx_j$

   subject to $\sum_{j=1}^{n} a_{ij}x_j = b_i \qquad i = 1, 2, \ldots, m$

   and $x_j \geq 0 \qquad j = 1, 2, \ldots, n$

2. Minimize $\mathbf{c}x$

   subject to $Ax = b$

   and $x \geq 0$

   ct where $\mathbf{c} = (c_1, c_2, ..., c_n)$ is a row vector, $x = (x_1, x_2, ..., x_n)^T$ is a column vector, $A = [a_{ij}]_{mxn}, \mathbf{b} = (b_1, b_2, ..., b_m)$ is a column vector, and 0 is a n-dimensional null column vector.

3. Minimize $\mathbf{c}x$

   subject to $x_1P_1 + x_2P_2 + ... + x_nP_n = P_0$

   where P_j for $j = 1, 2, ..., n$ is the j^{th} column of the matrix A and $P_0 = b$ (Saul I. GASS, 1964)

- **Solving linear programming problems using the Simplex method**

Simplex method is a general procedure for solving linear programming problems, published by George Dantzig in 1947. The Simplex method guarantees to find the optimal solution in a finite number of steps. It starts at the basic feasible solution and moves along the border of the feasible region step by step from the corner point (extreme point) to an adjacent corner point where larger(for maximization) or smaller (for minimization) value of objective function is obtained at each step. This procedure is summarized as follows.

Step 1 : Start with the initial basic point feasible solution.

Step 2 : Check the optimality of the current basic feasible solution. If none of its adjacent basic feasible solutions have better objective values, the current basic feasible solution is optimal. Otherwise, go to step 3.

Step 3 : Move from the current basic feasible solution to a better adjacent basic feasible solution. Repeat this step until the stopping conditions are met.

2.1.3 Integer Programming (IP)

In many practical problems, only integer values of decision variables are needed. Problems in which this is the case are called integer programming (IP) problems. The mathematical model for integer programming is simply the linear programming model with one additional restriction, all variables must be integral. If only variables are required to have integer values, then this model is referred to *Mixed Integer Programming* (MIP). Therefore, the minimization of the MIP is.

$$
\begin{array}{llll}
\text{Minimize} & & Z = \sum_{1}^{n} c_j x_j & \\
\text{subject to} & \sum_{j=1}^{n} a_{ij} x_j = b_i & & \text{for } i = 1, 2, \ldots, m \\
\text{and} & & x_j \geq 0 & \text{for } j = 1, 2, \ldots, n \\
 & & x_j \text{ is integer,} & \text{for } j = 1, 2, \ldots, m \quad (m \leq n).
\end{array}
$$

If $m = n$, this problem becomes the pure IP problem.

In this thesis, an architectural layout design problem is modeled as the MIP. There are several different algorithms to solve it. Popular methods are based on solving the LP relaxation because solving integer programming directly is more difficult than solving linear programming. A branch and Bound algrirthm is one of the algorithms relating to LP relaxation. We will describe in more details in the following section.

2.1.4 A Branch and Bound Algorithm for MIP

A branch and bound algorithm is a widely used approach for solving MIP presenting in this section. The structure of this algorithm was first developed by R.J. Dakin, based on a pioneering work from A. H. Land and A. G. Doig. Branch and bound algorithm uses the divide and conquer method with the best first search strategy. The basic concept is to divide a large problem into smaller ones. The dividing or branch part is a process that partitions the entire set of feasible solutions into smaller subsets. The conquering part is done by estimating how good a solution can be for each smaller problem. We may have to divide the problem further, until we get an optimal solution. The feasible region of the a subproblem is a subset of the feasible region of the original problem.

Next, we describe three basic steps of the branch and bound algorithm consisting of branching, bounding and fathoming.

- **Branching**

This process is to partition a feasible region into several smaller subproblems. The partitioning will be repeated recursively in each subproblem until a subproblem is fathomed. The rule of fathoming will be described in the next section. Branching is generally represented in terms of a tree structure.

Let S be the feasible region and the set of all solutions.

S_i : A subset of S where $i = 1, 2, ..., n$

The branching process will lead to a partition of S in which,

$$S_1 \bigcup S_2 \bigcup \cdots \bigcup S_n = S,$$

$$S_i \bigcap S_j = \emptyset \ , \ i \neq j$$

The partitioning involves choosing variables to create new subproblems. The variables will be chosen specifying in a range of values. Since some variables in the MIP problem are restricted to be integer. Therefore, only the restricted variables that has a non-integer value will be chosen, see figure 2.1.2 .

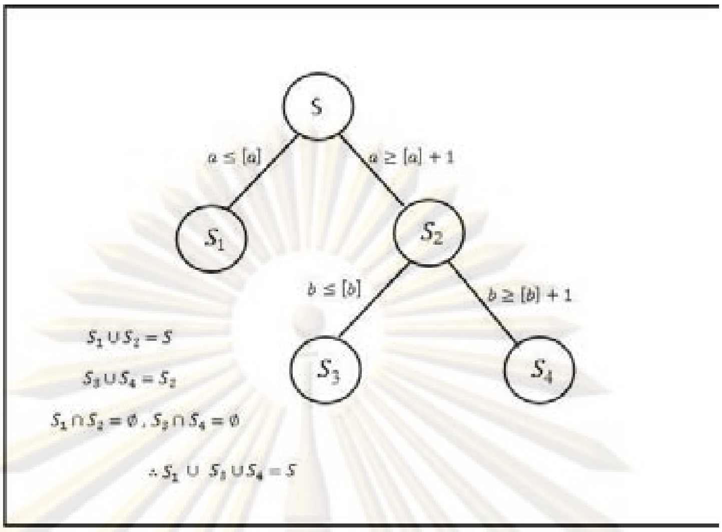

Figure 2.1.2: The branch and bound tree

Figure 2.1.2 illustrates the concept of branching for the case where there are only four supbproblems. S represents the original feasible solution set while S_1, S_3, S_4 represent feasible solutions of subproblems divided from the original one. The variables a and b are variables that have a non-integer value. The variables used in the branching process are called *branching variables.*

- **Bounding**

In order to determine the best feasible solution, we need to find bounds from solving relaxations to develop bounds for a solution. To maximize the problem, the lower bound is given as the best current solution.

Given Z = optimal value of associated LP relaxation.

Z^* is the best upper bound.

Z_* is the best known feasible solution.

Then, the bound of the solution of the maximization problem is

$$Z_* \le Z \le Z^*$$

Note that a subproblem that has the optimal solution worse than Z_* can be fathomed, and dismissed from a further consideration. The fathoming rule will be discribed more in the next section.

- **Fathoming**

A subproblem can be fathomed based on the following three rules described as follows.

Rule 1. The relaxation of the subproblem has an optimal solution with $Z \leq Z_*$ for maximization problem and $Z_* \leq Z$ for minimization problem, where Z_* is the value of the best current solution.

Rule 2. The LP relaxation of the subproblem has no feasible solution.

Rule 3. The LP relaxation of a subproblem has an optimal solution that have all integer values (or binary if it is BIP).

2.2 Introduction to Genetic Algorithm

This section describes some fundamental basics of a genetic algorithm.

A genetic algorithm is a part of evolutionary computing inspired by Darwin's theory about evolution.

2.2.1 Overview of Genetic Algorithm

A genetic algorithm (GA) was invented and developed by John Holland and his students and colleagues at the University of Michigan in the 1960's and 1970's. Genetic algorithm has been widely studied and applied in science and engineering world. The algorithm is used as an optimization method that has been applied to many areas. Although the range of problems that a genetic algorithm has been applied is quite broad, this algorithm is often viewed as a function optimizer.

Genetic algorithm is usually applied to optimization problems that are difficult to solve by a mathematical formulation. It is also used to resolved NP-hard and NP-complete such as traveling saleman problem (TSP), scheduling and design problems. It performs searching throughout the solution space to find the near optimal solution.

An implementation of a genetic algorithm starts with a set of solutions called population. These solutions may be represented by character strings which are referred to chromosomes. A chromosome is made up from discrete units called genes. The population normally randomly initialized. The structure of GA is illustrated as follows.

BEGIN GA

Creat an initial population of P and Evaluate the fitness of each chromosome.

WHILE $n \leq N$ **DO**

BEGIN

1.Select parents from population via roulette wheel selection.

WHILE stopping condition not fulfilled **DO**

BEGIN

2.Choose at random a pair of parents for mating.

3.Produce offsprings from the chosed parents by crossover operator.

4.Process each offspring by the mutation operator.

END

5.Replace the old population of chromosomes by the stronger ones.

6.Evaluate the fitness of each chromosome in the new population

END

END GA

The above algorithm refers to the evaluation, crossover operator to exchange bit string and mutation operator to introduce random perturbations in the search process. These concepts are now described precisely in the next section.

2.2.2 Representation and Operators

The performance of the GA is controlled by the following factors.

- **Encoding**

The encoding process is often the most difficult aspect of solving a problem using genetic algorithm. It depends on many fators. It is often hard to find an appropriate representation to efficiently perform the crossover process. The popular representation is using a string of zeros and ones, called binary string representation which is not restricted to number. There are several ways to represent it such as permutation encoding, value encoding, tree encoding and matrix encoding etc. For this thesis, we will explain binary string representation.

For instance, suppose we have a problem which has solutions in the set of $\{0, \ldots, 5\}$. The binary representations of 4 and 5 are 110 and 011 respectively. The example of chromosomes with binary representation is shown as follow.

Parent A :	001	001	101	100	010
Parent B :	001	110	100	010	000

Figure 2.2.1: Example of chromosome A and B

This representation is often not natural for many problems and sometimes correction must be made after crossover and/or mutation.

- **Evaluation**

We use the evalution function to decide how good a chromosome is, which plays important role in genetic algorthms. For optimization problems, it usually is an objective function. The value of the evaluation function is calculated for an individual of population, called fitness value. This fitness may use to decide the probability that a particular chromosome would be chosen to contribute to the next generation.

