

Factory Floor Planning Optimization using Metaheuristics

SIAH KUOK LIM

MASTER OF COMPUTER SCIENCE

FACULTY OF ENGINEERING AND SCIENCE
UNIVERSITI TUNKU ABDUL RAHMAN
JULY 2013

**FACTORY FLOOR PLANNING OPTIMIZATION USING
METAHEURISTICS**

By

SIAH KUOK LIM

A thesis submitted to the
Department of Internet Engineering and Computer Science,
Faculty of Engineering and Science,
Universiti Tunku Abdul Rahman,
in partial fulfillment of the requirements for the degree of
Master of Computer Science
July 2013

ABSTRACT

FACTORY FLOOR PLANNING OPTIMIZATION USING METAHEURISTICS

SIAH KUOK LIM

The factory operational performance is largely influenced by the layout of manufacturing facilities. Factory layout is the science of arranging equipment, space, and activities subject to the manufacturing operational rules and policy for optimal line execution. Apart from maximizing the space utilization, it is a challenging task to come up a feasible design optimizing all other critical areas such as (1) process and work flow, (2) proper allocation of space and resources, (3) ease of access to supply and materials, and (4) manning ratio feasibility.

Generating a feasible layout incorporating all the constraints is a classic “NP-Hard problem”. In Intel, the layout industrial engineers manually perform “what-if” exercises based on their best judgment to arrive at a better solution among all other numerous possibilities. Not only is the method labor intensive, it does not necessary guarantee them an optimal solution. In addition, even the use of normal automation program to obtain the best solution by generating all possible combination is impossible. Therefore, a scientific way of finding the optimize solution will prove helpful. This model will be able to help suggest a feasible layout or evaluate an existing layout based on the fitness value.

To search for the optimal solution, this research uses Fast Simulated Annealing (FSA), Genetic Algorithm (GA) with conditional crossover and Genetic Programming (GP). All three search algorithms are stochastic strategy for searching an optimal state instead of exhaustively searching all possible combinations. With the inputs and configuration parameters such machines’ dimension, quantity etc., the model will iterate until a feasible layout (optimal solution) is found. The best layout searched will be kept in memory and will be replaced until a better layout is found in later iteration. As more iteration had run the confidence level of obtaining a more optimized layout increases.

This research model was successfully used for “what-if” scenario analysis for NCO6 factory, layout design for new catalyst machines. The model was used to generate the best layout for catalyst relocation based on changes impact to the existing layout design as well as cost for changes. With the pros and cons of each layout suggested, the result had help in decision making for catalyst re-layout effort. Changes are made as suggested in the results analysis. As seen from the promising result from this model, it was used to simulate various optimal full factory layout options for Intel Kulim Microprocessor and Chipset Organization (KMCO) and the upcoming A9 factory in Intel Vietnam.

ACKNOWLEDGEMENT

Foremost, I would like to express my sincere gratitude to my advisor, Associate Professor Tay Yong Haur from Computer Vision and Intelligent Systems (CVIS) group for the continuous support of my Master study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Master study.

Besides my advisor, I would like to thank the rest of my thesis committee for their encouragement, insightful comments, and hard questions. This thesis is more comprehensive and refined with their tactful advice.

In particular, I am grateful to Dr. Lee Wen Jau for enlightening me the first glance of research. He had supported the whole collaboration efforts between Intel Technology Malaysia and Universiti Tunku Abdul Rahman (UTAR) as my Master research.

A special thanks to Teoh Choo Wee and Lim Chen Beng, layout engineers from Intel Corporation, Kulim to provide the domain knowledge in factory floor planning. They helped in validation and provided valuable feedbacks to improve the model. With their expertise in this domain, only then model results are qualified for factory management endorsement.

Last but not least, I would like to thank my family: my parents Siah See Wah and Lim Bee Leng, for giving birth to me at the first place and supporting me spiritually throughout my life.

APPROVAL SHEET

This dissertation/thesis entitled “**FACTORY FLOOR PLANNING OPTIMIZATION USING METAHEURISTICS**” was prepared by SIAH KUOK LIM and submitted as partial fulfillment of the requirements for the degree of Master of Computer Science at Universiti Tunku Abdul Rahman.

Approved by:

(Assoc. Prof. Dr. Tay Yong Haur)

Date:.....

Supervisor

Department of Internet Engineering and Computer Science

Faculty of Engineering and Science

Universiti Tunku Abdul Rahman

Date: 19 JUNE 2013

It is hereby certified *SIAH KUOK LIM* (ID No: *06UEM02019*)
has completed this thesis entitled “Factory Floor Planning Optimization using
Metaheuristics” under the supervision of Dr. Tay Yong Haur from the Department
of Internet Engineering and Computer Science, Faculty of Engineering and
Science.

I understand that University will upload softcopy of my thesis in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

(SIAH KUOK LIM)

DECLARATION

I hereby declare that the dissertation is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTAR or other institutions.

Name SIAH KUOK LIM

Date 12 July 2013

TABLE OF CONTENTS

	Page
ABSTRACT	ii
ACKNOWLEDGEMENTS	iii
APPROVAL SHEET	iv
SUBMISSION SHEET	v
DECLARATION	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	ix
LIST OF FIGURES	x
LIST OF ABBREVIATIONS	xii
 CHAPTER	
 1.0 INTRODUCTION	 1
1.1 Factory Floor Planning Criteria	2
1.1.1 Optimization Goals	2
1.1.2 Optimization Constraints	4
1.2 Relevant Research	5
1.3 Rationale	5
1.4 Objectives	6
1.5 Thesis Outline	7
 2.0 LITERATURE REVIEW	 8
2.1 Overview of Facility Layout Problem (FLP)	8
2.2 Metaheuristics Approach to FLP	10
2.2.1 FLP Approach Based on Simulated Annealing (SA)	10
2.2.2 FLP Approach Based on Genetic Algorithm (GA)	12
2.2.3 FLP Approach Based on Genetic Programming (GP)	14
 3.0 APPROACH AND ALGORITHMS	 19
3.1 Model Overview	19
3.2 Layout Representation using B*-Tree	19
3.3 Optimization Algorithm	21
3.4 Fast Simulated Annealing (FSA)	21
3.5 Genetic Algorithm (GA)	24
3.5.1 GA Crossover	27
3.5.2 GA Mutation	28
3.6 Genetic Programming (GP)	29
3.6.1 GP Crossover	31
3.6.2 GP Mutation	32
3.7 Optimization Goal	32
3.7.1 Calculate Space Utilization and Overlap Placement	33
3.7.2 Calculate Material Flow	34
3.8 Obtaining Manning Ratio Arrangement	35
 4.0 DATA PREPARATION	 38

4.1	Case Studies Overview	38
4.2	Case Study 1: Mock-up Data for Model Validation	38
4.3	Case Study 2: Real Industrial Data	40
5.0	EXPERIMENTAL RESULTS AND ANALYSIS	44
5.1	Experiments Overview	44
5.2	Case Study 1: Results and Analysis	44
5.3	Real industrial factory Retrofits	51
5.4	Experiments Results Summary	59
6.0	CONCLUSIONS AND FUTURE WORKS	61
6.1	Conclusions	61
6.2	Future Works	62

LIST OF TABLES

Table		Page
4.1	Scaling factor configuration setting	38
4.2	Case study 1 configuration setting	40
4.3	Input setting for experiment 6	41
4.4	Input setting for experiment 7	42
4.5	Input setting for experiment 8	43
5.1	Experiment 1 results with $b = 1$ and $c = 0$	45
5.2	Experiment 2 results with $b = 0$ and $c = 1$	46
5.3	Experiment 3 results with $b = 1$ and $c = 179$	47
5.4	Experiment 4 results with $b = 1$ and $c = 89.5$	48
5.5	Experiment 5 results with $b = 1$ and $c = 268.5$	49
5.6	GA benchmark result using different crossover rate	54
5.7	GP benchmark result using different crossover rate	55
5.8	Experiment 6 performance summary	56
5.9	Experiment 7 performance summary	58

LIST OF FIGURES

Figures	Page
2.1 Expression tree	16
3.1 Top view of a factory layout	20
3.2 B*-tree representing the placement	20
3.3 Three stages of FSA, temperature VS search time	22
3.4 Genetic Algorithm Structure	26
3.5 GA: Swapping between parent 1 and parent 2	27
3.6 Bad offspring generated from crossover	28
3.7 GP expression tree	31
3.8 GP: Swapping between parent 1 and parent 2	32
3.9 Factory layout design	33
3.10 Matrices representation of factory layout	34
3.11 Material flow (based on Euclidean distance)	34
3.12 Four machines and an operator cross box arrangement	36
3.13 Arrangement that support flexibility manning ratio of 1 operator to control 6 machines	36
4.1 Sample data of a perfect square	39
4.2 Manual suggested layout by engineers	42
5.1 Experiment 6 result using FSA	52
5.2 Search trend of the three stages FSA; fitness value VS iteration (time)	53
5.3 Experiment 6 result using GA	53
5.4 Experiment 6 result using GP	55
5.5 Experiment 7 result using FSA, GA and GP	56

5.6	Existing NCO6, Penang factory layout	59
5.7	NCO6, Penang suggested factory layout	59

LIST OF ABBREVIATIONS

NP	Non-deterministic polynomial time
FLP	Facility Layout Problem
SA	Simulated Annealing
FSA	Fast Simulated Annealing
GA	Genetic Algorithm
GP	Genetic Programming
OP	Overlapping Placement
OS	Occupied Space
WS	waste space
SU	Space Utilization
MF	Material Flow
FOL	Front of Line
EOL	End of Line

CHAPTER 1.0

INTRODUCTION

Laying out a factory involves deciding where to put all the facilities, machines, equipment and staff in the manufacturing operation. Layout determines the way in which materials and other materials flow through the operation. Relatively small changes in the position of a machine in a factory can affect the flow of materials considerably. This in turn can affect the costs and effectiveness of the overall manufacturing operation. Getting it wrong can lead to inefficiency, inflexibility, large volumes of inventory and work in progress, high costs and inefficient space (Aleisa, 2005). Changing a layout can be expensive and difficult, so it is best to get it right first time.

The first decision is to determine the type of manufacturing operation that must be accommodated. Intel Assembly and Test Malaysia is a semiconductor manufacturing. All products go through the same process flow, assembly to testing and packaging. Hence this research, we follow the practice of process layout. Process layout arrange similar manufacturing processes (assembly, test area and packaging) are located together to improve utilisation.

It is a big challenge and time consuming for engineer to come out with an optimize factory layout. Layout facilities optimization is categorized as NP-hard problem in complexity theory (H'astad, 2003). As a result, this project requires a high degree of difficulties to derive an optimal solution manually. This will further elaborate in the literature review.

1.1 Factory Floor Planning Criteria

The major objective, when designing the factory layout is to design a physical arrangement that most economically achieves the required output quantity and quality. Achieving the required output (Aleisa, 2005), involves the improvement of:

- process and work flow
- proper allocation of space and resources
- easier access to supplies and materials
- plant efficiency increase
- maximize the use of space

1.1.1 Optimization Goals

In this model, there are two optimization goals. Each goal is an evaluation method which returns the fitness of the layout respective to that particular goal. The priority of each goal is governed by a constant number as the scaling factor. The sum of all the goals multiplied by its own scaling factor will be the total fitness of the whole state. The objective function in this case is minimizing the total fitness of the state. The optimization goals are stated as below:

- Space Utilization:

Different machines will have different dimension in term of length and width. The height of the machine is ignored because machines cannot be stacked on top of each other. Hence, this research's result is a two

dimensions (2D) optimized factory layout of a top down view. An optimized layout will need to fit these different machines in the most space-utilized way.

- **Material Flow:**

Material transfer from an operation area to another is critical to achieve the minimum travelling distance for an operator between two locations. This will result in better utilization of resources (operators/trolleys), e.g. finished units to the exit.

- **Arrangement to Accommodate Flexible Manning Ratio:**

Arranging machines with their control facing the opposite machine control (face-to-face) will ease operator of operating more machines. Machines arrangement must have the shortest travelling distance from one machine control to another and give a wide view sight of all machines control. In a dynamic environment of a factory, manning ratio can be increased or decreased depending on the available resources and run rate. Hence, the arrangement of the machines is critical in order to accommodate these dynamic changes.

- **Overlapping placement avoidance:**

In a factory environment, there are many existing areas that can't be further rearranged or having overlapping placements. For example, existing areas such as meeting rooms, power supply areas, future expansion areas or aisle path should not place any machines, work-in-progress lots or routes.

1.1.2 Optimization Constraints

Optimization constraints are the rules of governing the solution which never generate a layout that conflict with the constraints. Constraints do not have any scaling factor as goal, it is modelled as rules imposing any states must follow all the constraints defined. The optimization constraints are stated as below:

- Machine clearance:

Machine clearance is a predefined distance gap of machines placed side by side, front to front, or back to back. The predefined gap is for module engineers' movement during maintenance, wiring and facilities installation, safety issue and reduces the heat generated from the machines.

