MODELLING AND OPTIMISATION OF ASSEMBLY LINE BALANCING PROBLEM WITH RESOURCE CONSTRAINT

NUR HAIRUNNISA BINTI KAMARUDIN

Master of Engineering (Mechanical)

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : Nur Hairunnisa Binti Kamarudin

Date of Birth : 7 August 1985

Title : Modelling and Optimisation of Assembly Line Balancing Problem with Resource Constraint

Academic Session : SEM 2 2018/2019

I declare that this thesis is classified as:

☐	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐	RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑	OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

_____	_____
(Student's Signature)	(Supervisor's Signature)
850807-02-5964	Dr. Mohd Fadzil Faisae Bin Ab. Rashid
New IC/Passport Number	Name of Supervisor
Date:	Date:

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Master of Engineering (Mechanical).

_____

(Supervisor's Signature)

Full Name : DR. MOHD FADZIL FAISAE BIN AB. RASHID

Position : ASSOCIATE PROFESSOR

Date :

STUDENT'S DECLARATION

I hereby declare that the work in this thesis 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 Universiti Malaysia Pahang or any other institutions.

_____

(Student's Signature)

Full Name : NUR HAIRUNNISA BINTI KAMARUDIN

ID Number : MMM14007

Date :

MODELLING AND OPTIMISATION OF ASSEMBLY LINE BALANCING PROBLEM WITH RESOURCE CONSTRAINT

NUR HAIRUNNISA BINTI KAMARUDIN

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Master of Engineering (Mechanical)

Faculty of Mechanical Engineering
UNIVERSITI MALAYSIA PAHANG

APRIL 2019

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to my supervisor Dr. Mohd Fadzil Faisae for imparting his knowledge and expertise in this research. Without his guidance, encouragement and patience this research will not be complete.

I would also like to express my very profound gratitude to my parents and husband, for providing me with unfailing support and continuous encouragement through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Finally, to my darling Zareef for being such a good and understanding little boy throughout my years of study, and making it possible for me to complete what I started. To those who indirectly contributed in this research, your kindness means a lot to me. Thank you very much.

ABSTRACT

Assembly Line Balancing (ALB) is about distributing the assembly tasks into workstations with the almost equal workload. Previous research mostly assumed that all workstations are having similar capabilities including the machines, tools and worker skills. Recently, researchers started to consider the resource constraints in ALB such as machine and worker. Optimisation of ALB with resource constraints gives a huge benefit to the industry such as increase line efficiency, optimise the resources utilisation and can reduce production cost. This research presents Assembly Line Balancing with resource constraints (ALB-RC) for a simple model with the objectives to minimise the workstation, machine and worker. For the optimisation purpose, this research introduces Genetic Algorithm (GA) with two new crossovers. The crossovers are developed using a ranking approach and known as rank-based crossover type I and type II (RBC-I and RBC-II). The GA with new crossover is tested against popular combinatorial crossovers with a wide range of problem difficulties consisting of 17 benchmark problems. The performance of the proposed GA with new crossover in optimisation ALB-RC is finally validated using an industrial case study. The computational experiment results indicated that the proposed GA with new crossovers are able to find the optimal solution for ALB-RC better than popular combinatorial crossovers. Meanwhile, the results of industrial case study validated that the proposed ALB-RC model is capable to be used for the real industrial problem. At the same time, the result indicated that the GA with rank-based crossover is capable to optimise real-life problem. As a comparison, the number of workstation, machine/tools and workers had reduced between 10 – 15% for the optimised layout using GA with RBC, compared with the original layout in the case study problem.