- **Crossover**

Crossover is a straightforward procedure where the two chromosomes are recombined to new chromosomes which are copied into the new generation. Not every chromosome is used in crossover. The chromosomes are chosen randomly based on a fitness value of each chromosome given by the evaluation function. Chromosomes with the highest fitness are more likely to be chosen. After the crossover is performed, new chromosomes created by crossover called offspring are moved into the new generation. For this reason, the next generation are expected to be better than the previous generation because the best chromosomes from previous generation were used to create the new generation. Crossover continues until the new generation is full.

The simplest case of crossover is one-point crossover operator. Pick two random chromosomes to be parents for crossover. A random position between 1 and $L-1$ along the two parent chromosomes are chosen, where L is the chromosome's length. Then, switch all genes after that point to create two offsprings. For example using our parent chromosomes in figure 2.2.2.

Parent A : 001 001 101 100 010

Parent B : 001 110 100 010 000

Figure 2.2.2: Two parent chromosomes

We randomly choose the crossover point, here is at position three. Then, switch all genes after the random position.

Offspring A : 001 001 101| 010 000

Offspring B : 001 110 100| 100 010

Figure 2.2.3: Two offspring from one-point crossover

Some new generated offspring might not satisfy the representation criteria. To change the generated offspring into the legal representation we need a *repair algorithm*. Moreover, it is possible that selected chromosomes are copied to the new population directly without any modification. This will ensure that good chromosomes can be preserved from one generation to the next. We can also have two - point crossover. Two cut points are chosen randomly and the substring located between two cut points are exchanged, see figure 2.2.4.

Parent A : 001 | 001 101| 100 010

Parent B : 001 | 110 100| 010 000

Figure 2.2.4: Two parent chromosomes A and B with two random cut points

To get two offspring, swich the genes between the two points.

There are various derivation of the crossover routines. Other generalizations, like the M-point crossover and the uniform crossover (Syswerda,1989) may be

Offspring A : 001 | 110 100| 100 010

Offspring B : 001 | 001 101| 010 000

Figure 2.2.5: Two offspring from two-point crossover

found in literature.

- **Mutation**

Due to the randomness of the crossover process may occasionally find a local optimum but not the global optimum. Because the chromosomes close to the local optimum will have a better fitness, they will be selected to crossover. This may cause GA to find the local optimum instead of the global optimum. So the mutation is used to introduce random perturbations into the search space. It is created to avoid a local optimum trap. It is essential to introduce diversity in homogeneous populations, and to restore bit values that cannot be recovered via crossover. Therefore, mutation is a completely random way for obtaining to possible solutions that would otherwise may be missed.

The bits of the two offsprings generated by the crossover operator are then processed by the mutation operator. The operator is applied to each bit with a probability equal to the mutation rate, which is close to zero. A chromosome is selected from the new generation. Then, we randomly choose a point to mutate and switch that point from 0 to 1 or from 1 to 0. For instance, we have an example with the random mutation point at third position.

Offspring A : 001 001 **101** 010 000

We have,

Offspring A* : 001 001 **010** 010 000

2.2.3 Characteristics of the genetic search

The search performed Genetic algorithm can be characterized in the following manner (Goldberg 1989):

1. Genetic algorithms manipulate bit strings or chromosomes encoding useful information about the problem, but they do not manipulate the information (no decoding or interpretation).

2. Genetic algorithms use the evaluation of a chromosome, as returned by the fitness function, to guide the search. They do not use any other information about the fitness function or the application domain.

3. The search is run in parallel from a population to population.

4. The transition from one chromosome to another in the search space is done stochastically.

In this section, we describe a simple genetic algorithm that solves a combinatorial problem. For more details about a genetic algorithm, the reader is suggested to read the standard book such as GA in Search Optimization, and Machine Learning (Goldberg,1989). The next section we will explain how a genetic algorithm is applied to our problem.

CHAPTER III

Problem Methodology

In this thesis, we propose two methodologies which will be described in two sections. In the first section, we describe how to formulate the mathematical model of the problem as a mixed integer programming model base on multi-objective function. For the second one we use a genetic algorithm as a machine learning algorithm to learn a special order set (SOS).

3.1 Mathematical model based on multi-objective function

3.1.1 Problem description

In this thesis, we define a room as a rectangular room based on the coordinated system. The relationship between distinct rooms ensures two rooms cannot occupy the same space. Every room must be inside the main building boundary. We use a point at the center of the room as the reference point and a point at the top left corner of the boundary area as the reference origin (0,0).

Design variables are defined as follow.

$x_i =$ X coordinate of the center of the room i.

$y_i =$ Y coordinate of the center of the room i.

$E_i =$ The distance between the center and east wall of the room i.

$N_i =$ The distance between the center and north wall of the room i.

Additionally, width and height of the boundary area are represented by parameters W and H, respectively. In addition, we use w_{min_i}, w_{max_i} which are the minimal and maximal width of the room i in order to control the width of the room i. Similarly, we use h_{min_i} , h_{max_i} to control the height of the room i. Thus we have,

$$w_{min_i} \leq w_i \leq w_{max_i} \tag{III.1}$$

$$h_{min_i} \leq h_i \leq h_{max_i} \tag{III.2}$$

Moreover, T_{ij} is the minimal access way between room i and room j, see figure 3.1.1

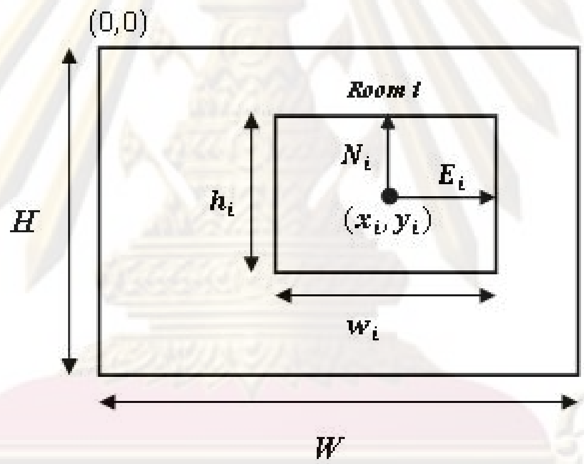

Figure 3.1.1: A room and a boundary representation

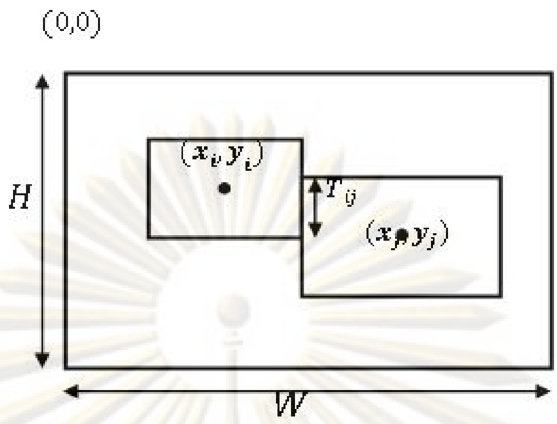

Figure 3.1.2: Relationship between the room i and j

3.1.2 Multi-objective linear programming

In the real world problem, it is rare for any problem to concern for only a single objective. Multi-objective optimization known as multi-criteria or multi-attribute optimization is the process of simultaneously optimizing two or more conflicting objectives subject to certain constraints. A standard technique for multi-objective problem is to minimize a positively weighted convex sum of the objectives, that is,

$$\sum_{n=1}^{n} \mu_i f_i(x), \quad \mu_i \geq 0,\ i = 1, 2, 3, \ldots, n.$$

By choosing the different weights u_i, for the different objectives, the preference of the decision-maker is taken into account. As the objective functions are generally of different magnitudes, they might have to be normalized first. Although the formulation is simple, the method requires a special treatment. The decision-maker must determine the weights used in the objective function according to his/her a specific purpose procedure.

- **The layout design multi-objective functions formulation**.

In this thesis, we concentrate on three objective functions, which are maximizing room area, minimizing the distance between rooms and the adding objective function which aim to eliminate alternative solutions. To cope with these multi-objective preferences, we combine three objective functions into a summation of weighted components. These weights can be adjusted according to architect's favor. In our experiment, we use equal weights to measure performance of our model.

$$\text{Minimize}\ \ u_1(x_i + y_i) + u_2 \sum (d^x_{ij} + d^y_{ij}) - u_3 \sum z_i \qquad \text{(III.3)}$$

Where x_i, y_i are X and Y coordinate of the room i, $i = 1, 2, , n$,

d^x_{ij}, d^y_{ij} are absolute distance on X and Y coordinate of room i and j $\forall i < j$,

z_i is a value that less than w_i and h_i $i = 1, 2, , n$,

u_1, u_2, u_3 are the weight values.

Objective III.3 denotes the minimization of the multiobjective optimization where the u_1 is the weight of the room i positioning to the nearest top left corner, u_2 is the weight of the total absolute distance and u_3 is the weight of the maximizing approximated room area. If an architect prefers larger room area then the weighted sum of u_3 is set to be greater than u_2. If an architect prefers a short total distance between rooms then u_2 is set to be greater than u_3. Hence, architect can generate alternative solutions by selecting different room i to be placed near the top left corner or reassign the desired objective weights. The z_i represents the maximized value between w_i and h_i that we can use to approximate the maximized area. Next, we describe each objective in detail.

- **Placing a room position near the origin**

Nevertheless, there always exist alternative solutions with the same objective value due to the layout rotation and symmetric which could affect the total solution time. In order to eliminate some unused solutions, we fix the room which is selected randomly from available rooms to be placed at the nearest origin of the boundary area, from III.4 here is room i. The formulation can be stated as follow.

Choose $i \in \{1, 2, \ldots, n\}$.

$$\text{Minimize} \quad (x_i + y_i) \tag{III.4}$$

where x_i is X coordinate of room i,

y_i is Y coordinate of room i.

The idea of this formulation is to minimize the summation of x_i and y_i. Being that (x_i, y_i) is the reference point at the center of the room and it identifies the location of the room, the value of x_i and y_i matter. If the reference point is nearest to the origin, the summation of x_i and y_i will be the smallest.