- Implement route in between different operations:

Routes in between operations are critical for lots movement, machine relocation and safety issue for emergency escape. The main route is defined as the aisle path located in the middle of the factory which allows feasible movement of the whole factory. Unlike machines clearance, routes have larger gap and the main purpose is for factory movement activities.

- Same machines group together:

Same machines are important to get placed together due the fact that it demands the same facilities such as poles, ventilation and wirings. Splitting same machine into two different areas will create unnecessary

facilities changes, infeasible factory material flow and infeasible
manning ratio layout

1.2 Relevant Research

Related research for facility layout problem (FLP) for factory floor-planning had been done and tested in the past recent years. There were many techniques such as Genetic Algorithm (Lu, 2008) Simulated Annealing (Laursen, 1993), Fuzzy Logic, Ant Colony optimization and etc. being researched for FLP. However, techniques such as Genetic Algorithm and Simulated Annealing are the more common techniques that used by most researchers for facilities layout optimization (Aleisa, 2005). A survey on these two techniques will be discussed further in literature review section.

The literature findings regarding related algorithm and techniques explored to tackle FLP. Referring to (Chen, 2006) research, the approach of using FSA and B*-tree for a NP-hard problem yield a faster and stable convergence to desired solutions. B*-tree structure is used in Genetic Algorithm (Lu, 2008) and Genetic Programming (Jaime, 1996) studies.

1.3 Rationale

This research aims to derive an optimal solution for factory layout design. It is a collaboration effort done with Intel Assembly and Test Malaysia, PG6 factory in Penang. All data and requirements are obtained from engineers based on Intel safety guidelines for factory design. The

requirements include true machines dimension with clearance space, factory design constraints, setup for optimization goals priority and design specification.

B*-tree were used as representation for a layout design (Chang, 2000). Different representation methods were use due the different search algorithms studied in this research. Three models were developed using Fast Simulated Annealing, Genetic Algorithm and Genetic Programming search. All these search algorithms developed using VB.NET.

Experiments are conducted using Intel Manufacturing Factory in NCO6, Penang NetComm and Chipset layout design (encrypted data). Data are real machines dimension for assembly and test manufacturing. Solutions derived from the models are verified by layout engineers. All iterations ran converging to these solutions are used as benchmark data points against three search algorithms.

1.4 Objectives

There are two main objectives for this research collaboration with Intel Manufacturing. These objectives are as below:

1. Provides a clear way to scientifically measure the quality of a layout solution for Intel Manufacturing.
2. Observe and benchmark against different search algorithms, FSA, GA and GP for layout optimization.

1.5 Thesis Outline

This thesis structure will be separated as follows:

Literature review: We discuss on the nature of facilities layout problem and the considerations of an optimal factory layout. It also surveys studies on simulated annealing, fast simulated annealing, genetic algorithm and genetic programming algorithms that were used for solving this problem.

Approach & Algorithm: In this chapter we shall discuss in depth of the model design. This includes the optimization goals and constraints in general for all three algorithms. Three algorithms design being introduced in our model are FSA, GA and GP. Each algorithm has its respective layout representation method.

Data preparation: This chapter serves as the introduction of the data used for our experiments. The structure of our experiments being carried out is defined in details in this chapter. There will be case study 1 using mock-up data and case study 2 using real industrial data and problem statement.

Experiments & Results: Results of our experiments carried out are shown here. Results from case study 1 serve as model validation and observation purposes while case study 2 for industrial solution and algorithms benchmarking.

Conclusions and Future Works: Conclusion from our analysis results of this research based on the experiments' results. To make the model comprehensive, we will discuss on the extension works for this research. This touches the area where the model still lacking and algorithm performance issue to be tackle.

CHAPTER 2.0

LITERATURE REVIEW

2.1 Overview of Facility Layout Problem (FLP)

Facility Layout Problem (FLP) is a study of combinatorial optimization problem which arises in a variety of problems such as layout design of factory, hospitals, schools, and airports; printed circuit board design, backboard wiring problems, typewriters, warehouses, hydraulic turbine design, etc. (Singh, 2006). FLP has been generally formulated as a Quadratic Assignment Problem (QAP) introduced by (Koopmans, 1957) which is NP-hard (Garey, 1979), (Kusiak, 1987)

In complexity theory (Laursen, 1993), NP-hard problems are the most difficult problems in NP (non-deterministic polynomial time) in the sense that they are the smallest subclass of NP that could conceivably remain outside of P, the class of deterministic polynomial-time problems. The reason is that a deterministic, polynomial-time solution to any NP-hard problem would also be a solution to every other problem in NP. A powerful computer cannot handle a large instance of the problem. Hence, it is a big challenge and time consuming for engineer to come out with an optimize factory layout.

The effectiveness and efficiency of factory performance is largely influenced by the layout of its' manufacturing facilities. Factory layout is the arrangement of activities (operation, process and etc.), features and spaces in

consideration of the relationship that exists between them (H'astad, 2003). Issues such as costs, work in process inventory, lead-times, productivity, resource utilization (space, operates and etc.) and delivery performance are significantly caused by layout of facilities. 30-75% of total manufacturing costs are partly attributed by materials handling and layout (Mecklenburgh, 1985).

Factor layout planning constraints, includes global issues such as plant location, building design, material handling, etc. In general, factory layout analysis includes a study of the production line process flow charts, material flow diagrams, product routings, processing times, development of from-to charts(table containing. flow values from one department to another in a form), relationship diagrams between different departments in the facility and the cost of material movement (H'astad, 2003).

The major objective when designing the factor layout is to design a physical arrangement that most economically achieves the required output quantity and quality. Achieving the required output (Chen, 2006), involves the improvement of:

- process and work flow
- proper allocation of space and resources
- easier access to supplies and materials
- plant efficiency increase
- maximize the use of space
- safety improvement
- cost savings

Others research such as Mecklenburgh (Chen, 2006) and Francis et al. (Chang, 2000), minimizing the material handling cost is the most considered. Reduced material movement (Holland, 1992) lowers work-in-process levels and throughput times, less product damage, simplified material control and scheduling, and less overall congestion. Hence, when minimizing material handling cost, other objectives are achieved simultaneously (Yong, 1992).

2.2 Metaheuristics Approach to FLP

Metaheuristics are generally applied to problems for which there is no satisfactory problem-specific algorithm, heuristic or when it is not practical to implement both methods. Most commonly used Metaheuristics are targeted to combinatorial optimization problems such as FLP. Various Metaheuristics such as Simulated Annealing (SA) and Genetic Algorithm (GA) are currently used to approximate the solution of the FLP large search space (Singh, 2006).

2.2.1 FLP Approach Based on Simulated Annealing (SA)

SA is a stochastic strategy for searching the ground state. The SA algorithm derives its name from the fact that its behavior is controlled principally by the temperature T as in the thermal annealing process. It is an optimization scheme with non-zero probability for accepting inferior (uphill) solutions. The probability depends on the difference of the solution quality and the temperature. The probability is typically defined by $\min\left\{1, e^{-\frac{\Delta C}{T}}\right\}$ where ΔC is the difference of the cost of the neighboring state and that of the current

state, and T is the current temperature as stated in (Chen, 2006).

However, the excessive running time is a significant drawback of the classical SA process. To reduce the running time of SA for searching for desired solutions more efficiently, several annealing schemes of controlling the temperature changes during the annealing process have been proposed. For instance, Fast Simulated Annealing (FSA) is a semi-local search and consists of occasional long jumps (Chen, 2006). The cooling schedule of the FSA algorithm is inversely linear in time which is fast compared with the classical simulated annealing (CSA) which is strictly a local search and requires the cooling schedule to be inversely proportional to the logarithmic function of time.

A solution was proposed by (Chen, 2006) using a Fast Simulated Annealing (Fast-SA) process to integrate the random search with hill climbing more efficiently. The annealing process consists of three stages:

- 1) High-temperature random search stage
- 2) Pseudo greedy local search stage
- 3) Hill-climbing search stage

In the first stage, the T is set to infinity, so that the probability of accepting an inferior solution approaches 1. The process is like a random search to find the best solution. In the second stage, T is set to 0. Since the temperature is very low, it can only accept a very small number of inferior solutions, which is like a greedy local search. This process is called the pseudo-greedy local search stage. The third stage is the hill climbing search stage. The temperature is raised again to facilitate the hill climbing search stage. Thus, it can escape from the local minimum and search for better solutions. The temperature

reduces gradually, and very likely it finally converges to a globally optimal solution as mentioned in (Chen, 2006).

Since the new simulated annealing scheme save a lot of iterations to explore the solution space, it could devote more time to finding better solutions in the hill climbing stage. This makes the annealing much more efficient and effective. From the experimental results, the new Fast-SA scheme and the B*-tree representation have shown that it leads to faster and more stable convergence to desired floor plan solutions. As claimed in the research (Chen, 2006), Fast-SA is the best choice for the floor planning problem addressed here (it achieved 13.9X speedup over classical SA for finding a floor plan solution of less than 5% dead space for this case).

2.2.2 FLP Approach Based on Genetic Algorithm (GA)

In optimization problem, GA was normally used due to their well-known strength of their robustness. GA robustness is mainly caused by the fact that they deal with sample of candidate solutions to an optimization problem at a time (Koza, 1992). To search for the global optimum, it first process starting from a small set of feasible solutions (population) and generating the new solutions in some random fashion. Performance of GA is problem dependent because the parameter setting and representation scheme depends on the nature of the problem (Aleisa, 2005).

In (Lu, 2008), GA was implemented to optimize facility layout designs in the capital goods industry. The research study on the optimization (GA) to achieve minimized material movement for given schedule of work. Geometric

information on resources and building constraints was included in the model as well. In manufacturing layout, the design problems may be classified as either green field or brown field. These two classes were common constraints found in manufacturing layout planning. A green field problem involves the design of a new manufacturing facility. The designer was free to select processes, machines, transport, layout, building and infrastructure. Brown field problems relate to the redesign of a facility with existing buildings, machine tools and material handling equipment. Brown field problems were often highly constrained, whereas green field problems offer more design choice.

At first stage, the Genetic Algorithm process involves encoding information on resources into chromosomes. Each chromosome is represented as an alphanumeric string that has three parts, the machine number, its rectangular size and location. Individual chromosomes are then randomly selected to produce a population of chromosomes (candidate solutions). Chromosomes are then randomly selected for crossover and mutation operations with the probabilities specified. Crossover combines the characteristics of two parents to produce an offspring, whereas mutation produces random changes in a single chromosome. A repair function then identifies and rectifies infeasible machine sequences. It starts by identifying if any genes produced by crossover are duplicated. Any duplicated genes are swapped between the offspring to ensure that each chromosome contains a gene associated with each resource. The fitness testing algorithm first translates the sequence of machines within the chromosome into a layout by using a placement algorithm. This GA is a construction algorithm as the

placement algorithm generates an entire layout from scratch.

This implementation was tested in two case studies mentioned in (Lu, 2008) research paper. The green field problem mentioned in (Lu, 2008) showed that the solution converged and reduced 70% of the total rectilinear distance travelled. The results obtained when additional constraints were introduced to reflect a brown field design problem also converged. In this case there was an improvement of 30% compared to the company's layout before. However, this research only optimized the layout of manufacturing facilities by minimizing material movement for given schedule of work. This research is still far from a real world factory layout problem. A factory layout problem consists of more constraints and thus causes the search to become more complex to be modeled in GA.

2.2.3 FLP Approach Based on Genetic Programming (GP)

The first experiments with GP were reported by Stephen F. Smith in 1980 and Michael L. Cramer in 1985 (Koza, 1992). Later in the 1992, GP is being research further by Koza. The term GP (Koza, 1992) has two meanings. First, it is often used to subsume all evolutionary algorithms that have tree data structures as genotypes. Second, it is defined as the set of all evolutionary algorithms that breed algorithms using functional programming language, and similar constructs.

For many problems, the most natural representation for a solution is a hierarchical computer program rather than a fixed-length character string. The size and the shape of the hierarchical computer program that will solve a given

problem are generally not known in advance, so the program should have the potential of changing its size and shape. It is difficult, unnatural, and constraining to represent hierarchical computer programs of dynamically varying sizes and shapes with fixed-length character strings.

Representation schemes based on fixed-length character strings do not readily provide the hierarchical structure central to the organization of computer programs (into programs and subroutines) and the organization of behaviour (into tasks and subtasks). Representation schemes based on fixed-length character strings do not provide any convenient way of representing arbitrary computational procedures or of incorporating iteration or recursion when these capabilities are desirable or necessary to solve a problem. Moreover, such representation schemes do not have dynamic variability. The initial selection of string length limits in advance the number of internal states of the system and limits what the system can learn.