ABSTRAK

Pengimbangan rangkaian pemasangan (ALB) adalah berkaitan dengan pembahagian tugas pemasangan ke stesen kerja untuk mendapatkan beban kerja yang hampir sama di setiap stesen. Kajian-kajian terdahulu mengandaikan setiap stesen kerja mempunyai keupayaan yang sama dari segi mesin, peralatan dan kemahiran pekerja. Sejak akhir-akhir ini, penyelidik mula mempertimbangkan kekangan sumber dalam mengimbangi rangkaian pemasangan seperti mesin dan pekerja. Pengoptimuman ALB dengan kekangan sumber memberi manfaat besar kepada industri seperti meningkatkan kecekapan pemasangan, memanfaatkan sumber sepenuhnya dan dapat mengurangkan kos pengeluaran. Penyelidikan ini membentangkan masalah pengimbangan rangkaian pemasangan dengan kekangan sumber (ALB-RC) dengan objektif untuk meminimumkan stesen kerja, mesin dan bilangan pekerja. Untuk tujuan pengoptimuman, penyelidikan ini memperkenalkan algoritma genetik (GA) dengan dua silangan baru. Silangan ini dibangunkan dengan menggunakan pendekatan kedudukan dan dikenali sebagai jenis silangan berasaskan kedudukan jenis I dan jenis II (RBC-I dan RBC-II). GA dengan silangan baru diuji terhadap silangan-silangan gabungan sedia ada yang popular. Ia diuji ke atas 17 masalah piawai ALB. Prestasi GA yang dicadangkan dengan silangan baru pada pengoptimuman ALB-RC akhirnya disahkan menggunakan kajian kes industri. Hasil eksperimen pengkomputeran menunjukkan bahawa GA yang dicadangkan dengan silangan baru dapat mencari penyelesaian optimum yang lebih baik untuk ALB-RC berbanding silangan yang sedia ada. Sementara itu, keputusan daripada kajian kes di industri mengesahkan model ALB-RC yang dicadangkan boleh digunakan di lapangan industri yang sebenar. Pada masa yang sama, keputusan kajian kes juga menunjukkan GA dengan silangan yang baharu mampu mengoptimumkan masalah sebenar di industri dengan lebih baik. Sebagai perbandingan, bilangan stesen kerja, mesin dan pekerja telah dapat dikurangkan di antara 10 – 15 % melalui susunatur yang dioptimakan menggunakan GA dengan silangan baharu berbanding dengan susun atur asal di industri.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

A_s	Workstation availability index
ct	Cycle time
ct_{max}	Maximum cycle time
$\bar{f}$	Normalised objective function
F(x)	Combined fitness function
$f_i(x)$	i^{th} objective function
n	Number of assembly task
nws	Number of workstation
pt_i	Processing time in workstation *i*
t_i	Assembly time for task *i*
w_i	Constant weight for $f_i(x)$
y_{ms}	Machine availability index
z_{ws}	Worker availability index

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimisation
ALB	Assembly line balancing
ALB-RC	Assembly line balancing with resource constraints
CX	Cycle Crossover
DP	Dynamic programming
GA	Genetic Algorithm
GALBP	Generalised Assembly Line Balancing Problem
LE	Line efficiency
LP	Linear programming
MMAL	Mixed Model Assembly Line
MOGA	Multi-objective genetic algorithm
MuMAL	Multi Model Assembly Line
NLP	Non-linear programming
NP-hard	Non-deterministic polynomial-time hard
OX	Ordered Crossover
PM	Precedence matrix
PMX	Partially Matched Crossover
PSO	Particle Swarm Optimisation
RBC-I	Rank based crossover type I
RBC-II	Rank based crossover type II
SALBP	Simple Assembly Line Balancing Problem
SALBP-1	Simple Assembly Line Balancing Problem Type 1
SALBP-2	Simple Assembly Line Balancing Problem Type 2
SALBP-E	Simple Assembly Line Balancing Problem Type E
SI	Smoothness index
TSAL	Two sided Assembly Line
USAL	U-Shaped Assembly Line

CHAPTER 1

INTRODUCTION

1.1 Background

The manufacturing sector currently is evolving to a new phase and becoming more productive in producing a product. The manufacturer tends to compete on the cost and move up the manufacturing value chain. In order to increase line efficiency and optimise the assembly line output, a workforce and resources that are efficiently skilled and fully utilised play a major role (Amin & Karim, 2013).

There are many important aspects of manufacturing that need to be considered in order for it to be profitable for the company. One of the important aspect is the assembly process (Al-Ahmari et al., 2018). It is a process which involves joining parts together and produce the desired product on an assembly line. Assembly line usually consists of a number of task as well as workers, machines and tools carried by a number of workstation along the line (Sikora, et al., 2017). The main issue in the development of an assembly line is the arrangement of tasks, workers and resources to be performed (Grzechca & Foulds, 2015). A balanced assembly line will determine the production rate by making sure that each station has the same amount of work so that the idle time will be minimised.

Optimisation of assembly line balancing is vital to ensure the waste from the assembly process is minimal and in turn will minimise the cost of the production (Yin et al., 2018). Assembly optimisation in the production deals with the determination of optimum assembly sequence and determination of the optimum location of each resource (Rashid et al., 2012).