- **Minimizing the absolute room distance**

An interesting criteria of an architect's preference is a short distances between rooms. Calculating the distance as a linear function is not possible. The absolute distance function is preferred over the Euclidean distance function due to two reasons. The first reason is that it maintains the unit during the comparison. Therefore, there is no need to take the root of the sum square distance as the Euclidean distance. The second reason is that the walking distance from room to room can not join diagonally across the room to reach a target room. We could only walk along the boundary of any obstacle room. Manhattan Distance between two points in an Euclidean space is defined as the sum of the (absolute) differences of their coordinates.

For example, the Manhattan distance between the point of P_1 with the coordinates of (x_1, y_1) and the point of P_2 at (x_2, y_2) is

$$|x_1 - x_2| + |y_1 - y_2|$$

Then, the distance between any point of P_i at (x_i, y_i) and P_j at (x_j, y_j) is

$$d = \sum_{i,j=1}^{n} |x_i - x_j| + |y_i - y_j|,$$

where n is the number of points. The figure 3.1.2 illustrates the difference between the Manhattan distance and the Euclidean distance:

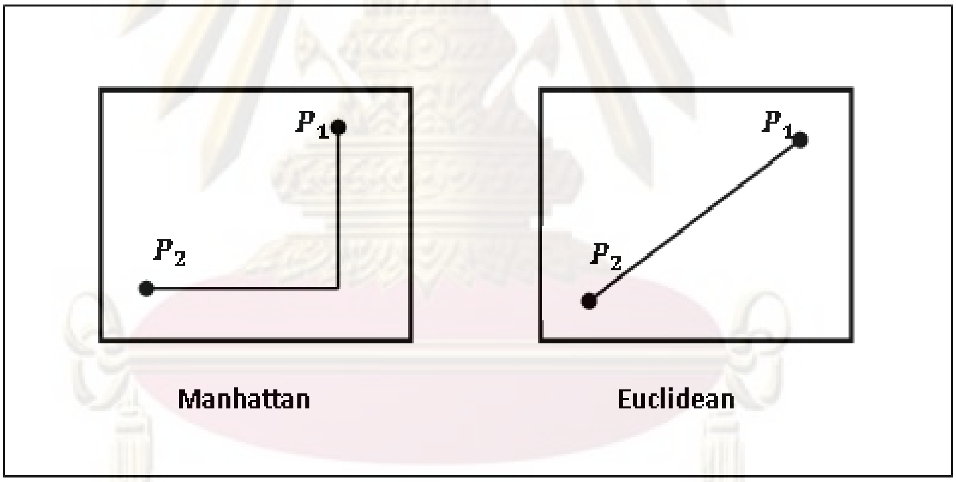

Figure 3.1.3: Manhattan and Euclidean distance

Then, our room distance objective is

$$\text{Minimize} \quad \sum(d^x_{ij} + d^y_{ij}) \tag{III.5}$$

$$\text{Subject to: } x_j - x_i \leq d^x_{ij}, \quad x_i - x_j \leq d^x_{ij}, \quad d^x_{ij} \geq 0,$$

$$y_j - y_i \leq d^y_{ij}, \quad y_i - y_j \leq d^y_{ij}, \quad d^y_{ij} \geq 0$$

- **Maximizing area of rooms**

Since the formulae of the area of a rectangular room is nonlinear, we have to find the area in another way in order to obtain linear formulae. Our objective is to maximize the room area so that we use the idea of maximizing each side of the room. So as the length and height increases, so does the area we obtain.

$$\text{Maximize} \quad \sum z_i \tag{III.6}$$

$$\begin{aligned} \text{subject to:} \quad z_i &\leq w_i \\ z_i &\leq h_i \end{aligned}$$

3.1.3 Problem constraint formulations

In this thesis, we formulate the problem as a linear function. Then we apply the absolute distance function called Manhattan distance to maintain the linear function, instead of using the commonly used Euclidean distance.

Location constraint identifies the location of rooms and explains the relationship between distinct rooms. Each pair of rooms does not need to be connected together. We use binary variables p_{ij} and q_{ij} to represent the four directions of north, south, east and west direction corresponding to the following constraints.

$$x_i + E_i \leq x_j - E_j + W * (p_{ij} + q_{ij}) \qquad p_{ij} = 0, q_{ij} = 0 \tag{III.7}$$

$$y_j + N_j \leq y_i - N_i + H * (1 + p_{ij} - q_{ij}) \qquad p_{ij} = 0, q_{ij} = 1 \tag{III.8}$$

$$x_j + E_j \leq x_i - E_i + W * (1 - p_{ij} + q_{ij}) \qquad p_{ij} = 1, q_{ij} = 0 \tag{III.9}$$

$$y_i + N_i \leq y_j - N_j + H * (2 - p_{ij} - q_{ij}) \qquad p_{ij} = 1, q_{ij} = 1 \tag{III.10}$$

From the location constraint, we use decision variables p_{ij} and q_{ij} to force the room i to be placed next to the north, south, east or west of the room j. Since the decision variables p_{ij} and q_{ij} are binary variables, four cases of (p_{ij}, q_{ij}) occur, which are (0, 0), (0, 1), (1, 0) and (1, 1).

For the first case, (p_{ij}, q_{ij}) is (0, 0). The solution must satisfy $x_i + E_i \le x_j - E_j$ for the constraint III.7 implying that the room j must be placed at the east of the room i, as in the figure 3.1.4.

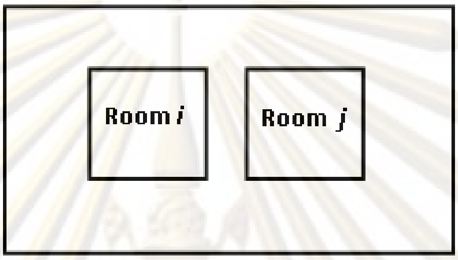

Figure 3.1.4: Location constraint representation for case $(p_{ij}, q_{ij}) = (0, 0)$

Simultaneously, constraint III.8 becomes $y_j + N_j \le y_i - N_i + H$. In view of the large value of H, the right-hand side of the constraint becomes a large positive value. Hence, any smaller positive $y_j + N_j$ will satisfy the constraint III.8. Similarly, constraint III.9 becomes $x_j + E_j \le x_i - E_i + W$ so that any positive value $x_j + E_j$ will be less than $x_i - E_i + W$ which means constraint III.9 is always satisfied for this case. Moreover, the constraint III.10 becomes $y_i + N_i \le y_j - N_j + 2H$. In view of the large value of H, every smaller value $y_i + N_i$ will be less than $y_j - N_j + 2H$, which guarantees that this constraint is always satisfied for the case of p_{ij} and q_{ij}.

The second case, (p_{ij}, q_{ij}) is set to be (0, 1) which leads the constraint III.8 to $y_j + N_j \le y_i - N_i$. It implies that the room j must be forced to place at the north of the room i, shown in the figure 3.1.5.

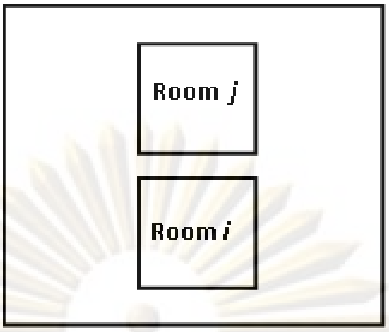

Figure 3.1.5: Location constraint representation for case $(p_{ij}, q_{ij}) = (0, 1)$

Constraint III.7 becomes $x_i + E_i \leq x_j - E_j + W$. In view of the large value of W, any small value $x_i + E_i$ will always be less than $x_j - E_j + W$, which means constraint III.7 is always satisfied for this case. At the same time, the constraint III.8 becomes $x_j + E_j \leq x_i - E_i + W$. Because of the same reason with constraint III.7, this constraint is satisfied. Similarly, constraint III.10 becomes $y_i + N_i \leq y_j - N_j + H$ which is also satisfied because of the large value of H.

The third case is similar to the other ones, the value of $(p_{ij}, q_{ij}) = (0, 1)$ makes the constraint III.9 become $x_j + E_j \leq x_i - E_i$ which leads the room j to be placed on the left of the room i, visualized as follow.

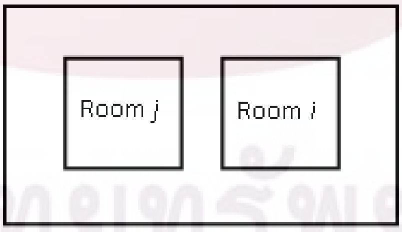

Figure 3.1.6: Location constraint representation for case $(p_{ij}, q_{ij}) = (1, 0)$

Simultaneously, the constraint III.7 becomes $x_i + E_i \leq x_j - E_j + W$. The large value of W on the right-hand side becomes a large positive value. For any smaller positive $x_i + E_i$ satisfies the constraint III.7 . While constraint III.8 and III.10 becomes $x_j + E_j \leq x_i - E_i + W$ and $y_i + N_i \leq y_j - N_j + H$. The positive values of

$x_j + E_j$ and $y_i + N_i$ are smaller than $x_i - E_i + W$ and $y_j - N_j + H$ which satisfy the constraint III.8 and III.10 respectively.

The last case, (p_{ij}, q_{ij}) is set to be (1, 1) which leads the constraint III.10 be $y_i + N_i \leq y_j - N_j + H$. This implies that the room j is forced to be placed south of the room i, as in figure 3.1.7.

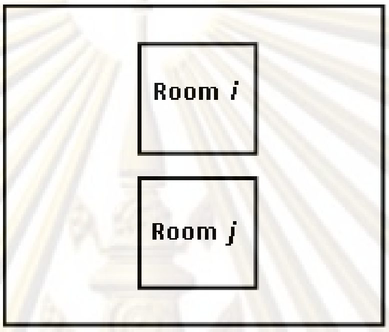

Figure 3.1.7: Location constraint representation for case $(p_{ij}, q_{ij}) = (1, 1)$

Similarly, constraint III.7 and III.9 becomes $x_i + E_i \leq x_j - E_j + W$ and $x_j + E_j \leq x_i - E_i + W$ respectively while the constraint III.8 becomes $y_j + N_j \leq y_i - N_i + H$. The large value of W and H in the right-hand side become a large positive value. Hence, any positive value $x_i + E_i$ will satisfy constraint III.7 and III.9, respectively. Also, any positive value $y_j + N_j$ satisfies constraint III.8.