GP paradigm continues the trend of dealing with the problem of representation in GA by increasing the complexity of the structures undergoing adaptation (Riccardo, 2008). In particular, the structures undergoing adaptation in GP are general, hierarchical computer programs of dynamically varying size and shape. Varies seemingly different problems in artificial intelligence, symbolic processing, and machine learning can be viewed as requiring discovery of a computer program that produces some desired output for particular inputs. The search space for GP is the space of all possible expressions created by compositions of the available functions and available terminals for the problem. These are called programs and usually expressed as syntax trees rather than as lines of code. The variables and

constants in the program are leaves of the tree. In GP they are called terminals, whilst the arithmetic operations are internal nodes called functions. The sets of allowed functions and terminals together form the primitive set of a GP system.

The manipulation of GP and GA structure for operations such as crossover and mutation is different. GP uses hierarchical structures while GA uses one-dimensional fixed-length linear strings, chromosome. In GP, terminal set and function set should be selected instead so that segmented sub tree is a set of GP system itself.

Expression trees are built from a set of functions F and a set of terminals T . Functions in F are an expression node that consists of binary or ternary arities. For example in Figure 2.1, binary function such as “+” and “-” operators and ternary function such as condition expression when x is equal to 1, 2 or other values.

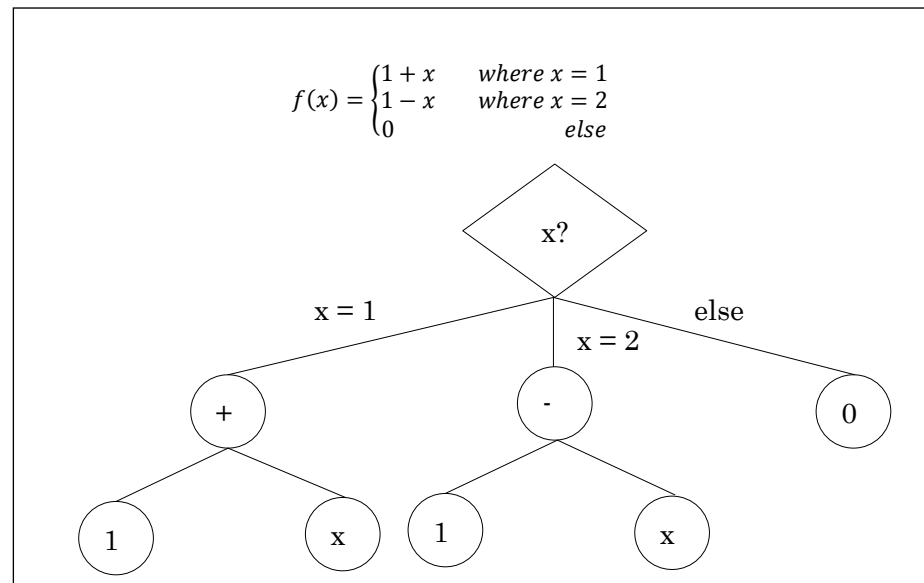


Figure 2.1 Expression tree

There's a great variety of possible program trees, in fact there is an infinite variety. The number of possible recursive compositions of functions

and terminals is infinite if we do not limit the tree's depth. In contrast the standard fixed chromosome-length encoding of most GA give a finite number of possible chromosomes.

The evolutionary process starts with an initial population of randomly generated computer expression trees composed. Genetic programming iteratively transforms a population of expression trees into a new generation by applying analogy of naturally occurring genetic operations. These operations are applied to individual(s) selected from the population. The individuals are probabilistically selected to participate in the genetic operations based on their fitness. The iterative transformation of the population is executed inside the main generational loop of the run of genetic programming. The execution steps of genetic programming are summarized as follows:

1. Randomly create an initial population (generation 0) of individual trees composed of the available functions and terminals.
2. Iteratively perform the following sub-steps on the population until the termination criterion is satisfied:
 - a. Execute each program in the population and ascertain its fitness (explicitly or implicitly) using the problem's fitness measure.
 - b. Select two individual trees from the population with a probability based on fitness (with reselection allowed) to participate in the genetic operations in (c).
 - c. Create new individual program(s) for the population by applying the following genetic operations with specified probabilities:

- i. Reproduction: Copy the selected individual program to the new population.
 - ii. Crossover: Create new offspring program(s) for the new population by recombining randomly chosen parts from two selected programs.
 - iii. Mutation: Create one new offspring program for the new population by randomly mutating a randomly chosen part of one selected program.
 - iv. Architecture-altering operations: Choose an architecture-altering operation from the available repertoire of such operations and create one new offspring program for the new population by applying the chosen architecture-altering operation to one selected program.
3. After the termination criterion is satisfied, the single best program in the population produced during the run (the best-so-far individual) is harvested and designated as the result of the run. If the run is successful, the result may be a solution (or approximate solution) to the problem.

The main evolutionary operations of GP run exactly like GA. However the representation of expression trees from GP and chromosome from GA that set them apart.

CHAPTER 3.0

APPROACH AND ALGORITHMS

3.1 Model Overview

In this chapter, we discuss on the design used by our model. Our model adopted three stochastic search algorithms for optimizing factory layout using FSA, GA and GP. Before we jump into the search algorithm design, we will show how a top down visual factory layout design is being translated to a B*-tree, chromosome and expression tree representation for FSA, GA and GP respectively.

3.2 Layout Representation Using B*-Tree

The tool adopts B*-Tree (Chang, 2000) representation for modeling cells layout planning as it is used in Chen and Chang research (Chen, 2006). Given a two dimensional factory layout (top view), a unique B*-tree in linear time to model the placement can be constructed. Given a B*-tree, a legal placement by packing the blocks in amortized linear time with a contour structure can be obtained as well. Figure 3.1 and 3.2 show the top view of a factory layout and its corresponding B*-tree. The B*-tree used was an ordered binary tree whose root corresponds to the block on the upper-left corner. B*-tree is constructed from an admissible placement in a recursive fashion.

Starting from the root, then recursively constructs the left and right branches but never more than two branches. If the node branch out from the left means it is placed on the left of its parent node and right branch is placed on the right side of its parent node.

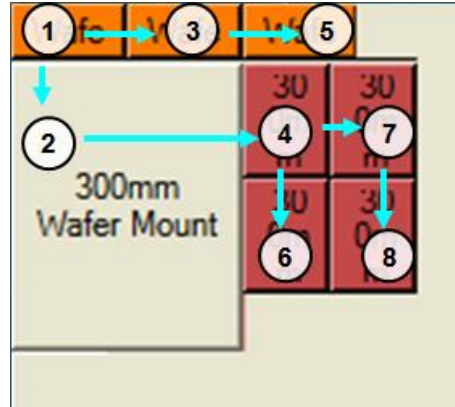


Figure 3.1 Top view of a factory layout

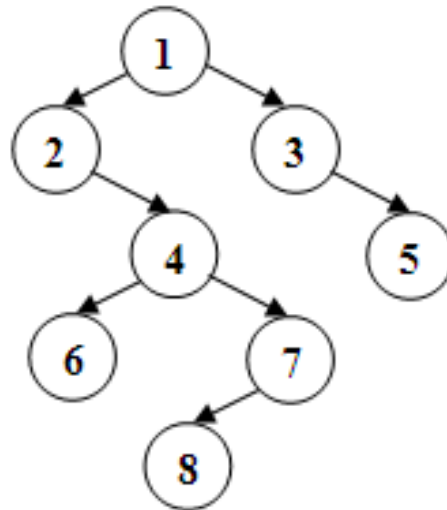


Figure 3.2 B*-tree representing the placement

In a B*-tree, the root is the most top left block and thus the coordinate of the block as in Eq (1).

$$(x_{root}, y_{root}) \in \mathbb{Q} \quad (1)$$

If node n_j is the right child of node n_i , block b_j is placed on the right-

hand side and adjacent to block b_i ; example as in Eq (2) given that x , y , w and l denotes the block properties of its x-axis coordinate, y-axis coordinate, width and length respectively. The coordinate of n_j is calculated by the properties of n_i as in Eq (2).

$$(x_j, y_j) = (x_i + w_i, y_i + l_i) \quad (2)$$

Otherwise, if node n_j is the left child of n_i , block b_j is placed below of block b_i , with the x-coordinate of b_j equal to that of b_i ; i.e., $x_j = x_i$. Therefore, given a B*-tree, the x-coordinates of all blocks can be determined by traversing the tree from top to bottom and vice versa of obtaining y-coordinates.

3.3 Optimization Algorithms

In this research, we use FSA, GA with conditional crossover and GP. All three search algorithms are stochastic strategy for searching an optimal state instead of exhaustively searching all possible combinations. The performance of each implementation will be discussed later in chapter 5 through our experiments observation.

3.4 Fast Simulated Annealing (FSA)

Simulated Annealing (SA) is a stochastic strategy for searching the ground state. The SA algorithm derives its name from the fact that its behavior is controlled principally by the temperature T as in the thermal annealing process. It is an optimization scheme with non-zero probability for accepting

inferior (uphill) solutions. The probability depends on the difference of the solution quality and the temperature, $P(A)$. The probability is typically defined as in Eq. (3).

$$P(A) = (1 < n < e^{-\Delta C/T}) \quad (3)$$

n is a random number, ΔC is the difference of the cost of the neighboring state and that of the current state, and T is the current temperature (Chen, 2006).

However, the excessive running time is a significant drawback of the classical SA process. To reduce the running time of SA for searching for desired solutions more efficiently, several annealing schemes of controlling the temperature changes during the annealing process have been proposed. For instance, FSA is a semi-local search consists of occasional long jumps. The cooling schedule of the FSA algorithm is inversely linear in time which is fast compared with the classical simulated annealing which is strictly a local search and requires the cooling schedule to be inversely proportional to the logarithmic function of time.

The model is using FSA as the optimization engine proposed by (Chen, 2006). The proposed FSA integrates random search with hill climbing more efficiently by manipulating the temperature to three stages as shown in figure 3.3. The three stages of the annealing process are mentioned in chapter 2.2.1.

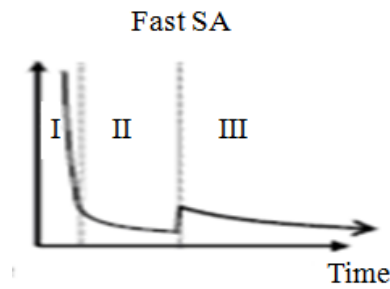


Figure 3.3 Three stages of FSA, temperature VS search time

In the first stage, the T is set to a very huge number, so that the probability of accepting an inferior solution approaches 1. The process is like a random search to find the best solution. In the second stage, T is set to 0. Since the temperature is very low, it can only accept a very small number of inferior solutions, which is like a greedy local search. This process is called the pseudo-greedy local search stage. The third stage is the hill climbing search stage. The temperature is raised again to facilitate the hill climbing search stage. Thus, it can escape from the local minimum and search for better solutions. The temperature reduces gradually, and very likely it finally converges to a global optimal solution.

The prototype tool, adopts the B*-tree representation to model a factory layout. Each B*-tree corresponds to a factory layout. Hence, the search space consists of all B*-trees with the given nodes (blocks). The FSA algorithm begins with a random generated initial solution. To find a neighboring solution, we manipulate a B*-tree to get another B*-tree by the following operations:

- Op1: Rotate a block

We randomly select a node in the B*-tree and rotate the block, which does not affect the B*-tree structure. This is done through swapping the values between the width and length of that specific block.

- Op2: Move a node/block to another place

We randomly delete a node and move it to another place in the B*-tree. The random selection will only select the leaves in the tree. The selected node is move to another empty position in the tree.

- Op3: Swap two nodes/blocks

We swap two nodes in the B*-tree. All the nodes in the tree will be randomly select for swapping with another random node without changing the width and length.

After generating a new neighboring state, the state will go through an evaluation function to obtain the fitness value. Based on the temperature, a worse neighboring state might be chosen instead.

3.5 Genetic Algorithm (GA)

Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics (Holland, 1992). GA behaves to imitate the development of new and better populations among different species during evolution (Tam, 1992). Unlike most of the heuristic search algorithms, GA conducts the search through the information of a population consisting of a subset of individual solutions. Each solution is evaluated by a fitness function to associate with a fitness value, which is the objective function value of the solution. In this case, we tried to minimize our fitness value which result in shorter travelling distance, less space wastage and less overlapping space. Solutions to optimization problems are coded to a finite B*-tree representation based on the machines quantities demand. The genetic algorithms work on these trees and encoding is done through the structure named chromosomes, where each chromosome is made up of units called genes. In the beginning of

the algorithm, each chromosome is generated randomly as the initial population. There are some determining factors that strongly affect the efficiency of genetic algorithms:

- Evaluation functions algorithms.
- The representation of the solutions.
- The generation of the initial population.
- The selection of individuals in an old population (parents) that will be allowed to affect the individuals of a new population.
- The genetic operators that are used to recombine the genetic heritage from the parents to produce children (crossover and mutation).

The selection of individuals that will be allowed to affect the following generation is based on the fitness of the individuals. This is done in such a way that individuals with better fitness are more likely to be chosen to become parents. The three population controls are as below:

1) Population control through candidate selection:

It has proven very efficient to search for locally optimal solutions in the neighborhood of the children (Holland, 1992). If one is able to find a better solution then it will replace the original child as a member of the new population.