Due to the complexity of assembly line balancing problem, it is crucial to have an optimum seeking the best solution method which practical for instances of more than a

few tasks and/or workstations (Jusop & Ab. Rashid, 2016). Various methods, approaches and procedures were introduced for solving the assembly line balancing problem that can satisfy production rates and at the same time can achieve the desired objective function such as linear programming, integer programming, dynamic programming and branch-and-bound approaches (Tasan & Tunali, 2008). Currently, the development of heuristic and meta-heuristic approach is inevitable since it is more practical for large problems and can locate an optimum solution.

Therefore, finding the best solution for assembly line balancing with resource constraint is significant for the manufacturing industry. Development of an algorithm which can allow and consider multi-skilled workers which can perform tasks that have been assigned is crucial in order to reduce the number of workers to be placed on the assembly line. Concerning on machine/tools assignment, the duplication of those resources along the assembly line can be reduced by selecting task using the same type of resources together in the same workstation. This research details the methodology and Genetic Algorithm used to model and optimise assembly line balancing problem with resource constraints (ALB-RC).

1.2 Problem Statement

Assembly Line Balancing Problem (ALBP) is the process of assigning tasks to workstations so that the predetermined objective function is satisfied, the production target is achieved and the constraints are not violated. The objective functions for ALBP can be to minimize number of workstations, to minimize the cycle time, or the production cost. Every task in assemble line are allocated to the workstation with respect to the assembly line constraints such as precedence relation between task and as well as limited processing resources (Taylor, 2010).

When assigning task to workstation, the main target for the line design is to minimize the number of workstations. This is because, the activation of workstation on assembly line will incur extra cost due to consumption of energy, maintenance of equipment, setup activities or labor requirement (Kovalev et al., 2017).

In majority of previous works, researchers make assumptions where any of assembly tasks can be processed or assembled in any workstations and resources are available without limit (Dong et al., 2018). This is certainly true for the product which

only requires a common or simple tool to be assembled. However, when the complexity of a product increased, it requires a special tool, machine or highly skilled labour to assemble that particular component. The optimization method that is used to achieve that objective function may produce a solution that use a high number of workers or duplication of machines/tools along the line. To satisfy the assembly plan, these resources may not be available or require extra cost (Mura & Dini, 2016). Therefore, the limitation of resources will be another constraint for the industry. This problem is known as assembly line balancing with resource constraints (ALB-RC).

In assigning assembly task to workstation with cosideration of constraints, there are many possibilities of assembly sequence to be evaluated in achieving the most optimum and best solution for ALBP (Azizoğlu & İmat, 2018). The possible task of sequence for *n* task and *r* constraints will be $n!/2^r$ (Baybars, 1986). This field of complexness in solving the problem deals with how fast can the propose method solve the problem by using resources like time, memory-space, number of processors, etc. The collection of all problems that can be solved in polynomial time using nondeterministic is called NP. Due to ALB has a computational complexity of the problem and the solution space is excessively increased when the number of tasks is increased, ALB is classified as NP-hard. As NP-hard, heuristics approach by using a good algorithm is the best approach used to solve and optimise ALBP with the larger size, various constraints and objectives (Rashid et al., 2012). The heuristic approach will find the nearest optimal and best solution for the problem in less time compared to other approaches.

1.3 Research Objectives

The research objectives are:

i. To establish a model of Assembly Line Balancing with resource constraints (ALB-RC) for Simple Assembly Line Balancing Problem Type 1 (SALBP-1).
ii. To propose an improved algorithm to optimise ALB-RC problem.
iii. To validate the proposed ALB-RC model and algorithm through industrial case study.

1.4 Scopes

This research explores the optimisation of assembly line balancing problem with resource constraints (ALB-RC). This research is limited to a simple model assembly problem type 1 (SALBP-1) with the objective of minimising number of workstation for a given cycle time.

In different with a large number of existing research in assembly line balancing which assume each of assembly workstation have similar capabilities and equipment, this research considers the resources constraint of assembly plant (including machine/tool and skill workers) as it is nearer to the real situation in industry.

This research focuses on optimisation of ALB with resource constraints using metaheuristic method that targeting a high-quality solution for large and complex problems. Therefore in this research, Genetic Algorithm with two new crossovers are introduced to search for the best feasible solution.