Connectivity constraint explains and identifies the location of the rooms of each pair of rooms that have to be connected together. We use the same two binary variables p_{ij} and q_{ij} with different sets of constraints. By using the same argument as the Location constraint, four cases of (p_{ij}, q_{ij}) occur.

$$x_i + E_i \geq x_j - E_j - W * (p_{ij} + q_{ij}), \qquad p_{ij} = 0, q_{ij} = 0 \qquad \text{(III.11)}$$

$$y_j + N_j \geq y_i - N_i - H * (1 + p_{ij} - q_{ij}), \qquad p_{ij} = 0, q_{ij} = 1 \qquad \text{(III.12)}$$

$$x_j + E_j \geq x_i - E_i - W * (1 - p_{ij} + q_{ij}), \qquad p_{ij} = 1, q_{ij} = 0 \qquad \text{(III.13)}$$

$$y_i + N_i \geq y_j - N_j - H * (2 - p_{ij} - q_{ij}) \qquad p_{ij} = 1, q_{ij} = 1 \qquad \text{(III.14)}$$

The first case, (p_{ij}, q_{ij}) is set to be (0, 0) which leads constraint III.11 to be $x_i + E_i \geq x_j - E_j$. It implies that the room j must be forced to be placed on the right of the room i, shown in the figure 3.1.8.

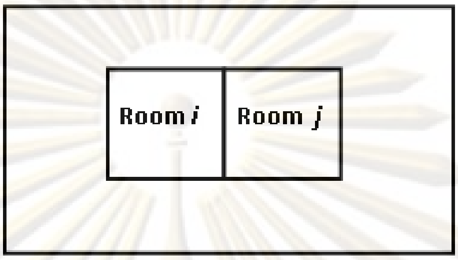

Figure 3.1.8: Connectivity constraint representation for case $(p_{ij}, q_{ij}) = (0, 0)$

Constraint III.12 becomes $y_j + N_j \geq y_i - N_i - H$. In view of the large value of H, the right-hand side of the constraint becomes a very small negative value. Hence, any positive $y_j + N_j$ will always be greater than $y_i - N_i - H$. Similarly, constraint III.13 becomes $x_j + E_j \geq x_i - E_i - W$ so that any positive value $x_j + E_j$ will be greater than $x_i - E_i - W$ which means constraint III.13 is always satisfied for this case. Moreover, constraint III.14 becomes $y_i + N_i \geq y_j - N_j - 2H$. In view of the large value of H, every positive value $y_i + N_i$ will be greater than $y_j - N_j - 2H$, which quarantees that this constraint is always satisfied for the case of p_{ij} and q_{ij}.

For second case, (p_{ij}, q_{ij}) is set to be (0, 1). Constraint III.12 becomes $y_j + N_j \geq y_i - N_i$, which force the room j to place at the north of the room i. Similar to the previous constraint, other constraints will be satisfied unconditionally due to the large value of H and W, see figure 3.1.9.

The third case, (p_{ij}, q_{ij}) is set to be (1, 0). Constraint III.13 becomes $x_j + E_j \geq x_i - E_i$, which forces room j to be placed on the left of the room i. Similar to the previous constraints, other constraints will be satisfied unconditionally due to the large value of H and W, see figure 3.1.10.

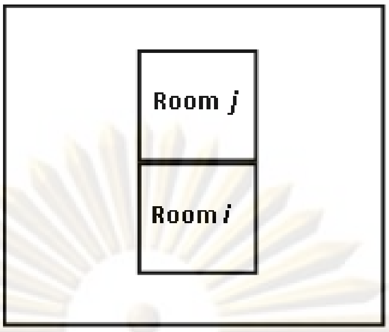

Figure 3.1.9: Connectivity constraint representation for case $(p_{ij}, q_{ij}) = (0, 1)$

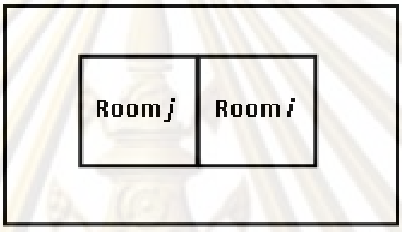

Figure 3.1.10: Connectivity constraint representation for case $(p_{ij}, q_{ij}) = (0, 1)$

The last case for conectivity constraint is (p_{ij}, q_{ij}) which becomes (1,1). Constraint III.14 becomes $y_i + N_i \geq y_j - N_j$, which forces the room j to be placed at the south of the room i. Similar to the previous constraint, other constraints will be satisfied unconditionally due to the large value of H and W, see figure 3.1.11.

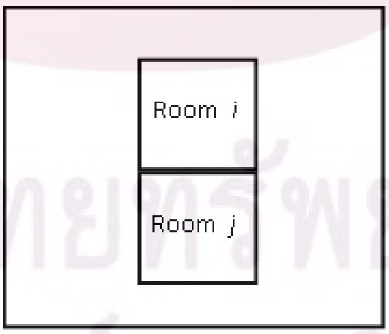

Figure 3.1.11: Connectivity constraint representation for case $(p_{ij}, q_{ij}) = (1, 1)$

Access-way constraint : If two rooms touch with each other, then the junction between two rooms must be wide enough to accommodate the accessway. We defined the minimal contact length of value T_{ij}.

$$y_j + N_j \leq y_i - N_i + T_{ij} - H * (q_{ij}), \qquad q_{ij} = 0 \qquad \text{(III.15)}$$

$$y_i + N_i \leq y_j - N_j + T_{ij} - H * (q_{ij}), \qquad q_{ij} = 0 \qquad \text{(III.16)}$$

$$x_j + E_j \leq x_i - E_i + T_{ij} - W * (1 - q_{ij}) \qquad q_{ij} = 1 \qquad \text{(III.17)}$$

$$x_i + E_i \leq x_j - E_j + T_{ij} - W * (1 - q_{ij}) \qquad q_{ij} = 1 \qquad \text{(III.18)}$$

Again q_{ij} is a binary variable, so q_{ij} can be either 1 or 0. If q_{ij} equals to 0, then constraint III.15 and III.16 becomes $y_j + N_j \leq y_i - N_i + T_{ij}$ and $y_i + N_i \leq y_j - N_j + T_{ij}$, respectively. While other constraints are satisfied unconditionally due to the large value of H and W. Constraint III.15 explains that the room j is adjacent to the room i at the upper corner of the room i, which has vertical contact, see figure 3.1.12 .

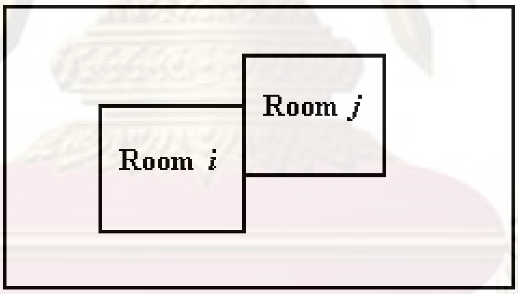

Figure 3.1.12: Access-way constraint representation for the adjacent area of the upper corner of the room i

Simutaneously, the room j is adjacent to the room i at the lower corner of the room i for constraint III.16, which has the vertical contact, shown as figure 3.1.13

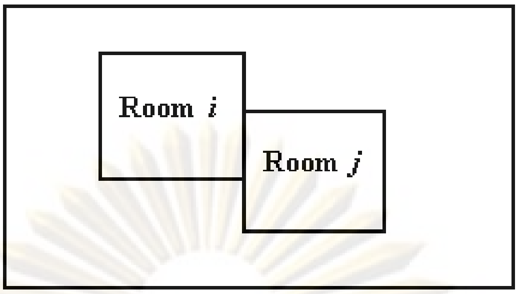

Figure 3.1.13: Access-way constraint representation for the adjacent area of the lower corner of the room i

Next, q_{ij} is 1. Constraint III.17 and III.18 becomes $x_j + E_j \leq x_i - E_i + T_{ij}$ and $x_i + E_i \leq x_j - E_j + T_{ij}$, respectively. While other constraints are satisfied unconditionally due to the large value of H and W. Constraint III.17 explains that the room j is adjacent to the room i at the left corner of the room i, see figure 3.1.14.

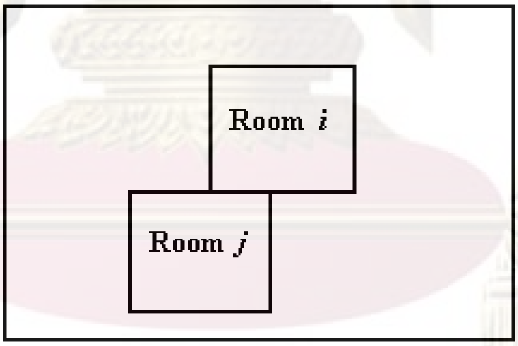

Figure 3.1.14: Access-way constraint representation for the adjacent area of the left corner of the room i

Simutaneously, the room j is adjacent to the room i at the left corner of the room i for constraint III.18, which has horizontal contact, shown as figure 3.1.15.

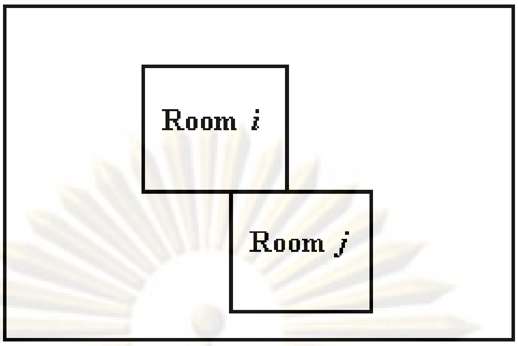

Figure 3.1.15: Access-way constraint representation for the adjacent area of the right corner of the room i

3.2 Genetic Algorithm technique

From the previous section, we have formulated the MIP to fit to the layout design problem. From the experiment, the computational time of the MIP model demonstrates that it can deal with a small-sized problem. For a larger sized problem, the computational time is still far from satisfactory. In order to accelerate the computational speed, a genetic algorithm has been adopted. In this thesis, we use the genetic algorithm as a robustness learning methodology to utilized an idea of the Special Order Set (SOS) based on the branching in a branch and bound algorithm.