2) Population control of new individuals:

It is possible that a child has a worse fitness than its parents. In such cases the child will not be accepted in the new generation.

3) Population Elimination:

For an ideal case, a new population should converge to a more optimal

solution from its previous population but this is not guarantee. In the worst scenario, the new population may result in having the same solution as its previous population. This shows that the search is trap in a local optimal and a new population initialization will be generate again.

Note that the GA implementation requires the specification of certain parameters such as population size, and number of generations and n control the size of the population. Then the genetic algorithm procedure can be described as in Figure 3.4.

```

Input: A problem instance
Output: A (sub-optimal) solution

BestSolution;
n = population size;
P = initialize P, and evaluate the fitness of the individuals in P
P2 = empty array for coming new population
m = mutation rate
cr = crossover rate

do
  BestSolution = rank(P)

  do
    if probability(cr)
      if fitness(crossover( $c_{i1}$ ,  $c_{i2}$ , P)) better than ( $c_{i1}$ ,  $c_{i2}$ )
        P2 add crossover( $c_{i1}$ ,  $c_{i2}$ , P)
      if mutation(m, P)
        mutate( $c_i$ )
    until size(P2) equal to n

  P=P2
  NewOptimalSolution = rank(P)
  if NewOptimalSolution equal to BestSolution
    Terminate P
    P = initialize P, and evaluate the fitness of the individuals in P
  else
    BestSolution= NewOptimalSolution (sub-optimal solution from
the population)
  until (termination condition is not satisfied)

```

Figure 3.4 Genetic Algorithm Structure

3.5.1 GA Crossover

By combining the encoded solution strings of two parents, two children are created. If one considers the biological origin of the genetic algorithms it makes sense to denote the coded solution string “genome” and look at this procedure as a result of mating.

1-point crossover is applied to perform a crossover operation as described in Figure 3.5. Firstly, a crossing point within the chromosome’s genes is chosen randomly. Then the new chromosomes get the header part and the tree structure from the first parent. The remaining genes will be obtained from the second parent in the order as they appear on the second parent.

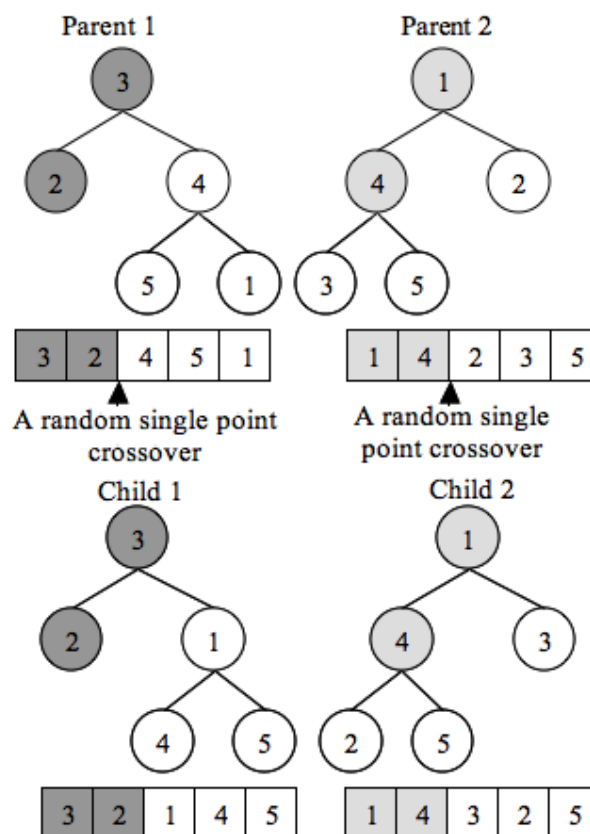


Figure 3.5 GA: Swapping between parent 1 and parent 2

To avoid chaotic behavior, in this work we introduce conditional

probability crossover method. Offspring generated from crossover can prompt poor fitness value due to infeasible machines layout as exhibited during experiments and results. Given a scenario where the offspring are worse than their parents as in Figure 3.6 crossover does not help to converge to the global solution. Hence, not all individuals in the new population are generated by this operator. The probability of applying this operator is controlled by a crossover rate which is a constant percentage respective to the next generation.

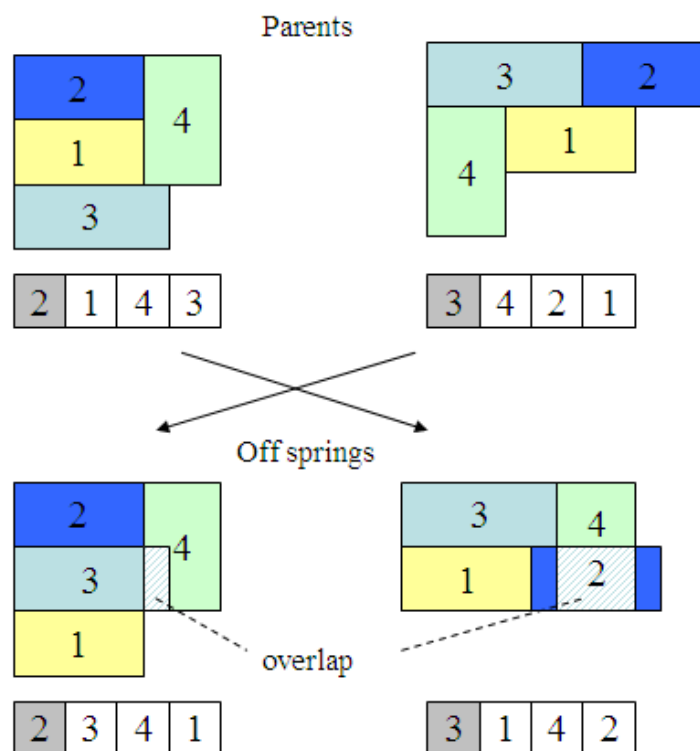


Figure 3.6 Bad offspring generated from crossover

3.5.2 GA Mutation

In order to give the populations new impulses some random changes in the genomes are allowed to occur. The mutation operator changes a “gene” in a solution with a probability of the mutation rate. By doing so, this reduce the

chances of the algorithm to behave as a greedy search and trap in a local optimal. Mutation are done by three different operations, these are:

Op1: Rotate a block.

We randomly select a node in the B*-tree and rotate the block, which does not affect the B*-tree structure. This is done through swapping the value between the width and length of that specific block.

Op2: Move a node/block to another place.

We will randomly delete a node and move it to another place in the B*-tree. The random selection will only select the leaves in the tree. The selected node is move to another empty position in the tree.

Op3: Swap two nodes/blocks.

We swap two nodes in the B*-tree. All the nodes in the tree will be randomly select for swapping with another random node without changing the width and length.

3.6 Genetic Programming (GP)

Genetic Programming (GP) is an automated methodology inspired by biological evolution to find computer program that best performs a user-defined task. It is therefore a particular machine learning technique that uses an evolutionary algorithm to optimize a population of computer programs according to a fitness landscape determined by a program's ability to perform a

given computational task.

In the early (and traditional) implementations of GP, program instructions and data values were organized in tree structure or known as expression tree (Koza, 1992). This method is adopted in Perez's work in "Solving Facility Layout Problems Using Genetic Programming" (Jaime, 1996). In this research on GP, the design of modeling our factory problem statement into GP will be using Perez's work as reference.

Just as GA, GP search through the information of a population consisting of a subset of individual solutions. However, these individual are represented in expression tree, not chromosome. Expression tree is evaluated by a fitness function to associate it with a fitness value, which quantify the quality of a solution. Since the B*-tree design is an expression tree and it fits perfectly fine for GP design without translation method as GA did earlier. However, this does not change the efficiency factors of the search performance. These determining factors of GP performance are as mentioned in chapter 3.3 earlier.

In this research, the representation is the same as the expression tree/program being used by J Garces-Perez in facilities layout optimization using GP (Jaime, 1996). Both expression trees are programs that consist of operators, top, down, left and right that describe the position of the leaves which is the machines in our layout. Figure 3.7 is the representation used in our research with the top down machine layout being translated from a tree expression.

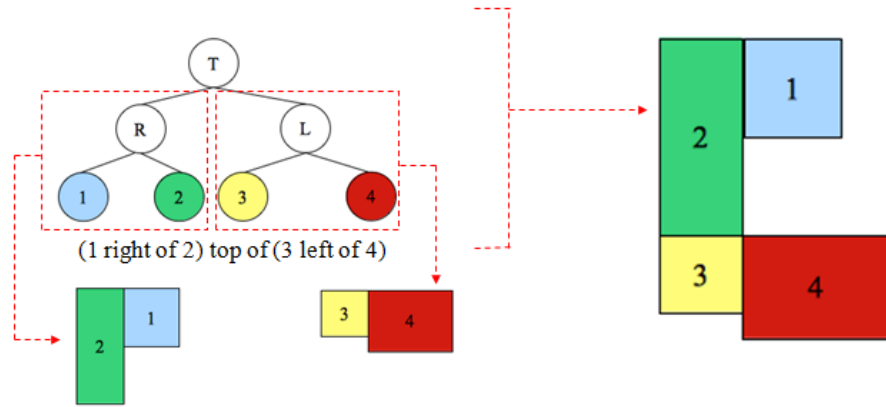


Figure 3.7 GP expression tree

In the beginning of the algorithm, each tree is generated randomly as the initial population. The following generation is based on the fitness of the individuals (tournament). This is done in such a way that individuals with better fitness are more likely to be chosen to become parents. The combination of the population consists of the following four operations:

3.6.1 GP Crossover

Crossover is applied on an individual by simply switching one of its nodes with another node from another individual in the population. With a tree-based representation, replacing a node means replacing the whole branch. This adds greater effectiveness to the crossover operator. The expressions resulting from crossover are very much different from their initial parents.

1-point crossover is applied to perform a crossover operation as described in Figure 3.8. One of the nodes in both parents tree structure is chosen random for crossover. The new child gets the tree structure from the first parent but the crossover point node is replaced with the sub branches of the second parents.

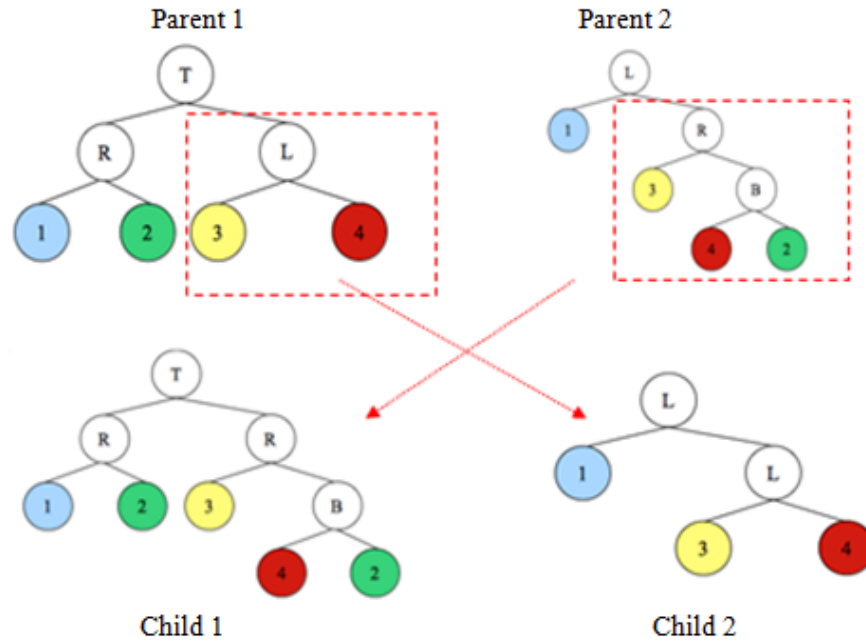


Figure 3.8 GP: Swapping between parent 1 and parent 2

3.6.2 GP Mutation

As for mutation and population control design, there are the same as GA that we mentioned earlier in chapter 3.4. GP implementation requires the specification of certain parameters such as population size, and number of generations. The genetic programming algorithm procedure is same as Figure 3.4. The difference lies in the crossover method due to GP representation of the optimal solution.

3.7 Optimization Goal

In this research, optimal result is based on four main optimization goals. Space utilization (SU) and material flow (MF) goals can be adjusted in

the fitness function. As for Overlapping Placement (OP) goal is factor in the formula of SU and manning ratio is preprocessing of the data. In later section, we will discuss in detail on how to obtain SU, MF, OP and manning ratio. Eq. (6) is the formulated optimization fitness function adopt in our model. The goal is to minimize our fitness function. The priority of these two goals is controlled by two constant as scaling factors, b and c .

$$\text{minimize Fitness value} = (\text{SU})b + (\text{MF})c \quad (4)$$

where b and c are constant.