The review of the literature for this research focuses on the classification of ALB problem, resource constraints in assembly line and the methods used by previous researchers to optimise assembly line with resource constraints. In addition, the problem is also limited to a simple assembly line problem which has accumulated a large number of works in the literature.

In this research, the proposed crossovers will be tested against popular combinatorial crossovers using benchmark problems. The generic test problem which varies in term of the size is obtained from http://assembly-line-balancing.mansci.de/ (Scholl, 1993). The benchmark test problem is set from small size problem started with 7 number of task to large size problem; 148 tasks. The proposed crossover is then will be compared with other types of crossover and solved in similar problems. It is then tested and applied to a real case study which involved a similar problem for validation purpose.

1.5 Thesis Organization

This thesis is structured into five chapters, and organised as follows:

Chapter 1: gives an overall view of the research including background and basic principles of assembly line balancing problem. It also presents the research problem and motivation.

Chapter 2: reviews related literature in ALBP and optimisation method used by previous researchers including Genetic Algorithms and implementation of Genetic Algorithms to solve SALBP. In this chapter, the literature survey is also performed to identify the research trends and research gaps in the area.

Chapter 3: presents the research methodology to present the overview of how this research is conducted. In this chapter, the problem modelling and Genetic Algorithm with Rank-based Crossovers used to solve the benchmark problem and case study from real manufacturing problem is presented and explained in details.

Chapter 4: present the result tested to the benchmark problem and the real manufacturing problem and followed by result discussion.

The final chapter, **Chapter 5:** concludes the research findings in general specifically on the achievements of research objectives and the limitations of this research. This chapter also suggests recommendations for future works.

CHAPTER 2

LITERATURE REVIEW

2.1 Classification of Assembly Line Balancing

Assembly lines are flow-oriented production system, which contains a several number of workstation. These workstations are arranged in series, parallel, two-sided or U-shaped assembly line base on the production requirement and suitability. Assembly line balancing (ALB) plays a vital function in a production system. The installation of an assembly line is a long-term decision and requires large capital investments. It is important that such a system is designed and balanced so that it works as efficiently as possible (Becker & Scholl, 2006).

Every workstation has a deterministic time or a production rate for the station to perform the carried task assigned. This production rate is set so that the desired amount of end product is produced within a certain time period (Baybars, 1986). This time is called cycle time and each task will have its own process time. If the sum of the processing times within a station is less than the cycle time, idle time is said to be present at that station (Erdal Erel et al., 1998).

Every task should be assigned to workstations by considering the precedence relation between tasks. These precedence relationships can be shown on a precedence diagram as in Figure 2.1. Precedence diagram is the illustrative representation of sequence of task in order of occurrence. The arrows linked between task shows that a task can be performed only after its predecessor tasks are performed. As an example, task number 7 can only be performed after task number 3, 4 and 5 completed.

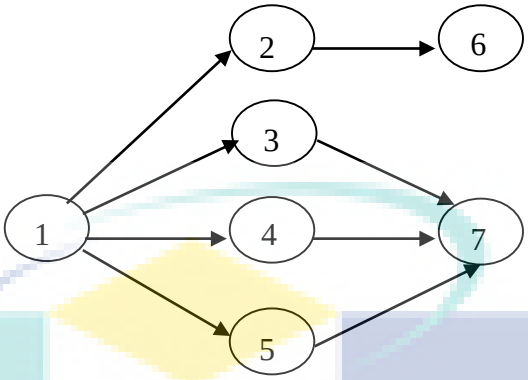


Figure 2.1 Precedence relation of 7 tasks example problem

Balancing an assembly line is usually comprise of assigning task to be processed to workstation which will optimise certain objectives such as number of workstations, cycle time, or the production cost. A grouping of the task to the workstation which satisfies a determined goal is known as the assembly line balancing problem (ALB) (Boysen et al., 2007). ALB can be classified into two groups; Simple Assembly Line Balancing Problem (SALBP) and Generalised Assembly Line Balancing Problem (GALBP) (Baybars, 1986).

2.1.1 Simple Assembly Line Balancing Problem (SALBP)

SALBP is the simplest version of ALBP. SALBP involves the production of single homogenous product as in Figure 2.2. This line comprises of number of workstations along a straight line in one-sided workstation. Each workstation will have certain assigned task and all tasks in the workstation must be complete before the process can be move to another workstation. The final product is produced after all tasks on every workstation are executed.