Our purpose for using a genetic algorithm is to guide the sequence of branching strategy in MIP solving process. It is adopted to search the branch and bound tree and used to help finding the good path along the tree structure to the optimal solution. The good path will correspond to the order of branching variables. We therefore utilize the learning algorithm GA to find an appropriate squences of p_{ij} and q_{ij} which are branching variables of our problem. After completing the learning process, the stronger gene from GA represents the appropriated SOS with a good path in the search tree. For this reason, the appropriated SOS helps to prune the search tree that leads the algorithm to reach the optimal solution faster.

Next, we will describe our GA principles in detail.

3.2.1 Chromosomes

A chromosome is represented by a string which is a sequence of branching variables. Each bit of string contains a random branching variable, here is either p_{ij} or q_{ij}, see figure 3.2.1 for example.

Given P_0 is an original problem. P_2, P_3, P_4,..., P_n are subproblems. p_{i_r,j_r} and q_{i_r,j_r} are binary variables, where r corresponds to the order of branching and $1 \leq r \leq n$, n is the last order of branching and n is finite.

Suppose P_0 is divided into two smaller subproblems P_1 and P_2 using binary variable $p_{i_1j_1}$. P_1 is divided into smaller subproblem P_3 and P_4 using binary variable $q_{i_2j_2}$ and P_2 is divided to be P_5 and P_6 using binary variable $q_{i_3j_3}$ and so on until P_n.

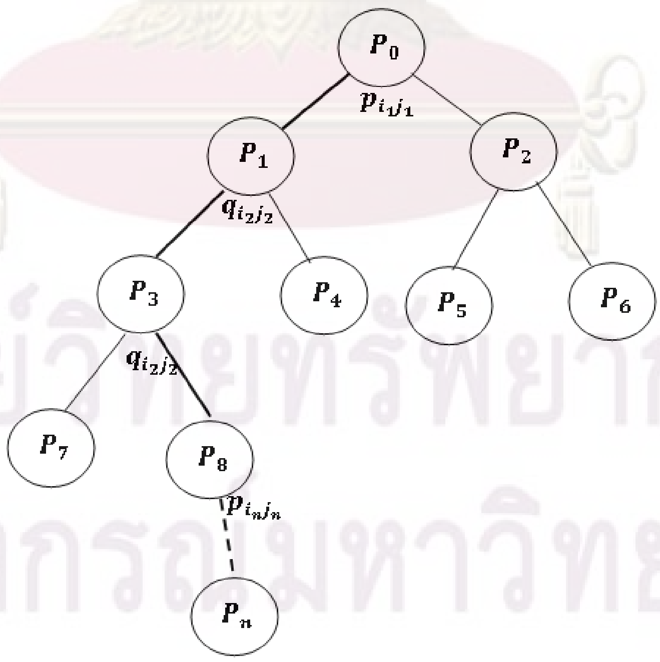

Figure 3.2.1: Representation of order of branching variables in tree structure

In the figure 3.2.1, we have a path from the top node(Problem P_0) to the bottom node (Problem P_n) pass through subproblems P_1, P_3, P_8,..., P_n. A sequence of branching variables is $\{p_{i_1j_1}, q_{i_2j_2}, q_{i_3j_3}, \ldots, p_{i_nj_n}\}$. Next, we will record this sequence into a chromosome by placing a first order of branching variable that is $p_{i_1j_1}$ at the first gene of chromosome. The second and third order and later order ones are placed consecutively to the right of the first gene, see figure 3.2.2.

$p_{i_1j_1}$	$q_{i_2j_2}$	$q_{i_3j_3}$	......	$p_{i_nj_n}$

Figure 3.2.2: A chromosome representation corresponding to a sequence of branching variables in tree structure

In this thesis, we use a two dimention(2D) binary string to store information of a sequence binary variables p_{ij} and q_{ij} because one dimention binary string can't be fit with entire information.The space of 2D binary string is $m \times n$,where the m presents the numbers of variables and the n represents the sequential order of variable p_{ij} and q_{ij}.

- **Encoding of a chromosome**

A chromosome is a chain formed by any characters. In genetics, the whole information of an individual structure is stored in a chromosome as genetic codes. The genome string is composed of a finite set of genes and their values. In this thesis, we encode the branching variables into a chromosome using 2D binary string. We fixed the numerical order for 12 variables of p_{ij} and q_{ij}, that are used in the SOS. Therefore, we need four bits to represent all possible cases of variables p_{ij} and q_{ij}. However, a four bits string can represent 16 different patterns which are larger than the number of the variables p_{ij} and q_{ij}. The remaining patterns will not be ignored during the GA run. Thus, for example, we can represent variable p_{ij} and q_{ij} for four rooms as follow.

$$
\begin{array}{ll}
p_{12} = 0001 & q_{12} = 0111 \\
p_{13} = 0010 & q_{13} = 1000 \\
p_{14} = 0011 & q_{14} = 1001 \\
p_{23} = 0100 & q_{23} = 1010 \\
p_{24} = 0101 & q_{24} = 1011 \\
p_{34} = 0110 & q_{34} = 1100
\end{array}
$$

Suppose we have the sequence of branching variables for 4 rooms in a chromosome as the following example.

$$p_{13}p_{12}q_{34}p_{23}q_{12}q_{23}$$

Therefore we have the 2D binary string representing the sequence of branching variables of the above example as:

$$
\begin{array}{cccccc}
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0
\end{array}
$$

Moreover, all numbers of zero (0000) are not used to representing any branching variable. Nevertheless, if the current pattern is not represented by any SOS variable, the algorithm will ignore and proceed with the next variable, and the index of this variable is not stored into a candidate SOS. This method ensures that only feasible SOS is created and will be used in the chromosome.

3.2.2 Operators

- **Selection**

As you have already known from the simple genetic algorithm which is described in Section II, chromosome are selected to be parents to crossover. According to Darwin's evolution theory the best ones should survive and create new offspring. There are many methods how to select the best chromosomes, for example roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others.

In this thesis, we use proportional selection known as roulette wheel selection. Parents are selected according to their fitness. All chromosomes in the population are placed in the roulette wheel. For example, if we have P chromosomes in the population, we will have P segments on the roulette wheel. The size of its segment depends on the fitness of a particular chromosome.

1. Sum up the fitness values of all chromosomes in the population.
2. Generate a random number between 0 and the sum of fitness values.
3. Select the chromosome whose fitness value added to the sum of the fitness values of the previous is greater than or equal to the random number.

Obviously, a chromosome with high fitness has a greater probabilty of being selected as a parent. This step will try to maintain good solution features to the next generation.

- **Crossover**

The order crossover using two parents and two crossover sites are selected randomly and the elements between the two selecting points in one of the parent are directly inherited by the offspring.

- **Mutation**

Mutation is the process applied to each offspring individually after the crossover. This operator creats new individual chromosome by a small change in a single individual chromosome by a random selection. In this thesis, the encoded SOS using the 2D binary string that the mutation is applied to a bit string. It sweeps down the bits and replace by randomly selected bit if the probability of the test passes.

3.2.3 Fitness function

In order to describe the details of the evaluated fitness function. This thesis uses the MIP optimization solver called CPLEX to evaluate the fitness value of GA. The setting of the CPLEX solver will be described below.

- **The optimization CPLEX solver**

CPLEX is an optimization software package. It is named for the simplex method and the C programming language. It was originally developed by Robert E. Bixby and distributed via CPLEX Optimization Inc. CPLEX can solve MIP problem and very large LP problems. Moreover, it has a modeling layer and is also available with several modeling systems like AIMMS, AMPL, GAMS IDE and OPL Development Studio. In this thesis, we develop a modeling language based on GAMS IDE and solve the model on CPLEX version 11.0.

- **Fitness evaluation**

As far as GA is concerned, it's better to have a higher fitness value to provide more opportunities to be chosen in breeding new chromosomes. In this thesis, the CPLEX solver has been used to solve the MIP using the SOS variables from GA which determines the largest score from the number of iterations.

At each transition, the value of computational iterations from GALMIP ($fitness\_score$) is subtracted from a standard fitness score ($standard\_fitness\_score$) which is obtained from the computational iterations of the MIP. The $fitness\_score$ higher than the $standard\_fitness\_score$ presents a better candidate of SOS (a strong gene) which will be stored into a text file. The GA fitness is measured from the subtraction of the computational iterations of MIP and the current computational iteration of GALMIP. We can describe the GA fitness with an equation as follow.

$$\text{Evaluate Fitness} = Standard\_fitness\_score - fitness\_score \tag{III.19}$$

CHAPTER IV

Experimental Design and Results

4.1 Experimental design

In this thesis, we design the layout as the instances with four distinct configurations in order to measure their performance capabilities. The outcomes have been carried out on a PC computer which has an Intel(R) Core (TM)2 Duo as a processer CPU and 2004 MB of memory. This experiment is simulated with 4, 5, 6 and 7 rooms, which are based on the following four distinct configurations;

1. linear configuration

2. rail configuration

3. connected wheel configuration

4. nested wheel configuration

See the figure 4.1.1 for the graphical representation of these four distinct patterns.

The minimum and maximum width and height of each room are defined to be between 5 and 10 meters. The boundary area is set on 100×100 square meters. Moreover, the weighted sum of u_1, u_2 and u_3 are equivalently set to 1.