3.7.1 Calculate Space Utilization and Overlapping Placement

From a B*-tree, the factory layout can be mapped to a matrices representation. For each coordinate in the layout, it will be mapped to matrices which indicate whether that particular space is being occupied by a block or more than one blocks or it is empty. The mapping is done from coordinate (0, 0) till the largest x-axis and y-axis coordinates which are occupied, $\max(\text{x-axis})$ and $\max(\text{y-axis})$. The rows and columns of the matrices is equivalent to the x and y coordinates of the factory layout. The translation from a factory layout design to a matrices representation is shown in Figure 3.9 and 3.10.

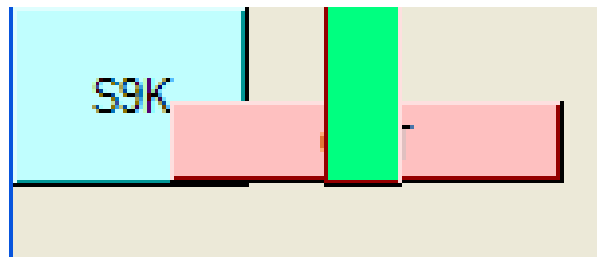


Figure 3.9 Factory layout design

```

11111111 1 1 00000110000000
11111111 1 1 00000110000000
11111112 2 2 11111221111111
11111112 2 2 11111221111111

```

Figure 3.10 Matrices representation of the factory layout

The areas of overlapping placement (OP), occupied space (OS), waste space (WS) and space utilization (SU) are defined as in Eq. (7).

$$\begin{aligned}
 (OP) &= (\text{sum of the matrices values} > 1) \\
 (OS) &= (\text{sum of the matrices values} = 1) \\
 (WS) &= (\text{sum of the matrices values} = 0) \\
 (SU) &= (WS) + (OS) + (OP * OS)
 \end{aligned} \tag{5}$$

3.7.2 Calculating Material Flow

Material flow is the sum of all distance between related operation areas. The start and end points of the distance path are from the centroids of both operation areas. In this case, the distance is the Euclidean distance between two centroids as shown in Figure 3.11.

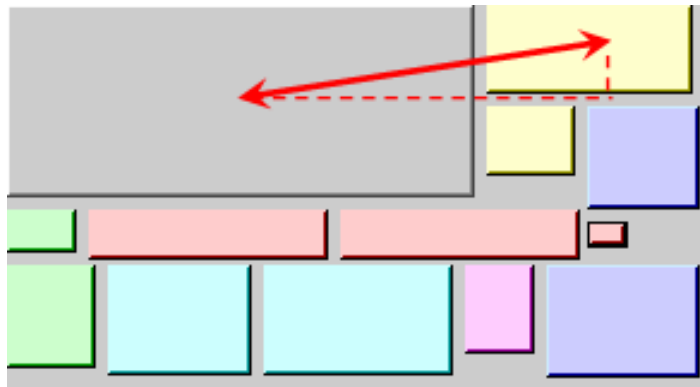


Figure 3.11 Material flow (based on Euclidean distance)

The flow path of the whole factory has to be predefined earlier. Each operation can have more than one flow path flowing in or out. A scaling factor

can be set to define the material flow priority. High runner products induce higher material flow. A scaling factor (constant multiplier) can be set to each particular material flow path to model the priority of the material flow influence. The scaling factor will be used to multiply with the Euclidean distance of that particular flow which boosts up the effect to the total material flow. For example, given testers A, B and C with material flow from AB is higher priority (larger constant) and BC (smaller constant) is lower priority, the total material flow (MF) distances incurred is calculated with the Eq (6) below,

$$MF = (Euclidean(A,B) \times g) + (Euclidean(B,C) \times h) \quad (6)$$

where both g and h are constants. As we are trying to minimize MF, larger scaling factors will amplify the respective calculated Euclidean distance more and has higher influence of the total MF.

3.8 Obtaining Manning Ratio Arrangement

The layout arrangement of the machines dictates the manning ratio that can be supported; e.g. one operator control 2, 3 or 4 machines. Arrangement that allows flexible manning ratio is by placing machines with their controller facing each other (cross box) as shown in Figure 3.12. A cross box arrangement allows equivalent distances for an operator travel from one machine controller to another. This arrangement exhibits feasible manning ratio flexibility for one operator to control 2 or 4 machines.

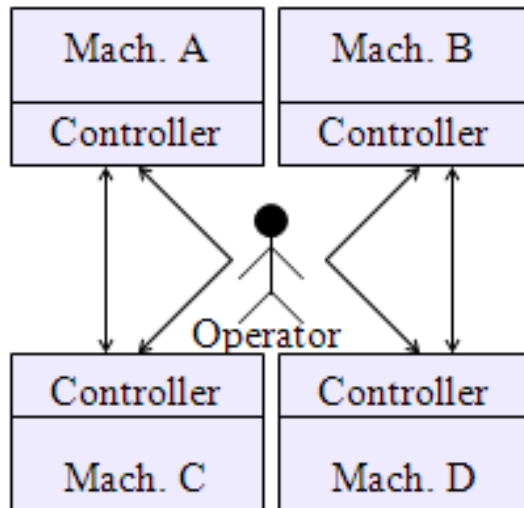


Figure 3.12 Four machines and an operator cross box arrangement.

The number of machines dictates the numbers of cross box arrangement that is needed. Cross box can be expended to allow more manning ratio feasibility as shown in Figure 3.13.

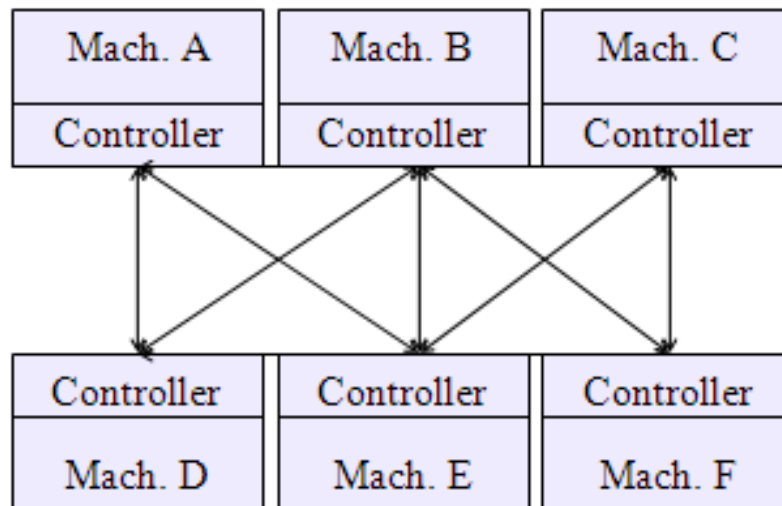


Figure 3.13 Arrangement that support flexibility manning ratio of 1 operator to control 6 machines

Manning ratio is a prefix requirement imposed the input data before sending to the model. Since this model is developed for Intel Manufacturing usage, every machine type is configured to a defined manning ratio number

and need to be followed exactly. For instance, given machine with quantity of 8 and manning ratio is define as 1 to 4 (an operator is able to control up to 4 machines), this machine is configured into the model as two square dimension of 2 X 2 of the specified machine type.

CHAPTER 4.0

DATA PREPARATION

4.1 Case Studies Overview

Two case studies are used to validate our models. In case study 1, mock-up data is created to observe the model result based on different scaling factors setting of the fitness function. In case study 2, real industrial data from Intel Manufacturing is used. The results shown in case study 2 are verified by layout industrial engineers from Intel Manufacturing.

4.2 Case Study 1: Mock-up Data for Model Validation

In Case study 1, five experiments are conducted to test the effect of different scaling factors of the fitness function as mentioned in chapter 3. The scaling factors of b and c as mentioned in chapter 3.7, Eq 6, was configured according to Table 4.1. This observation helped us to understand the effect balancing the optimization goal priority with the right setting.

Scaling Factor	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5
b	1	0	1	1	1
c	0	1	179	89.5	268.5

Table 4.1 Scaling Factor Configuration Setting

Both experiment 1 and 2 are extreme cases where we only considered space utilization and material flow respectively. With the fitness value derived

from experiment 1 and 2, we calculated the degree of differences between them and use the value in experiment 3. In chapter 5, fitness value for experiment 1 is 16×10^4 and fitness value for experiment 2 is 896. This means that space utilization value is 179 times larger than material flow value in rough estimation. Hence in experiment 3, c is set to 179. In experiment 4 and 5, we create a lower and upper bound using half of 179 as reference which is 89.5 and 268.5 respectively.

Sample data used in case studies 1 are rectangles that can be combines back to a square. A square is sliced into 7 smaller rectangles as shown in Figure 4.1. This helped us to validate the model and suggest the most optimal layout based on space utilization without considering the process flow of connected rectangles. The process flow of each connected rectangles are listed in Table 4.2 with equal priority with value 2 and 0 means that it is an independent component without any process flow to another component.

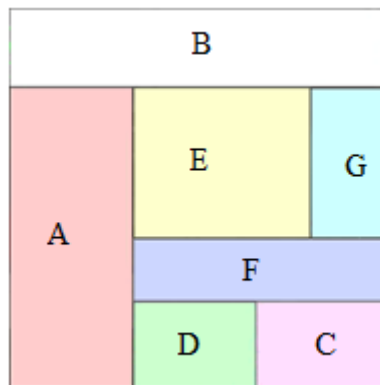


Figure 4.1 Sample data of a perfect square

Rectangle	Width	Length	Connected Rectangles	Priority Value
A	130	318	B	2
B	82	400	A	2
C	93	140	D	2

D	93	130	C	2
E	160	185	F	2
F	65	270	E	2
G	85	160		0 (independent rectangle)

Table 4.2 Case study 1 configuration setting

4.3 Case Study 2: Real Industrial Data

In case study 2, we used real industrial data based on Intel Manufacturing NCO6, Penang NetComm and Chipset factory layout. These data are based on real layout challenges faced by engineers and managements. Often, what is the best decision to decide the best factory layout retrofit is common for an existing factory. Through manual layout design there is no scientific way to measurement a layout quality. Hence, in this case study, three experiments, i.e. experiment 6, 7 and 8, took place in giving a suggested layout with a fitness value as quality indicator.

In experiment 6, data is obtained from capacity engineers to provide maximum machines quantities needed to fit into a factory under extreme production condition. In this experiment, we shall observe how a suggested layout is able to fit all machines into a limit empty space without jeopardizing the process flow. Such layout will serve as a guideline for future layout planning. The input setting for the experiment run is as in Table 4.3

Tester	Width	Length	Controller location	Quantity	Priority Value to FOL
A	9	11	Length	46	6
B	19	22	Width	16	5

C	22	24	Length	5	4
D	15	18	Width	4	3
E	19	26	Width	14	2
F	14	20	Width	10	1

Table 4.3 Input Setting For Experiment 6

In experiment 7, data is obtained from layout engineers where machines retrofit took place. A total of 11 new machines for tester A and tester B respectively, need to be arranged in an existing layout. There was existing tester B which already in place. Hence, it is best if new tester B is placed next to the existing tester B. It is a simple machine arrangement allocation. However, what visually look good in a layout design doesn't mean there's no better alternative. This experiment helps to validate the quality of the manual design and provide another alternative suggestion. The manual designs suggested by engineers are as in Figure 4.2. The input setting for the experiment run is as in Table 4.4.

This is a simple experiment but it was a real scenario happened in NCO6, Penang factory for Catalyst and CMT machines retrofit effort. Since machines retrofit is a high costing task, management requires scientific data to prove the best course of action. In the end, management accepted the solution from the model instead of the manual layout.

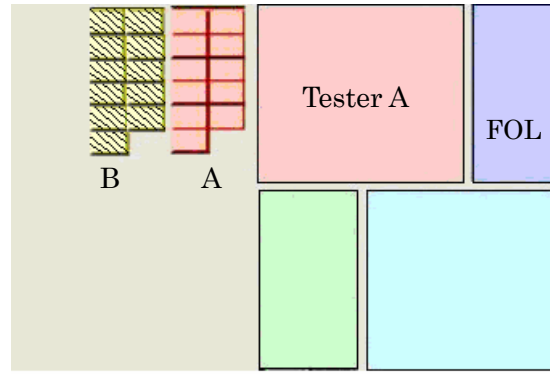




Figure 4.2 Manual suggested layout by engineers

Tester	Width	Length	Controller Location	Quantity	Material Flow	Priority Value
A	15	18	Width	11	Tester A	3
B	15	18	Width	11	FOL	2

Table 4.4 Input setting for experiment 7

In experiment 8, data is obtained from layout engineers where the machines quantities are exactly the same as the existing factory. The purpose of the experiment is to evaluate the existing layout quality and provide a better layout if available. Through the fitness value, we shall calculate the improvement of a suggested layout compare to the existing layout. The input setting for the experiment run is as in Table 4.5. The graphical representation is print screen image of each particular machine from AutoCAD instead of using plain colored buttons. Note that the different visual representation in this section does not dictate the model differently than the previous experiment.