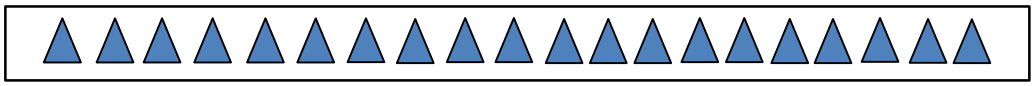


Figure 2.2 Single model assembly line

SALBP can be categorised to four (4) different types based on the objective function as in Table 2.1.

Table 2.1 Classification of SALBP

Type	Given	Objective
SALBP-1	Cycle Time	Minimise no. of workstation
SALBP-2	Number of workstation	Minimise cycle time
SALBP-E	-	Maximise line efficiency
SALBP-F	Cycle Time & Number of workstation	Obtain feasible balance

Simple Assembly Line Balancing Problem Type 1 (SALBP-1): The objective in SALBP-1 is to minimise the number of workstations, n_{ws} on the assembly line for a given cycle time, ct. In the process of assigning task to workstation, the task without predecessor or the task that the predecessors have already been assigned will be considered to be assinged to the station with consideration that the processing time for the task is less than or equal to the time that still available for the station. If no task is found, a new station will be opened (Dolgui & Proth, 2013).

Simple Assembly Line Balancing Problem Type 2 (SALBP-2): The objective function for SALBP-2 is to minimise the cycle time, *ct* for a given number of stations, n_{ws} on the line. The optimization of SALBP-2 can maximise the production rate of an existing assembly line (Dolgui & Proth, 2013).

Simple Assembly Line Balancing Problem Type E (SALBP-E): SALBP-E is the most general problem version. The objective function is to maximise the line efficiency by minimising the cycle time and number of workstations. The line efficiency is calculated by using following equation:

$$LE = \frac{\sum_{i=1}^{nws} pt_i}{nws \times ct} \times 100 \quad (2.1)$$

where n_{ws} is number of workstations, pt_i is processing time in workstation i^{th} and ct is the cycle time.

Simple Assembly Line Balancing Problem Type F (SALBP-F): SALBP-F having the objective function to get a feasible balance for a given number of workstation, n_{ws} and given cycle time, ct.

SALBP is based on a set of limiting assumptions which used to reduce the complex problem of assembly line configuration to focus on the actual problem when assigning tasks to workstations (Boysen et al., 2008). However, in balancing of real world SALBP will also involves the observation of additional aspects which affect the structure of the optimization solution. The extension of problems in SALBP can be solved simultaneously with the main objectives function of SALBP.

SALBP has accumulated a large number of works and has deal with single objective as well as multi-objectives situations. Kao et al. (2010), Taylor et al. (2010), M Fathi (2011), Scholl et al. (2011), Pape (2015) considered SALBP-1 in their research. While, Mutlu et al. (2013), Borba & Ritt (2014), Triki et al. (2014), Zacharia & Nearchou (2016), Moreira at al. (2017) considered ALBP Type 2. Only a small number of previous research study on SALBP-E and SALBP-F as it is more complicated compare with SALBP-1 and SALBP-2.

2.1.2 General Assembly Line Balancing Problem (GALBP)

Meanwhile, General Assembly Line Balancing Problem (GALBP) includes the entire problems that are not considered in SALBP. GALBP have a mixed model assembly line balancing, parallel, U-shaped and two-sided lines with stochastic dependent processing times (Tasan & Tunali, 2008).

Mixed Model Assembly Line (MMAL): Mixed-model assembly lines (MMAL) allow the simultaneous assembly of different products on a single assembly line in an intermixed sequence as in Figure 2.3. Different models of products could be parts of a

base product or constitute a special package of products, so inherently their assembly process is somehow similar and assembly models mostly differ in performance times. (Ramezanian & Ezzatpanah, 2015)

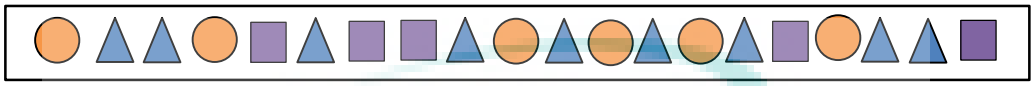


Figure 2.3 Mixed-model assembly line

Multi-Model Assembly Line (MuMAL): In Multi-Model Assembly Line (MuMAL), the product produce is varied and are manufactured in batches. Whenever another batch of product is to be processed, a setup which requires time and resources will occur (Boysen et al., 2007). The ALBP for MuMAL also involves a lot sizing problem instead of batch sequencing. The assembly line for MuMAL is presented in Figure 2.4.