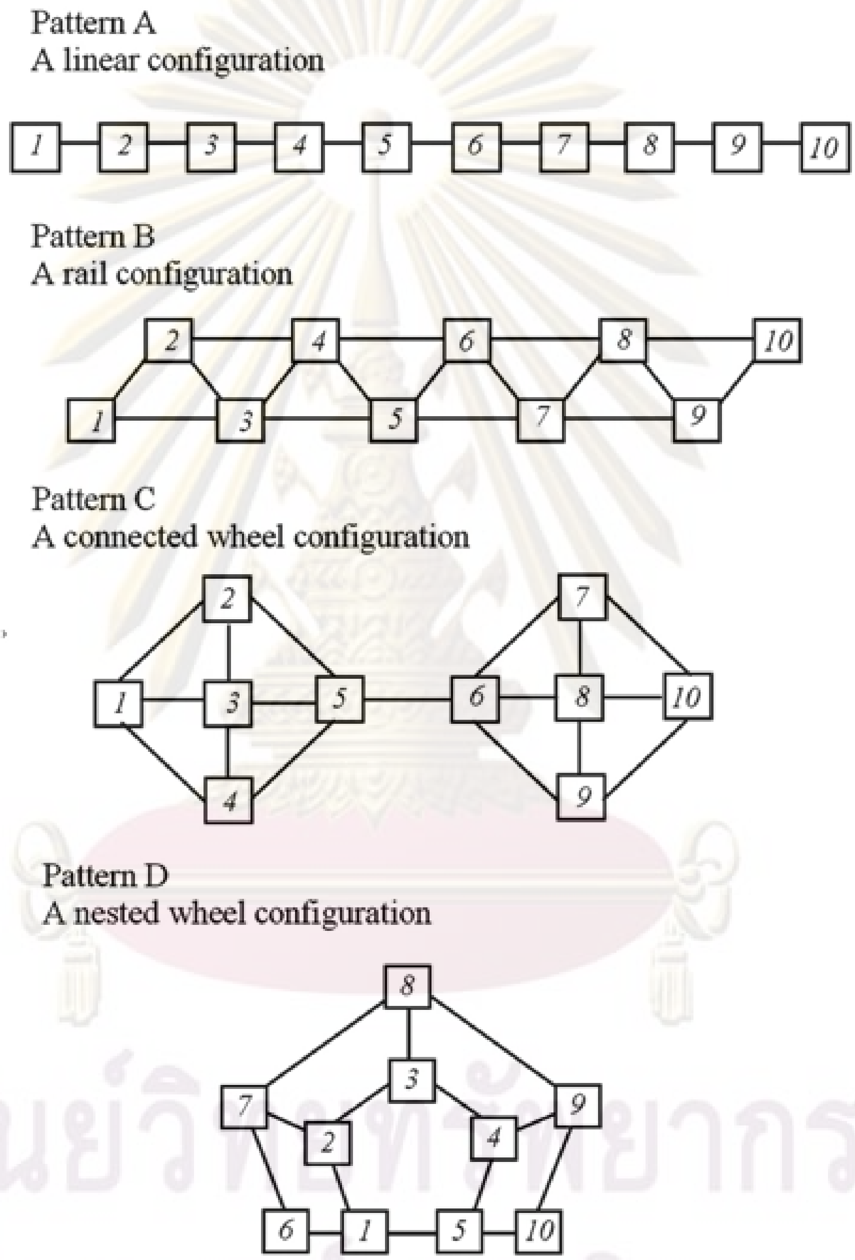

Figure 4.1.1: The distinct patterns A, B, C and D of 10 rooms

4.2 Parameters and design setting for a genetic algotithm

This section will covers the settings of parameters which are used in genetic operator in order to achieve a desireable solution and performance, it consists of parameters of population size, crossover probability and mutation probability.

In order to achieve the desirable results and performance, appropriated GA parameters have to be set. Several researchers have been trying to understand the complex interactions among the GA parameters and are trying to design them to fit into any world problems. In 1975, De Jong presented the GA parameters that have been adopted widely which it is known as "standard" settings with a population size of 50 to 100, a crossover probability of 0.9 and mutation probability of 0.001. However, these "standard "settings are not suitable for all problems. It depends on the nature of the function being evaluated and the way of encoding variables being used (Goldberg,1985;Hart and Below, 1991; Deb, 1999). Later in 2000, Lobo suggested using an appropriated GA parameter that determines by parameters trial and error.

Next, we will describe how to set each parameter in detail.

4.2.1 Population size

The population size parameter is a major factor in determining the quality of the solutions, so it has to be considered carefully. The population size depends on the nature of the problem needed; not too big and not too small. If the population is too small, the genetic algorithm may not explore enough the solution space to consistently find good solutions. On the other hand, if the population size is too large, the algorithm will waste unnecessary computational resources.

According to De Jong (1975), the appropriated size of population is usually in the range of 50 to 100. The layout design is experimented using the population size of 100.

4.2.2 Crossover and mutation probability

Crossover operator is important because it ensures good mixing of candidate solutions. The higher the crossover probability, the more promising solutions are mixed. A crossover probability of 1.0 indicates that the crossover process happen with all selected chromosomes. Golberg and holland advocated that the better results are achieved by the use of a high crossover probability in the range [0.8,1] and a low mutation probability in the range [0,0.01].

We use the common crossover of 0.9 and mutation 0.001 suggested by De jong(1975) and Goldberg (1985). For the reason being that the high levels of mutation are the most disruptive and also achieve the lowest levels of construction. The chance that a new candidate gene is found decreases. The performance of GA is not so influenced by these operators than the population sizes and generations.

4.2.3 The length of string representation

Before creating a string, binary variables p_{ij} and q_{ij} need to be ordered and given an index. The string will then be filled in by the index of those variables. In order to determine the maximum length of the chromosome, we need to find the posibility of connectivity between the room i and the room j which is identified by p_{ij} and q_{ij}. For example of 4 rooms we have 6 possible connectivities between each room. Therefore we have 6 variables of p_{ij} and 6 variables of q_{ij} to identify the connectivities, which consist of $\{p_{12}, p_{13}, q_{13}, p_{14}, p_{23}, p_{24}, p_{34}\}$ and $\{q_{12}, q_{13}, q_{13}, q_{14}, q_{23}, q_{24}, q_{34}\}$. The total results of a SOS variable consists of the combination of both variables p_{ij} and q_{ij}. Therefore, we can determine an SOS variable length in a candidate SOS using the equation,

$$C(n,r) = \frac{n!}{r! \times (n-r)!}, \qquad \text{(IV.1)}$$

where n is the number of rooms and r is the number of SOS variables used in the problem. Since there are only two binary variables used in each problem, the value of r is equal to 2 in this thesis.

4.2.4 Generations and stopping criterion

The stopping criterion are created by predefining the number of generations. The algorithm will stop when the number of generations is reached.

4.3 The result of MIP and GALMIP

In this section, the objective values and the number of iterations between MIP and GALMIP of 4-7 rooms among the distinct patterns of A, B, C and D are shown in the table 4.1.

Moreover, the experiment has been performed with population size of 10, generation iteration of 100, crossover probability of 0.9 and mutation probability of 0.001. see table 4.1.

Table 4.1: Iteration comparision between MIP and GALMIP

Room no.	Pattern	Objective value	MIP	GALMIP
4	A	25	1249	1116
	B	27	1523	1471
	C	28	1730	1600
	D	25	1256	1076
5	A	60	21470	19548
	B	60	15773	11119
	C	65	19264	17013
	D	60	21726	17684
6	A	100	296568	275767
	B	103	106968	73817
	C	115	204405	192828
	D	104	413618	215698
7	A	160	10376928	10012139
	B	170	705809	491427
	C	178	2059350	2014997
	D	161	6314838	3318656

Table 4.2: Iteration comparision of AL-MIP, MIP, AL-MIP+GA and GALMIP

Room no.	patterns	AL-MIP	MIP	AL-MIP+GA	GALMIP
4	A	1.10×10^3	1.25×10^3	4.53×10^2	1.12×10^3
	B	1.19×10^3	1.52×10^3	4.55×10^2	1.47×10^3
	C	1.39×10^3	1.73×10^3	5.29×10^2	1.60×10^3
	D	1.07×10^3	1.26×10^3	4.81×10^2	1.08×10^3
5	A	1.31×10^4	2.15×10^4	3.69×10^3	1.95×10^4
	B	8.88×10^4	1.58×10^4	2.41×10^3	1.11×10^4
	C	1.11×10^4	1.93×10^4	1.82×10^3	1.70×10^4
	D	1.52×10^4	2.17×10^4	3.81×10^3	1.80×10^4
6	A	1.84×10^5	2.96×10^5	2.78×10^4	2.76×10^5
	B	4.38×10^4	1.07×10^5	1.06×10^4	7.38×10^4
	C	6.11×10^4	2.04×10^5	5.40×10^4	1.93×10^5
	D	2.27×10^5	4.14×10^5	2.39×10^4	2.16×10^5
7	A	5.25×10^6	1.04×10^7	1.83×10^5	1.00×10^5
	B	1.76×10^5	7.06×10^5	4.08×10^4	4.91×10^4
	C	2.01×10^5	2.06×10^6	3.23×10^4	2.01×10^5
	D	1.11×10^6	6.31×10^6	1.04×10^5	4.32×10^5

Table 4.1 shows the objective value and number of iterations of each configuration between MIP and GALMIP. The four distinct configurations illustrate the various computational iterations. According to the table 4.1, the results demonstrate that they depend on the structural connectivity. For 4 rooms, all configurations have similar number of iterations while the number of iterations start to be different when the number of rooms is larger. From 5, 6 and 7 rooms,we can see that the computational iterations of rail configuration (Pattern B) and the connected wheel configuration (pattern C) have similar number of iterations while a linear configuration (pattern A) and a nested wheel configuration (pattern D) have quite far more iterations than both patterns B and C. Significantly, the linear configuration has higher computational iterations than the nested wheel configuration for 7 rooms. Besides, the linear configuration has more iterations than the rail configuration about 10 times, more than 3 times for the connected wheel configuration and almost 1 time for the nested wheel configuration. This illustrates that the structural connectivity matters. Moreover, the number of iterations of the rail configuraton is the least for every experiment.

Table 4.2 presents the comparision among the results of AL-MIP, MIP, AL-MIP+GA and GALMIP in term of the number of iterations, where AL-MIP and MIP are mathematical models, AL-MIP+GA and GALMIP are models using the application of Genetic Algorithm. AL-MIP and AL-MIP+GA are the models formulated by Kamol and Krung[21] while MIP and GALMIP are our model in this thesis.