Tester	Graphical Representation	Width	Length	Controller Location	Quantity	Priority Value to FOL & EOL
A		9	11	Length	22	6
B		19	22	Width	11	5





C		22	24	Length	7	4
D		15	18	Width	17	3
E		19	26	Width	6	2
F		14	20	Width	8	1

Table 4.5 Input setting for experiment 8

CHAPTER 5.0

EXPERIMENTAL RESULTS AND ANALYSIS

5.1 Experiments Overview

In this chapter, we will discuss on the analysis results with experiments defined from chapter 4. Experiments defined are split into to two case studies. Case study 1 is experiments ran using mockup data to analyze the scaling factors setting of the fitness function. In case study 2, experiments ran using real industrial problem and data from Intel Manufacturing.

5.2 Case Study 1: Results and Analysis

In case study 1, our models ran with mock-up data specified in chapter 4.1. The results of our model are as in Table 5.1, 5.2, 5.3, 5.4 and 5.5 for experiment 1, 2, 3, 4 and 5 as mentioned in chapter 4. The table shows the fitness value as our indicator of a layout quality.

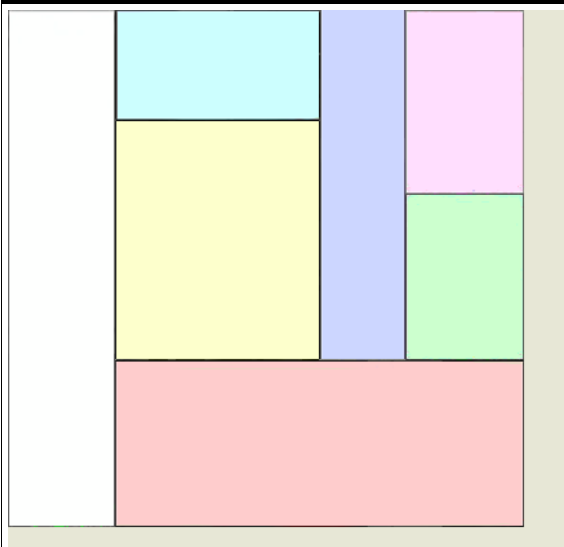
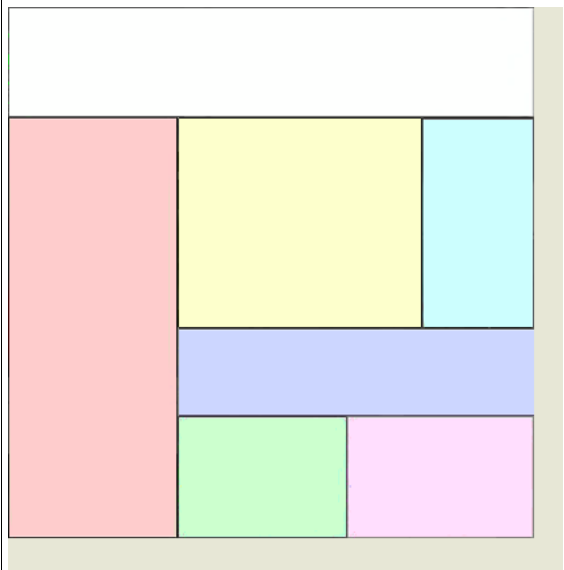
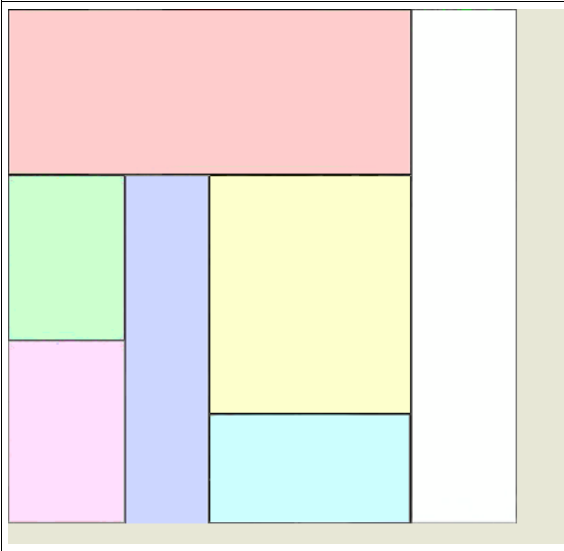
Layout	Algorithm	Fitness Value
	FSA	16×10^4
	GA	16×10^4
	GP	16×10^4

Table 5.1 Experiment 1 results with $b = 1$ and $c = 0$

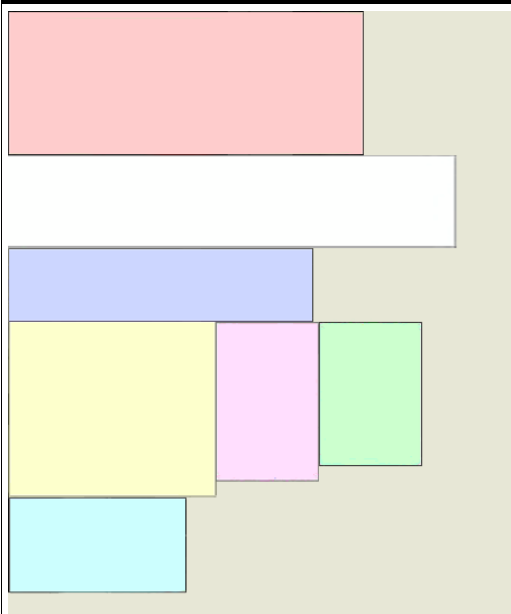
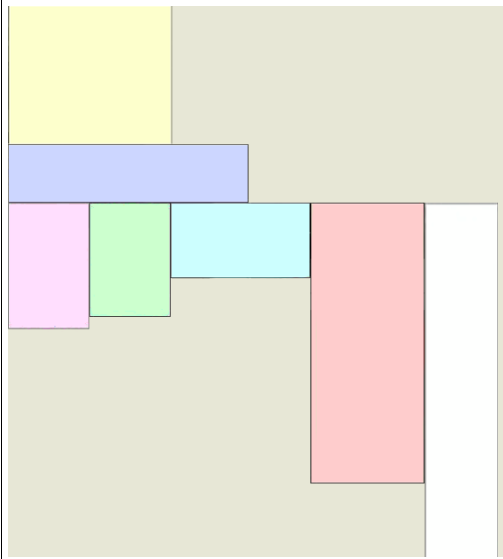
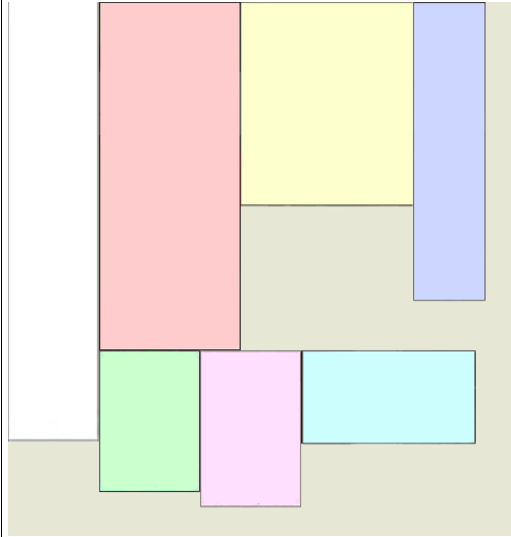
Layout	Algorithm	Fitness Value
	FSA	896
	GA	896
	GP	896

Table 5.2 Experiment 2 results with $b = 0$ and $c = 1$

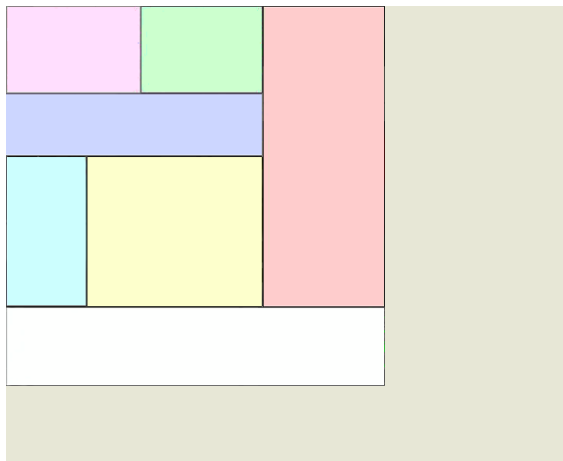
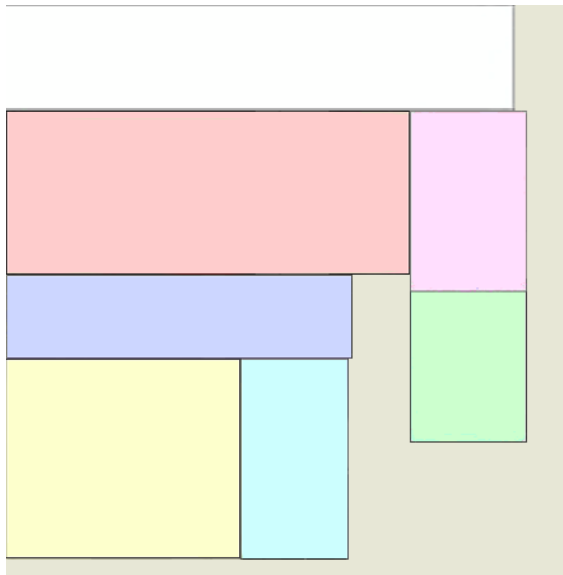
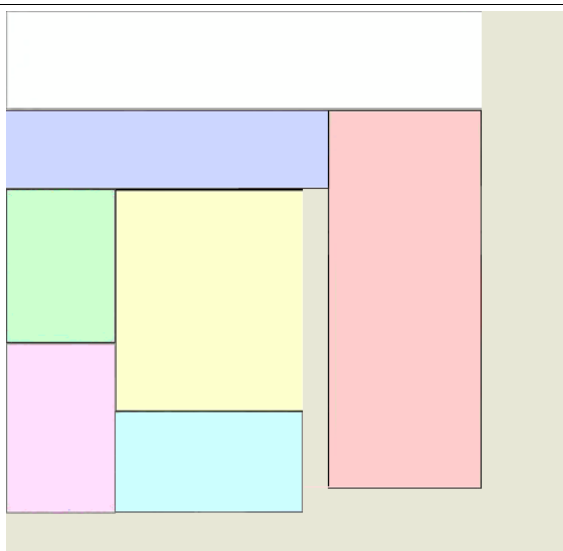
Layout	Algorithm	Fitness Value
	FSA	27.0×10^4
	GA	33.6×10^4
	GP	29.3×10^4

Table 5.3 Experiment 3 results with $b = 1$ and $c = 179$

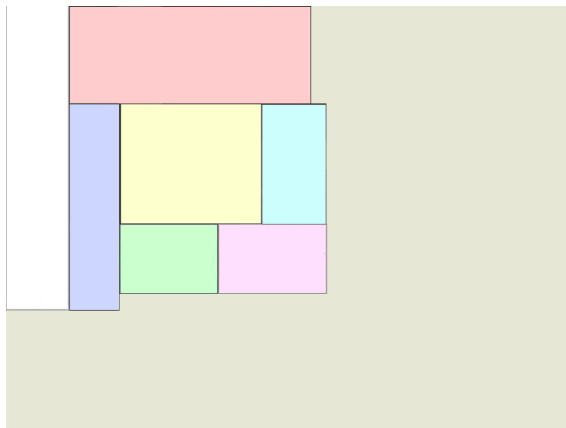
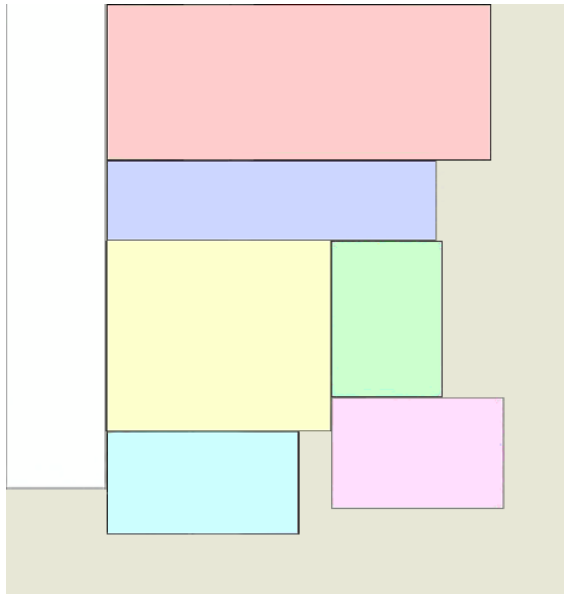
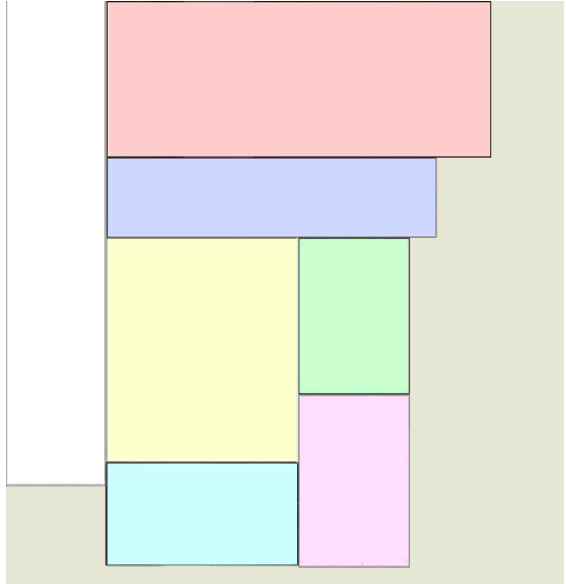
Layout	Algorithm	Fitness Value
	FSA	29.3×10^4
	GA	35.1×10^4
	GP	30.8×10^4