Figure 2.4 Multi-model assembly line

U-Shaped Assembly Line (USAL): The main characteristic of a U-Shaped assembly line layout which make it different from a straight assembly line is that the starting point and exit point of a product is at the same position and workstations are arranged around a U-shaped line as in Figure 2.5. The U-shaped assembly line has shown a better advantage over the straight assembly line since worker may perform more than one task located in different places of the assembly line. Moreover, USAL also allows more option in assigning tasks to workstations and therefore the number of workstations needed for USAL layout is less than the number of workstations needed for the traditional straight assembly line (Masood Fathi et al., 2018; Grzechca & Foulds, 2015).

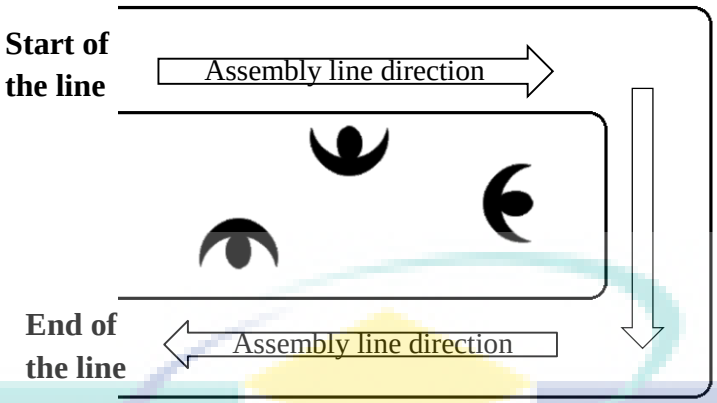


Figure 2.5 U-shaped assembly line

Two-sided Assembly Line (TSAL): Two-sided assembly lines as in Figure 2.6 are usually engaged in an industry that produces a large-sized high volume products (Janardhanan, et al., 2018). Different assembly tasks are carried out on the same product at a specific side either on the left side or right side of the product (Tuncel & Aydin, 2014). The advantages of using TSAL in production line compared to the single assembly line is that the line length can be shorten, save the cost of tools and fixtures, and reduce the material handling and operator movement (Wang et al., 2014).

Figure 2.6 Two-sided assembly line

2.2 ALB Constraints

In obtaining an efficient and productive assembly line, the assembly line should be balanced and by assigning tasks to each station in a way that the objectives are satisfied, the demand is met and all the constraints are not violated. The constraints in an

assembly line can be categorised into two types which are absolute constraints and optimisation constraint.

2.2.1 Absolute Constraint

The absolute constraints are the constraints which will lead to infeasible assembly sequence if violated and when optimisation constraint is violated, it will effect on the quality of assembly sequence. Lower quality of assembly sequence will be produced (Rashid et al., 2012). The absolute constraints that may have in a SALBP-1 are the occurrence constraint, precedence constraint and cycle time constraint.

2.2.1.1 Assignment Constraint

Assignment / occurrence constraint will assure that each task cannot be split into two or more workstations. For this purpose, the assignment / occurrence constraint was formulated as in Equation 2.2 to ensure that each task is assigned to only one workstation. In this equation, the sum of workstations in which a task is assigned must equal to '1' and this is applied to all tasks on all lines (Kucukkoc & Zhang, 2015).

$$\sum_{i=1}^{n} x_{is} \leq 1 \qquad 2.2$$

2.2.1.2 Precedence Constraint

Precedence constraint is the connection between each task in the assembly process and can be represented in a precedence diagram as in Figure 2.1 or in matrix form as in Table 2.2. In this matrix, if task *i* is a predecessor of task *j*, PM (*i, j*) = 1. Otherwise, the matrix will be left empty.

Table 2.2 Example of precedence matrix (PM)

i	***j***						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1		*1*	*1*	*1*	*1*		
2						1	
3							1
4							1
5							1
6							
7							

The possibilities of having a feasible solution arrangement of task to workstation without violating the precedence relation is by assigning task *i* earlier in other workstation than task *j* or assigning task *i* on the same queue with task *j* on the same workstation but task *i* is started and completed before task *j* is started (Kucukkoc & Zhang, 2015).