According to the comparison between the number of iterations of AL-MIP and MIP, using MIP+GA achieves an average of 3 to 48 percentage gains comparing to the traditional MIP. For pattern A, the reduction is around 7-11 percentage. For pattern B, the reduction is around 3-30 percentage . For pattern C, the reduction is around 7-11 percentage. For pattern D, the reduction is around 14-48 percentage. For pattern B and D, the reduction percentage increases as the number of rooms increase. This trend is different for patterns A and B, the reductions of A and B are not stable as the problem size grows. Therefore, we can conclude that the reduction of iterations depends on structural connectivity. Since the depth from any two farthest nodes of pattern D has the least depth, the path to the solution of this pattern is shorter than the other patterns. GA can effectively find appropriate SOS than other patterns.

- **The results of Special Order Set and its application**

This part ilustrates the cadidate SOS obtained from our experiments which vary from 4-7 rooms of patterns A, B, C and D presented by table 4.3 to 4.6.

Moreover, we implemented the SOS candidate into different sizes of rooms, we discovered that it can be applied to rooms with the area sizes of $6 \times 12, 7 \times 14$ and 10×15 square meters which are shown in table 4.7, 4.8 and 4.9 .

Table 4.3: Illustrates 4 rooms candidate SOS variable p_{ij} and q_{ij}

Branching Order	Pattern A	Pattern B	Pattern C	Pattern D
1	q_{13}	q_{24}	q_{14}	q_{13}
2	p_{24}	q_{24}	q_{12}	p_{23}
3	q_{24}	q_{23}	p_{12}	p_{13}
4	p_{24}	q_{13}	p_{13}	p_{14}
5	q_{12}	p_{35}	p_{14}	q_{14}
6	p_{24}	p_{12}	q_{34}	q_{13}
7	q_{14}	q_{14}	p_{24}	p_{13}
8	p_{23}	p_{23}		q_{24}
9	q_{24}	p_{23}		q_{23}
10	p_{14}	q_{14}		q_{24}
11	p_{14}	p_{14}		p_{24}
12	q_{14}	q_{34}		
13		p_{34}		
14				
15				

Table 4.4: Illustrates 5 rooms candidate SOS variable p_{ij} and q_{ij}

Branching Order	Pattern A	Pattern B	Pattern C	Pattern D
1	p_{35}	q_{25}	p_{34}	q_{14}
2	q_{14}	p_{15}	p_{25}	q_{12}
3	q_{35}	q_{35}	p_{12}	q_{24}
4	p_{13}	q_{35}	p_{14}	p_{12}
5	p_{13}	q_{14}	q_{13}	p_{14}
6	q_{25}	p_{35}	p_{13}	q_{34}
7	p_{25}	p_{13}	q_{24}	q_{13}
8	p_{13}	p_{35}	q_{15}	p_{45}
9	q_{13}	q_{25}	q_{25}	p_{45}
10	p_{12}	p_{35}	q_{35}	q_{14}
11	q_{15}	q_{15}	p_{35}	p_{34}
12	q_{15}	q_{25}	p_{34}	q_{34}
13	q_{32}	q_{15}	p_{25}	q_{35}
14		p_{13}	p_{45}	q_{35}
15		q_{14}	p_{35}	
16		p_{12}		
17		q_{15}		
18		q_{14}		

Table 4.5: Illustrates 6 rooms candidate SOS variable p_{ij} and q_{ij}

Branching Order	Pattern A	Pattern B	Pattern C	Pattern D
1	p_{12}	p_{26}	p_{46}	q_{23}
2	p_{23}	p_{23}	q_{46}	q_{14}
3	p_{34}	q_{46}	p_{26}	q_{13}
4	q_{36}	q_{36}	q_{12}	p_{35}
5	q_{13}	q_{13}	p_{45}	q_{12}
6	q_{15}	p_{16}	q_{26}	p_{46}
7	q_{13}	q_{13}	p_{36}	q_{36}
8	p_{35}	p_{15}	p_{16}	q_{13}
9	p_{24}	q_{45}	p_{24}	q_{13}
10	q_{16}	p_{12}	q_{24}	q_{15}
11	p_{36}	p_{36}	q_{26}	q_{36}
12	q_{24}	p_{35}	q_{45}	q_{35}
13	p_{14}	p_{45}	q_{25}	q_{25}
14	q_{36}	p_{46}	q_{14}	p_{13}
15	q_{15}	p_{12}	q_{16}	p_{25}
16	q_{35}	q_{36}	p_{34}	p_{15}
17	p_{45}	q_{15}	q_{15}	q_{56}
18	p_{56}	p_{25}	q_{13}	p_{12}
19	q_{14}	q_{25}	q_{36}	q_{13}
20	q_{25}	q_{26}	p_{34}	q_{16}
21	q_{35}	q_{25}	q_{45}	p_{13}
22	p_{56}	q_{14}	q_{24}	p_{46}
23	q_{16}	q_{16}	q_{15}	q_{14}

Branching Order	Pattern A	Pattern B	Pattern C	Pattern D
24	p_{46}	p_{13}	p_{15}	p_{46}
25	p_{14}	p_{13}	q_{46}	p_{36}
26	q_{46}	q_{13}	q_{35}	q_{56}
27	q_{36}		q_{14}	q_{36}
28			q_{35}	p_{16}
29				q_{46}

Table 4.6: Illustrates 7 rooms candidate SOS variable p_{ij} and q_{ij}

Branching Order	Pattern A	Pattern B	Pattern C	Pattern D
1	p_{12}	p_{14}	p_{14}	q_{25}
2	p_{23}	p_{23}	q_{27}	q_{17}
3	p_{34}	q_{46}	p_{35}	q_{16}
4	q_{36}	q_{32}	q_{16}	p_{34}
5	q_{17}	q_{26}	p_{45}	q_{23}
6	q_{15}	p_{16}	q_{14}	p_{47}
7	q_{13}	q_{27}	p_{37}	q_{36}
8	p_{35}	p_{35}	p_{16}	q_{12}
9	p_{24}	q_{45}	p_{67}	q_{17}
10	q_{16}	p_{14}	q_{34}	q_{24}
11	p_{36}	p_{16}	q_{23}	q_{26}
12	q_{27}	p_{36}	q_{16}	q_{37}
13	p_{14}	p_{45}	q_{17}	q_{25}
14	q_{36}	p_{46}	q_{15}	p_{67}
15	q_{15}	p_{17}	q_{16}	p_{46}
16	q_{35}	q_{36}	p_{34}	p_{46}
17	p_{45}	q_{15}	q_{34}	q_{17}
18	p_{57}	p_{24}	q_{25}	p_{12}
19	q_{14}	q_{13}	q_{15}	q_{13}
20	q_{25}	q_{27}	p_{34}	q_{16}
21	q_{35}	q_{24}	q_{46}	p_{15}
22	p_{56}	q_{16}	q_{27}	p_{46}
23	q_{16}	q_{14}	q_{15}	q_{26}

Branching Order	Pattern A	Pattern B	Pattern C	Pattern D
24	p_{46}	p_{23}	p_{15}	p_{23}
25	p_{14}	p_{56}	q_{46}	p_{34}
26	q_{46}	q_{34}	q_{35}	q_{14}
27	q_{36}	q_{12}	q_{14}	q_{25}
28		q_{21}	q_{37}	p_{27}
29		q_{37}		q_{17}
30				q_{13}
31				q_{25}

Table 4.7: The result of applying the SOS candidate with the area room size of 6×12 sqm

Room no.	Pattern	Objective value	MIP	GALMIP
4	A	30	1310	1120
	B	32	1760	1510
	C	33	1619	1523
	D	30	1269	1093
5	A	72	25629	24321
	B	72	16594	9719
	C	78	22827	20311
	D	72	23619	19817
6	A	120	303890	287353
	B	123	126562	72445
	C	137	178424	135642
	D	124	202209	195677
7	A	192	4693558	3885060
	B	204	693251	592544
	C	213	1060839	758441
	D	193	4505278	4035278

Table 4.8: The result of applying the SOS candidate with the area room size of 7×14 sqm

Room no.	Pattern	Objective value	MIP	GALMIP
4	A	35	1254	1090
	B	37	1714	1450
	C	38	1527	1430
	D	35	1184	1027
5	A	84	28214	15296
	B	84	18837	9750
	C	91	19340	12505
	D	84	23439	17408
6	A	140	241198	177238
	B	143	117644	71839
	C	159	249450	92594
	D	144	247578	201746
7	A	224	1338352	9095641
	B	238	1223413	1012564
	C	248	1915401	1021142
	D	225	3080527	2388756

Table 4.9: The result of applying the SOS candidate with the area room size of 10×15 sqm

Room no.	Pattern	Objective value	MIP	GALMIP
4	A	50	1235	1100
	B	52	1654	1493
	C	53	1534	1480
	D	50	1338	1229
5	A	120	24750	20215
	B	120	16348	9966
	C	130	21535	19457
	D	120	21409	17185
6	A	200	334583	292198
	B	203	76541	71487
	C	225	136069	105919
	D	204	277729	218869
7	A	320	8897503	7200259
	B	340	922959	759145
	C	353	1338383	994356
	D	321	1956657	1023065

CHAPTER V

CONCLUSION AND SUGGESTION

We use the mathematical model based on the model of Kamol and Krung in 2005 [21] in order to investigate the model and compare the number of iterations. In [21], they used the point at the top left corner of the room to represent the reference point. In a practical point at the center of the room is more common for architects to design the buiding. In this thesis, we therefore change the reference point to the center point referring to the work of Michealek in [22]. The model using the reference point at the center called MIP. Then, we propose the model called GALMIP in order to reduce the computational iterations that are produced from MIP model.

5.1 Conclusion

From our experiments, the results can be concluded as follows.

1. The MIP model is easy to formulate the layout design problem.

2. The SOS candidate obtained from learning methodology can reduce the search space and therefore the number of computational iterations are reduced. Similarly, we can conclude that GALMIP has a better performance than MIP in terms of the number of iterations.