Table 5.4 Experiment 4 results with $b = 1$ and $c = 89.5$

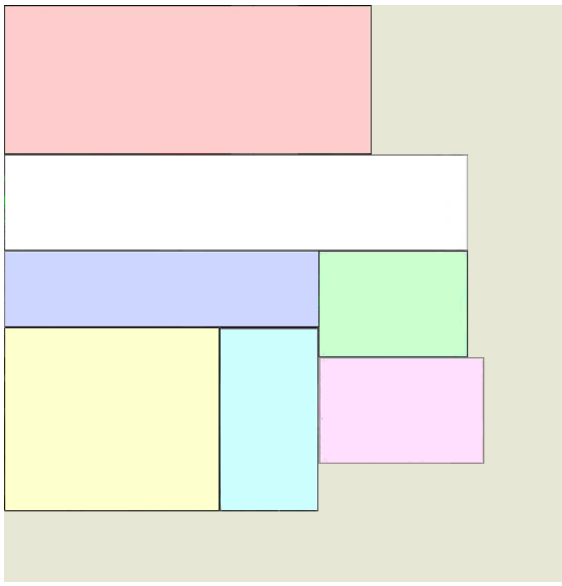
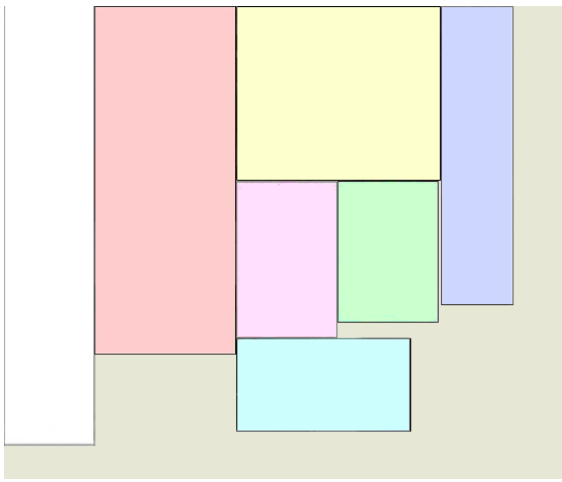
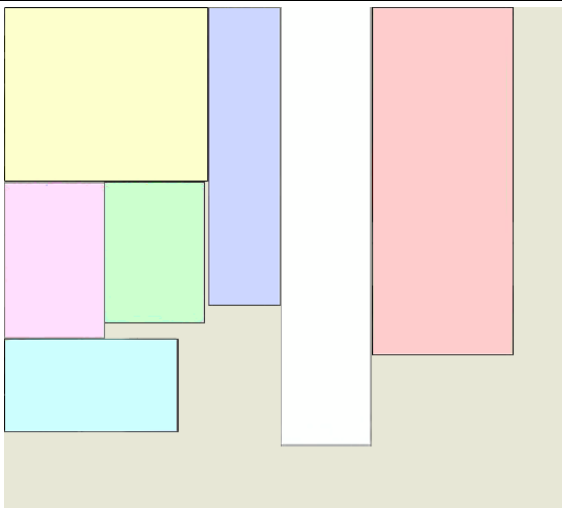
Layout	Algorithm	Fitness Value
	FSA	40.6×10^4
	GA	42.8×10^4
	GP	42.8×10^4

Table 5.5 Experiment 5 results with $b = 1$ and $c = 268.5$

In experiment 1, b is set to 1 and c is set 0 which means only the space utilization goal is considered and not material flow. All algorithms converged to the same result where fitness value equal to 16×10^4 . The result showed the best layout orientation for space utilization due to the reason that no empty space existing in between the rectangles. All rectangles are fit perfectly back to the original square that we sliced out.

In experiment 2, b is set to 0 and c is set to 1 where only material flow goal is considered but not space utilization. All algorithms converged to the same result with fitness value equal to 896. All connected rectangles with material flow set in the configuration and being placed side by side. This experiment is used to estimate the most optimal solution for material flow as reference. Both fitness value from experiment 1 and 2 are used to estimate the scaling factors of b and c in later experiments, 3, 4 and 5. Since space utilization value is estimated to be 179 times larger than material flow, hence in experiment 3, c is set to 179.

Experiment 3 exhibits the best optimal solution for our case study with FSA algorithm when b is set to 1 and c is set to 179. The result from FSA showed a fitness value of 27×10^4 which is best compare to GA and GP with 33.6×10^4 and 29.3×10^4 respectively. Layout from FSA result showed all connected rectangles are lay side by side and it managed to derive the best space utilization solution as in experiment 1.

In experiment 4 and 5, b is set to 1 and c is set to 89.5 as the lower limit and b is set to 1 and c is set to 268.5 as upper limit respectively. Though all optimal results showed the connected rectangles are placed side by side, the fitness value indicates that FSA result in experiment 3 is the better one.

This shows the important of setting the right balance which reduce the bias between two optimization goals. However, in this experiment we are only running with seven rectangles as input data. This allows our model to converge to a consistent result of both extreme case for space utilization and material flow. In real industrial data, obtaining such result with a large numbers of input data required a long processing time.

Process flow is deprioritized when compared to space utilization in Intel Manufacturing practice. Having a healthy material flow is a nice to have practice but it is not as important as space utilization. Since Intel Malaysia is an assembly and test manufacturing, materials travel within the factory does not incurred cost as claimed in this respective industrial environment. However, space utilization can be translated to cost as every empty space in factory is capacity of factory to store materials, machine and etc. As a result, having fair scaling factors setting in the fitness function does not always guarantee the best solution rather it should be adjusted to business need.

5.3 Real Industrial Factory Retrofits

Experiments were done on Intel Manufacturing Factory in NCO6, Penang NetComm and Chipset, where the machine types, machine quantity, machine dimension, machine controller location, material flow, and existing area are given as inputs. NCO6 is high mix low volume factory that consist of more constraints compare to CPU factory due to CPU factory has less process flow routes. These results were analysis through observation in term of space utilization, material flow, manning ratio and overlapping placement avoidance.

In experiment 6, given an existing factory floor, the front of line (FOL) dimension is 194 x 79 at location (0, 0). From FOL, materials flow to tester A, B, C, D, E and F. Tester A is the highest product runner with the most material flow activities and followed by tester B, C, D, E and F according. The details of the testers are shown in Table 4.3.

With the input parameters passed to the prototype tool the result are shown in figure 5.1 with fitness value of 4592.7×10^4 . Every iteration, the search will converge to an optimal solution. The best optimal solution will be stored until a better optimal solution is found in the next iteration.

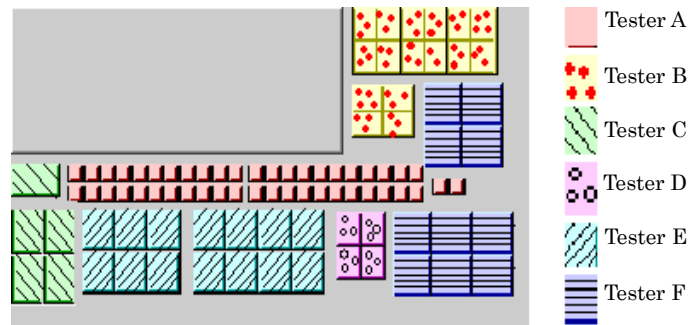


Figure 5.1 Experiment 6 result using FSA

The convergence of the search characteristic is shown in Figure 5.2. In the first stage 1, the temperature is set to high and decrease overtime. During this stage, the search behaves as random search. In the stage 2, the temperature had decreased to very low. The search behaves as a pseudo-greedy local search and converges to a local optimal. Lastly in stage 3 the temperature rises again and force search behaves as a hill climbing search. This will allow the search to escape from local optimal and having higher chances to converge to a better solution.

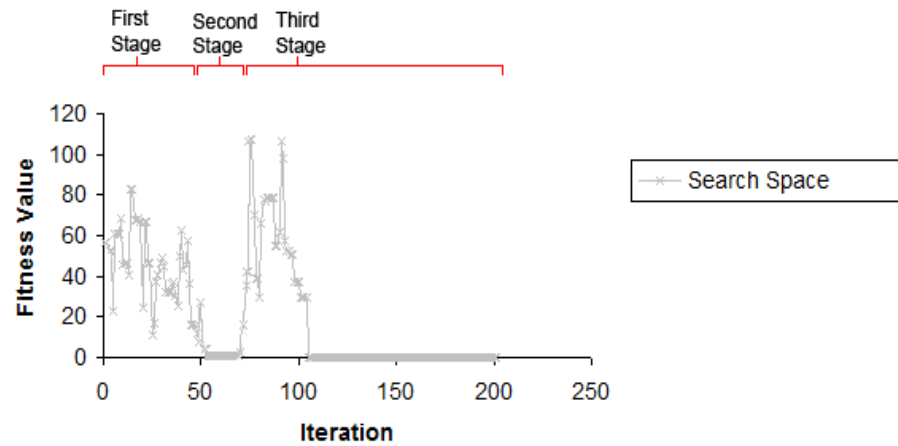


Figure 5.2 Search trend of the three stages FSA; fitness value VS iteration (time)

With the input parameters passed to the prototype tool using GA. The result is shown in Figure 5.3 after running 625 generations with 10% crossover probability rate. The algorithm performance was tested using different crossover rate. The algorithm is stopped when it reaches the approximately similar results in term of fitness level as shown in Table 5.6.

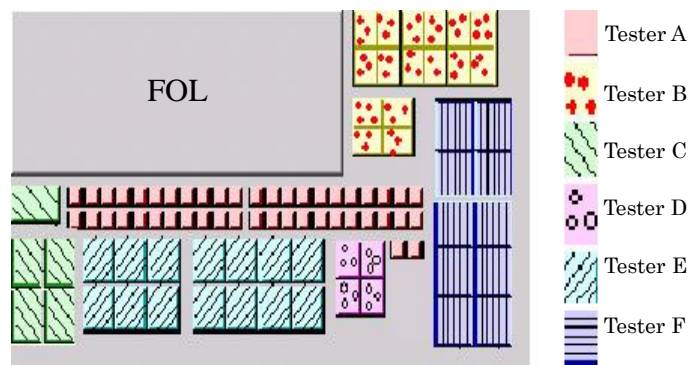


Figure 5.3 Experiment 6 result using GA

Crossover Rate	Numbers of Generation	Fitness Value
100%	2171	4593.9×10^4
50%	1184	4594.1×10^4

10.00%	625	4593.5 X 10 ⁴
--------	-----	--------------------------

Table 5.6 GA benchmark result using different crossover rate

Result from the prototype tool shows that:

Space utilization:

Although the result exhibit minimum waste space, all testers are able to fit inside the given empty space. The space utilization is optimal since there are empty existing on the outer side on the factory and only a small empty area (in the middle of tester yellow, blue and red) surrounded by testers.

Material flow:

The solution suggest managed to place tester red, yellow and green side by side with the FOL living pink, blue and purple on the outer side. This arrangement is optimal since the priority defined are red, yellow and green as the top three priorities and only follow by pink, blue and purple.

Manning ratio:

By using the cross box as guideline, the manning ratio is feasible for all tester respectively to their total machine. No arrangement prohibits the manning ratio that can be support.

Overlapping placement avoidance:

The result shows no overlapping space which is valid since the overlapping space scaling factor is set with a very large constant. This will enforce the goal to be as a rule instead of a goal.

In our second run using the same input parameters, result derived was as shown in Figure 5.4 using GP. The algorithm performance was tested using different crossover rate. The algorithm is stopped when it reaches the approximately similar results in term of fitness level as shown in Table 5.7. After running 483 generations with 10% crossover probability rate, it showed the best fitness value of 4593.5×10^4 . Interestingly, result obtained from GP is exactly the same as GA but converge faster than GA. Both GP and GA showed that a trend of lower crossover rate help in converging to a better solution.