2.2.1.3 Cycle Time Constraint

Meanwhile, for cycle time constraint, the total operation times of the tasks in a workstation should not be greater than the cycle time. The formulation for cycle time constraint are depend on the type of ALB. For SALBP-1, the equation for cycle time constraint are formulated as follows (Rashid et al., 2012).

$$\sum_{i=1}^{n} pt_i \cdot x_{ij} \leq C \qquad 2.3$$

In this equation, pt_i refers to the processing time for task *i* and *C* is predetermined cycle time for the assembly line.

2.2.2 Optimisation Constraint

Optimisation constraint is the constraint that presence in the process of optimizing the desired objective function with respect to some variables.

2.2.2.1 Space Constraint

In an automotive industry which usually involves two-sided assembly line balancing problem, besides assuring that tasks are assigned to workstations as the respective order of precedence between the tasks, space requirements are the other constraints to be considered (Bautista & Pereira, 2007). Space requirement for workstations is determined by the size of workstations where a few workstations will utilise a small space compared to opening more workstations which will increase the space requirement (Rada-vilela et al., 2013).

2.2.2.2 Zoning Constraint

The zoning constraints can be categorised into two types, positive zoning constraint and negative zoning constraint. Positive zoning constraints are related with assigning tasks that need the same equipment to the same workstation. On the other hand, negative zoning constraint happened due to technological issues and some tasks cannot be grouped in the same workstation for safety reasons (Akpinar et al., 2017).

2.2.2.3 Resource Constraint

ALB research works have also addressed problems that consider some other additional constraints apart from those three constraints. Assembly lines comprise of a combination of tasks in different workstations, one or more dedicated machines/tools together with workers. The issue of line balancing with the minimum or a limited number of resources (machines and workers) has always been a serious problem in the industry (Ağpak et al., 2005). Equipment and workers should assign to task and workstations so that maximum efficiency, maximum usage of resources and minimum number of workstations of the production line can be achieved. Therefore, this will reduce the total costs and the complexity of an assembly line (Mura & Dini, 2016).

Previously, researchers had studied the line balancing with resource constraints. Ağpak & Gökçen (2005) started the ALB-RC by considering two resources and solve the problem using integer programming. Next, Corominas et al. (2011) proposed a model to

support generalised constraints problem. Özdemir & Ayağ (2011) consider equipment constraint when assigning task to workstation for SALBP. In this paper, the researcher use branch and bound algorithm together with the analytic hierarchy process (AHP) method to determine the optimum solution in minimising the equipment cost for production line. Koltai & Tatay (2013) later proposed a model and optimise the ALB with worker skill constraint. The purpose is to match the assembly task with the level of the worker skill. Jayaswal & Agarwal (2014) conducted research on assign tasks to workstations, and resources (equipment and assistants) to tasks with the objectives function is to minimised total cost of workstation and resource utilisation. This research is modelled to a U-shaped assembly line balancing using Simulated Annealing. Besides that, Jusop & Rashid (2017) optimise the multi-objective ALB with general resources using domination concept.

2.3 Optimisation Approaches for ALB

A lot of assembly line balancing problems have been solved using various optimisation methods. Previously, traditional optimisation techniques such as linear programming (LP), non-linear programming (NLP), and dynamic programming (DP) have had major roles in solving these problems. However, the weaknesses in the traditional techniques produce demand for other types of algorithms, such as exact and heuristic approach. The objective of optimisation methods is to find an optimal or the closes optimal solution with low computational effort measured by the time (computation time) and space (computer memory) consumed by the method.

Exact approach is the method of choice if it can solve an optimisation problem with the effort that grows polynomials with the problem size. This method has guarantee in finding an optimal solution. Classical exact resolution methods (i.e. enumerative, branch and bound, dynamic programming, linear and integer programming, etc.) allow the finding of optimal solutions for assembly line balancing problem, but they are often extremely time-consuming and inefficient in solving large-scale problems due to the inherent NP-hard nature of the ALB (Yeh & Kao, 2009).