3. The reduction of iterations depends on the structural connectivity.

5.2 Application of the candidate SOS

Since the size of each room can be varied in real situation of layout designing for building, we attempt to apply our candidate SOS to other room sizes. We have found that the candidate SOS obtained from GA can be applied with some patterns, meaning that the number of iterations of MIP can be reduced by using the same candidate SOS. Nevertheless, the candidate can not be applied to every room size. Therefore, we can conclude that the binary variables p_{ij} and q_{ij} might not be suitable to be learned by GA. We might need to find other relations that is suitable for GA.

5.3 Suggestion

Our approach can be further developed as a possible perspective direction to improve an architectural layout design problem as the following suggests.

1. We can add new objectives or constraints to the model to improve optimization behavior or quality of the layouts.

2. Since the shape of the layouts is not restricted, we can generalize them more by using more complex shapes that are non-rectangular.

3. Since our thesis concentrates on only one floor of the layout design, we can improve the model based on muliple-level floor in order to fit the layout design for the real architect.

REFERENCES

[1] D. F. Wong and C. L. Liu. A new algorithm for floorplan design. *Proceedings of the 23rd ACM/IEEE Design Automation Conference* (June 1986) : 101-107.

[2] Y. C. Chang, Y. W. Chang, G. M. Wu and S. W. Wu. B*-trees:a new representation for non-slicing floorplans. *Proceedings of 37th Design Autiomation Conference* (June 2000) : 458-463.

[3] J. M. Lin, and Y. W. Chang. TGG : a transitive closure graph-based representation for non-slicing floorplans. *Proceedings of the 38th annual Design Automation Conference* (June 2001) : 764-769

[4] E. F. Y. Young, C. C. N. Chu, and Z. C. Shen. Twin binary sequences : a nonredundant representation for general nonslicing floorplan. *IEEE Transactions on Computer-Aided Design of Intergrated Circuits and System* 22 (April 2003) : 457-459.

[5] X. Hong,G. Huang,Y. Cai, J. Gu, S. Dong, and C. K. Cheng. Corner block list representation and its application to floorplan optimization. *IEEE Transactions on Curcuit and Systems II:Express and Briefs* 51 (May 2004) : 228-233.

[6] X. Hong,G. Huang,Y. Cai, J. Gu, S. Dong, and C. K. Cheng. Corner block list : an effective and efficient topological representation of nonslicing floorplan. *Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design* (2000) : 8-12.

[7] H. Onodera, Y. Taniguchi, and K. Tamaru. Branch-and-bound placement for building block layout. *Proceedings of the 28th ACM/IEEE Design Automation Conference* (1991) : 433 - 439.

[8] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Ikjatani. Rectangular packing-based module placement. *Proceedings of IEEE/ACM international conference on Computer-aided design* (November 1995) : 472-479.

[9] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani. Module placernent on BSG-structure and IC layout application. *Proceedings of IEEE/ACM international conference on Computer-aided design* (1996) : 484-491.

[10] P.N. Guo ,T. Takahashi ,C.-K. Cheng ,T. Yoshimura. Floorplanning Using a Tree Representation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 20 (February 2001) : 281289.

[11] P.N. Guo ,C.-K. Cheng ,T. Yoshimura. An O-tree Representation of Non-slicing floorplan and its applications. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (1999) : 268-273.

[12] G. C. Armour and E. S. Buffa. A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities. *Management Science* 9 (January 1963) : 294-309.

[13] P. H. Lavin. Use of graphs to decide the optimum layout of buildings. *Arcitects' Journal* 14 (October 1964) : 809-815.

[14] D. J. Van Camp, M. W. Carter, and A. Vannelli. A Nonlinear Optimization Approach for Solving Facility Layout Problems. *European Journal of Operational Research* 57 (1992) : 174-189.

[15] D. M. Tate, A.E. Smith. Unequal-area facility layout by genetic search. *IIE Transactions* 27 (April 1994) : 465-472.

[16] T. D. Thelma and P.M. Pardalos.Simulated annealing and genetic algorithms for the facility layout problem: a survey. *Computational Optimization and Applications* 7 (January 1997) : 111-126.

[17] R. D. Meller, V. Narayanan, and P. H. Vance. Optimal facility layout design.*Operation Research* 23 (1998) : 117.

[18] I-Cheng Yeh. Construction site layout using annealed neural network. *Journal of Computing in Civil Engineering* 9 (July 1995) : 205 -208.

[19] R. Bausys and I. Pankrasovaite. Optimization of architectural layout by improved genetic algorithm. *Journal of civil engineering and management* 11 (2005) : 13-21.

[20] C. J. Bloch and R. Krishnamurti. The counting of rectangular dissections. *Environment and Planning* 2 (1978) : 207-214.

[21] K. Keatruangkamala and K. Sinapiromsaran. Optimization Architectural Layout Design via Mixed Integer Programming. *Proceeding in CAADFutures* (2005) : 175-184.

[22] J. Michalek and P. Y. Papalambros. Interactive layout design optimization. *Engineering Optimization* 34 (February 2002) : 461-484.

[23] K. Keatruangkamala and K. Sinapiromsaran. Mixed Integer Programming Model with Non-circular and Guided Constraints for Architectural Layout Design Optimization. *Songklanakarin Journal of Science and Technology* (2005) : 3-29.

[24] J. Michalek and P. Y. Papalambros. Architectural layout design optimization. *Engineering Optimization* 34 (February 2002) : 461-484.

APPENDIX

Appendix

GAMS IDE model for MIP

This appendix section presents the GAMS IDE model for MIP methodology.

```
$ontext
________________________________________________
GAMS IDE model developed by Thitiya Theparot
________________________________________________
$Offtext
set          ROOM;
             ALIAS(ROOM,i);
             ALIAS(ROOM,j);
             ALIAS(ROOM,k);
set          LINK(i,j)
             CONNECT(i,j)
PARAMETERS
             DELTA
             PanelWidth
             PanelHeight
             Wmin(i)
             Wmax(i)
             Hmin(i)
             Hmax(i);
PARAMETER          WeightLeftCorner(i);
PARAMETERS         WeightMinDistance
                   WeightMaxArea;
VARIABLE z;
```

POSITIVE VARIABLES

zx(i,j)

zy(i,j)

za(i);

POSITIVE VARIABLES

x(i)

y(i)

E(i)

N(i);

BINARY VARIABLES

p(i,j)

q(i,j);

E.lo(i) = Wmin(i)/2;

E.up(i) = Wmax(i)/2;

N.lo(i) = Hmin(i)/2;

N.up(i) = Hmax(i)/2;

EQUATIONS

obj_Min

za_width(i)

za_height(i)

position_x(i)

position_y(i)

abs_plus_x(i,j)

abs_minus_x(i,j)

abs_plus_y(i,j)

abs_minus_y(i,j)

widthsize(i)

heightsize(i)

force_ij_left(i,j)

force_ij_bottom(i,j)

force_ij_right(i,j)

force_ij_top(i,j)

join_ij_left(i,j)

join_ij_bottom(i,j)

join_ij_right(i,j)

join_ij_top(i,j)

overlap_Up(i,j)

overlap_Down(i,j)

overlap_Left(i,j)

overlap_Right(i,j);

$$\begin{aligned}\text{obj\_Min..} = e = \ & \text{sum}(i, \text{WeightLeftCorner}(i) * (x(i) + y(i))) \\ & + \text{WeightMinDistance} * \text{sum}(\text{LINK}(i,j), zx(i,j) + zy(i,j)) \\ & - \text{WeightMaxArea} * \text{sum}(i, za(i));\end{aligned}$$

za_width(i).. $za(i) = l = 2 * E(i);$

za_height(i).. $za(i) = l = 2 * N(i);$

position_x(i).. $x(i) - E(i) = g = 0;$

position_y(i).. $y(i) - N(i) = g = 0;$

abs_plus_x(LINK(i,j)).. $x(i) - x(j) = l = zx(i,j);$

abs_minus_x(LINK(i,j)).. $x(j) - x(i) = l = zx(i,j);$

abs_plus_y(LINK(i,j)).. $y(i) - y(j) = l = zy(i,j);$

abs_minus_y(LINK(i,j)).. $y(j) - y(i) = l = zy(i,j);$

widthsize(i).. $x(i) + E(i) = l =$ PanelWidth;

heightsize(i).. $y(i) + N(i) = l =$ PanelHeight;

force_ij_left(LINK(i,j)).. $x(i) + E(i) = l = x(j) - E(j)$
+ PanelWidth*$(p(i,j) + q(i,j));$

force_ij_bottom(LINK(i,j)).. $y(j) + N(j) = l = y(i) - N(i)$
+ PanelHeight*$(1{+}p(i,j){-}q(i,j));$

force_ij_right(LINK(i,j)).. $x(j) + E(j) = l = x(i) - E(i)$
+ PanelWidth*$(1{-}p(i,j){+}q(i,j));$

force_ij_top(LINK(i,j)).. $y(i) + N(i) = l = y(j) - N(j)$
+ PanelHeight*$(2{-}p(i,j){-}q(i,j));$

Overlap_Up(CONNECT(i,j)).. $0 = g = y(i) - N(i)$ + DELTA - $y(j) - N(j)$
- PanelHeight*$(q(i,j));$

Overlap_Down(CONNECT(i,j)).. $0 = g = y(j) - N(j)$ + DELTA - $y(i) - N(i)$
- PanelHeight*$(q(i,j));$

Overlap_Left(CONNECT(i,j)).. $0 = g = x(i) - E(i)$ + DELTA - $x(j) - E(j)$
- PanelHeight*$(1 - q(i,j));$

Overlap_Right(CONNECT(i,j)).. $0 = g = x(j) - E(j)$ + DELTA - $x(i) - E(i)$
- PanelHeight*$(1 - q(i,j));$

VITAE

Name Thitiya Theparod

Date of Birth 7 August 1984

Place of Birth Kalasin, Thailand

Education B.Sc. (Mathematics), Khonkaen University, 2006