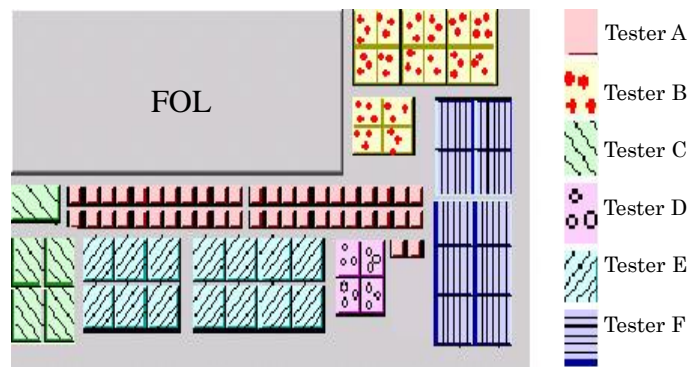


Figure 5.4 Experiment 6 result using GP

Crossover Rate	Numbers of Generation	Fitness Value
100%	1520	4593.9×10^4
50%	1462	4593.7×10^4
10.00%	483	4593.5×10^4

Table 5.7 GP benchmark result using different crossover rate

Result from the prototype tool shows that space utilization, material flow, manning ratio and overlapping placement avoidance are optimized as mentioned in earlier analysis for GA.

The summary of all three algorithms for experiment 6 is as shown in

Table 5.8. FSA found a better optimal solution and faster as compared to GP and GA. GP and GA converged to the same optimal solution but GP found the solution with lesser generations as compared to GA.

Algorithm	Fitness Value	Frequency	Time per cycle (50 samples)
FSA	4592.7×10^4	365263(iterations)	0.711 sec
GA	4593.5×10^4	625(generations)	6.36 mins
GP	4593.5×10^4	483 (generations)	8.05 mins

Table 5.8 Experiment 6 performance summary

In experiment 7, the existing factory floor had already occupied by existing tester B and other non-related testers. In this experiment, limited empty space and only two machines types were re-layout, A and B tester. Space utilization, material flow and manning ratio constraints could easier observe in this case study. The details of the testers are shown in Table 4.4.

The results from all three algorithms are the same and shown in figure 5.5. FSA converge to the optimal solution at 136 iterations, GA after running 13 generations and GP after 8 running generations. The numbers of iterations to converge to an optimal solution was less compared to experiment 6. This was due to the search space reduces as complexity reduce.

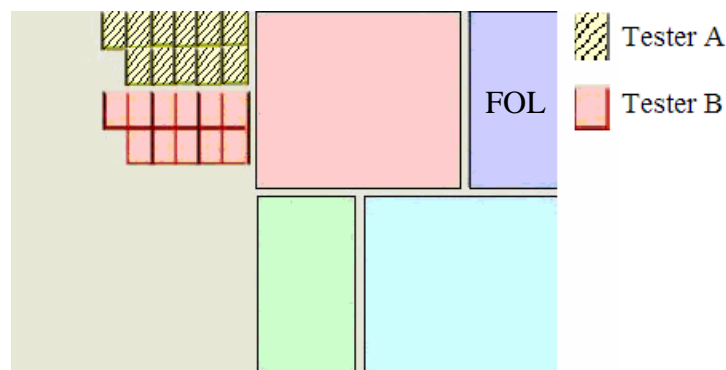


Figure 5.5 Experiment 7 result using FSA, GA and GP

Result from the prototype tool shows that:

Space utilization:

The space utilization is optimal but not fully utilized since one empty machine area was created in each tester yellow and red areas. This was due to the priority for manning ratio and material flow constraints have higher priority than space constraint.

Material flow:

Suggested solution showed that the 22 remaining tester red and blue were place at the available space which was closest to existing tester blue area. Material flow constraint was fully satisfied in this case study.

Manning ratio:

The manning ratio arrangement was feasible since both types of testers were arranged in a cross box manner. Manning ratio constraint was fully satisfied in this case study.

The summary of all three algorithms for experiment 7 is as shown in table 5.9. All three algorithms converge to the same optimal solution. FSA found the optimal solution the fastest and followed by GP and GA respectively.

Algorithm	Fitness Value	Frequency	Time per cycle (50 samples)
FSA	3.2×10^3	136(iterations)	0.711 sec
GA	3.2×10^3	13(generations)	6.36 mins

GP	3.2×10^3	8 (generations)	8.05 mins
----	-------------------	-----------------	-----------

Table 5.9 Experiment 7 performance summary

Based on Figure 4.2, the fitness value calculated is 3.5×10^3 . This means the solution found by the model is actually better alternative to the manual layout design. This simple experiment shows that although a layout design visually looks good but without a method of measurement we may overlook other better option.

Both experiments 6 and 7, show that the model is able to optimize the multiple constraints problem faced by factory floor planning. The numbers of machines and types drastically affect the search space since factory floor planning is a NP-complete problem.

The finally experiment 8 is done using real data from existing NCO6 factory layout. Table 4.5 shows the breakdown of all machines in quantities. This experiment is to test this model result against the existing layout which was done through many rounds of layout retrofit designed manually. To evaluate the quality of layout the fitness value is computed using the evaluation functions.

Based on the modal evaluation, it shows that the existing NCO6 layout in Figure 5.6 having the fitness value of 3694.7×10^4 and the fitness value for the suggested layout in figure 5.7 is 2519.93×10^4 using FSA. The different of both fitness values is 117484.64 and translated as 31% better than the existing layout in term of fitness value since we are minimizing our fitness value.

By visual inspection, Figure 5.6 showed testers A, B, C and F are scattered around. The layout of these testers with empty spaces creates inefficiency material flow, space utilization and manning ratio. Contrary from

Figure 5.6, Figure 5.7 group up all testers which waste space and provide an optimal manning ratio arrangement as defined in chapter 3.8. Lastly, the material flow was well arranged as the top priority testers are placed next by the FOL & EOL.



Figure 5.6 Existing NCO6, Penang factory layout



Figure 5.7 NCO6, Penang suggested factory layout

5.4 Experiments Results Summary

In case study 1, the model shows that it is able to reorganize a sliced square from seven rectangular components back to its original shape when space utilization is the only priority concerned. When material flow is the only priority concerned, all three algorithms converge to the same optimal solution.

With the two fitness value calculated from space utilization and material flow, we had derive a normalize scaling factors for b and c . It is observed that in experiment 3, the results obtained show a well balance of space utilization and material flow.

In case study 2, the optimal results obtained from the model are verified and endorsed by layout industrial engineers and manufacturing workgroup. In experiment 7 and 8, results found are better than initial solution from manual design in term of fitness value. These solutions are used for decision making and future improvement guidelines.

In term of algorithms performance, FSA shows better results in term of fitness value consistently compared to GA and GP as in Table 5.3, 5.8 and 5.9. As shown in Table 5.8 and 5.9, FSA has the fastest processing time followed by GP and GA.

CHAPTER 6.0

CONCLUSIONS AND FUTURE WORKS

6.1 Conclusions

In this research, we had developed and observed the performance between FSA, GA and GP on facilities layout optimization. The experiments proved that the prototype tools are able to improve a given initial solution using the proposed methodologies. Results generated in our experiments are verified and endorsed by experienced layout industrial engineers from Intel Manufacturing, Penang. The model allows flexibility in layout planning for new factory start up as well as existing factory with existing objects.

From our case study 1, we had showed the importance of setting the right scaling factor to avoid bias between different optimization goals. By using the extreme scenario for each optimization goals, the values derived can be used to estimate normalization scaling factors. Upon deriving all normalized scaling factors a more accurate optimization results can be archived with fine tuning.

However, interestingly FSA converges to an optimal solution with less iteration as compared to GP and GA. GP converged to an equivalent or better optimal solution found by GA with less generations being generated. Based on our hypothesis, GA did not perform as comparatively with the rest of the two algorithms due to chromosome encoding. Having our layout representation

design in expression tree it does not work well after translating to chromosome representation. The crossover method in GA after encoding to chromosome exhibit worse fitness state candidate from its parents and better off with less crossover rate. A suitable layout representation and crossover method will need to be research in order to make the GA or GP for improvement.

FSA had consistently performance better than GA and GP in both case studies. This helped us to conclude that our model is suitable to run with FSA than GA and GP, since both evolutionary searches are more random. By observing the time per cycle, GA and GP are slower than FSA since both evolution search methods are computational intensive.

6.1 Future Works

This research presented a two dimensional layout optimization specifically for semiconductor factory. This research did not consider three dimensional constraints. In new factory design, it may have walkable ceiling, air ventilation and etc. Certain machine has piping connected to the ceiling. At such situation, the model requires to factor in void area at the ceiling as well and not just the empty spaces of the factory floor.

In addition, the optimization goal's scaling factors used in the objective function can be improved by establishing an estimation function to predict normalized weights. The tuning of the optimization goals priority is an optimization research itself where it can be adopted by all general optimization models usage. This will be future work to study on multiple objectives function for SA (Sanghamitra, 2008), GA (Abdullah, 2006) and GP

(Lavinia, 2009).

As shown in our experiment 2, 3, 4, 6 and 7, the best optimal solution found is by FSA, followed by GP and GA respectively. This shows that our model implementation is suitable to run with a greedy search algorithm. Improvement for GA and GP, require a better representation structure for layout and crossover operator. The crossover operation can be improved to generate more feasible offspring. The new offspring are important as they help the new generation to converge to an optimal solution.

In our modal, mutation rate and crossover rate is a constant setting. Mutation rate for both GA and GP is set to 10% and crossover rate is set from 10%, 50% and 100%. As for future research, these two operations can be further refined with an optimal setting based on the problem statement. Through calculating the error threshold of the selection process, there's a correlation between the mutation and crossover rate (Gabriela, 2000).

REFERENCES

- Abdullah, K. ., D. W. C. & A. E. S., 2006. Multi-objective Optimization using Genetic Algorithms. *Reliability Engineering and System Safety*, September, pp. 992-1007.
- Aleisa, E. E. & L. L., 2005. *For Effective Facilities Planning: Layout Optimization then Simulation, or Vice Versa*. s.l., Proceedings of Winter Simulation Conference.
- Chang, Y. C. ., C. Y. W. ., W. G. M. & W. S. W., 2000. *B*-trees: A New Presentation for Non-slicing Floorplans*. Los Angeles, Proceedings of ACM/IEEE Design Automation Conference.
- Chen, T. C. & C. Y. W., 2006. Modern Floorplanning Based on B*-trees and Fast Simulated Annealing. *IEEE Trans. Computer-Aided Design*, 26(4).
- Chiang, W. C. & K. P., 1996. Improved Tabu Search Heuristic for Solving Facility Layout Design Problems. *International Journal of Production Research*, Volume 34, pp. 2565-2585.
- Gabriela, O. ., I. H. & H. B., 2000. *Optimal Mutation Rates and Selection Pressure*. s.l., Proceedings of the Genetic and Evolutionary Computation Conference.
- Garey, M. R. & J. D. S., 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: WH Freeman.
- H'astad, J., 2003. Inapproximability— Some History and Some Open Problems. *Proceedings 18th IEEE Annual Conference*, Volume 18, pp. 265-266.
- Holland, J. H., 1992. *Adaptation in Natural and Artificial Systems*. Cambridge: MIT Press.
- Jaime, G. P. D. A. S. & R. L. W., 1996. *Solving Facility Layout Problems using Genetic Programming*. Massachusetts, GECCO Proceeding of the First Annual Conference on Genetic Programming.
- Koopmans, T. C. & B. M., 1957. Assignment Problems and the Location of Economic Activities. *Econometrica*, Volume 25, pp. 53-76.
- Koza, J. R., 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge: A Bradford Book.
- Kusiak, A. & H. S., 1987. The Facility Layout Problem. *Eur J Operational Research*, Volume 29, pp. 229-251.
- Laursen, P. S., 1993. Simulated Annealing for the QAP-Optimal Tradeoff Between Simulation Time and Solution Quality. *Eur J Operational Research*, Volume 69, pp. 238-243.

- Lavinia, F. & A. P., 2009. Multiobjective Genetic Programming for Nonlinear System Identification. *Springer Link Adaptive and Natural Computing Algorithms*, Volume 5495, pp. 233-242.
- Lin, W. Y. & L. W. Y., 2003. Adapting Crossover and Mutation Rates. *Journal of Information Science and Engineering*, Volume 19, pp. 889-903.
- Lu, H. C. & H. C., 2008. Using a Genetic Algorithm Layout Design Tool to Optimise Facility Layout Designs in the Capital Goods Industry. *Conference Name: The Second International Conference on Operations and Supply Chain Management*, pp. 273-278.
- Mecklenburgh, J. C., 1985. *Process Plant Layout*. New York: Longman.
- Riccardo, P. ,. W. B. L. & N. F. M., 2008. *A Field Guide to Genetic Programming*. s.l.:Creative Common.
- Sanghamitra, B. & S. S., 2008. A Simulated Annealing-Based Multiobjective. *IEEE Transactions on Evolutionary Computation*, 12(3), pp. 269-283.
- Singh, S. P. & S. R. R. K., 2006. A Review of Different Approaches to the Facility Layout Problems. *The International Journal of Advanced Manufacturing Technology*, 30(September), pp. 5-6.
- Tam, K. Y., 1992. Genetic Algorithms, Function, Optimization and Facility Layout Design. *European Journal of Operational Research*, Volume 63, pp. 322-346.
- Yong, L., 1992. *Heuristic and Exact Algorithms for the Quadratic Assignment Problem*. Pennsylvania: Pennsylvania State University University Park.