Researchers previously have developed many exact solution approaches to find the optimal task assignment for the ALB. Liu et al. (2008) use branch and bound

algorithms for solving benchmark problem on SALBP-1. Özdemir & Ayağ (2011) propose an integrated approach by using branch and bound algorithm together with the analytic hierarchy process to allocate the task to workstation and select equipment for SALBP. Borba & Ritt (2014) introduce a new heuristic and exact method to solve assembly line worker assignment and balancing problem. Esmaeilbeigi et al. (2015) use mixed integer programming for type E simple assembly line balancing problem. Vilà & Pereira (2013) solve assembly line worker assignment and balancing problem by using exact enumeration algorithm. However, all optimal approaches used in exact solution approaches are computationally inefficient in solving large-scale problems due to the inherent NP-hard nature of the ALB.

Heuristic and metaheuristic techniques are powerful and flexible search methodologies that have successfully solve the medium and the large-sized assembly line (Roshani & Giglio, 2015). Heuristic and metaheuristic algorithms seek to produce good quality solutions in a short time and good enough for practical purposes. As opposed to exact methods which guarantee to give an optimum solution to the problem, heuristic methods suggest an approximate not guaranteed optimal solution but sufficient enough to solve complex problems. This method is used to expedite the process of finding an optimal solution where the classic method is too slow (Desale et al., 2015). Thus, heuristic methods will be the option to solve real optimisation problems of ALB. On the other hand, metaheuristic is a higher level procedure of heuristic which provides a good solution to an optimisation problem even without inadequate or limited information. A few assumptions about the optimisation problem is created in metaheuristic approach so that it can solve very large spaces of candidate solutions (Desale et al., 2015).

Various metaheuristics methods were used by researchers in optimising ALB. Table 2.3 and Figure 2.7 shows the summary of papers which used different metaheuristic methods to optimise ALBP from the year 2005 until 2017. According to the diagram, the three most dominant optimisation methods, used in 55% of the cited research are Genetic Algorithm (GA), Ant Colony Optimisation (ACO) and Particle Swarm Optimisation (PSO).

Table 2.3 Summary of literature in optimising ALB

Method	Author, Year	01	02	03	04	05	06	07	08
Genetic Algorithm (GA)	(Rajakumar et al., 2006)	x							
	(Moon et al., 2009)		x						
	(Yu & Yin, 2009)	x		x					
	(Hamta et al., 2011)		x		x				
	(Purnomo et al., 2013)				x				
	(Sivasankaran & Shahabudeen, 2014)			x					
	(Triki et al., 2014)		x		x				
	(Alavidoost et al., 2015)			x		x			x
	(Barathwaj et al., 2015)			x			x		
	(Zacharia & Nearchou, 2016)				x		x		
	(Mura & Dini, 2016)			x				x	
	(Raj et al., 2016)				x				
	(Zhao, Hsu, Chang, & Li, 2016)					x			
	(Jusop & Ab. Rashid, 2016)			x	x			x	
	(Jusop & Ab. Rashid, 2017)				x				
Ant Colony Optimization (ACO)	(Bautista & Pereira, 2007)			x					
	(Rada-vilela et al., 2013)			x					x
	(Kucukkoc & Zhang, 2016)							x	
Particle Swarm Optimization (PSO)	(Chutima & Kid-Arn, 2013)		x	x					x
	(Yuguang et al., 2016)	x			x				x
	(Che, 2017)				x	x			
Teaching Learning Based Optimization (TLBO)	(D. Li et al., 2016)		x			x	x		
	(Tang et al., 2017)								x
Bees Algorithm	(Özbakir & Tapkan, 2011)			x					
	(Tapkan et al., 2012)			x			x		
Simulated Annealing	(Jayaswal & Agarwal, 2014)			x					
	(Roshani & Giglio, 2015)				x				
Late Acceptance Hill Climbing	(Yuan et al., 2013)			x					
	(Wang et al., 2014)			x	x				
Combination GA and ACO	(Akpınar et al., 2013)								x

Table 2.3 Continued

Method	Author, Year	01	02	03	04	05	06	07	08
Other Heuristic	(Xu & Xiao, 2009)			x					
	(Kao et al., 2010)			x				x	
	(Scholl et al., 2011)				x				
	(Sternatz, 2014)				x				
	(Ramezanian & Ezzatpanah, 2015)				x				
	(Z. Li, Tang, & Zhang, 2017)						x		
	(Akpinar et al., 2017)	x							

01 – Workflow balancing

02 – Minimise assembly cost

03 - Minimise number of workstation

04 - Minimise cycle time

05 - Maximise line efficiency

06 - Maximise workload smoothness

07 - Maximise utilisation

08 - Other

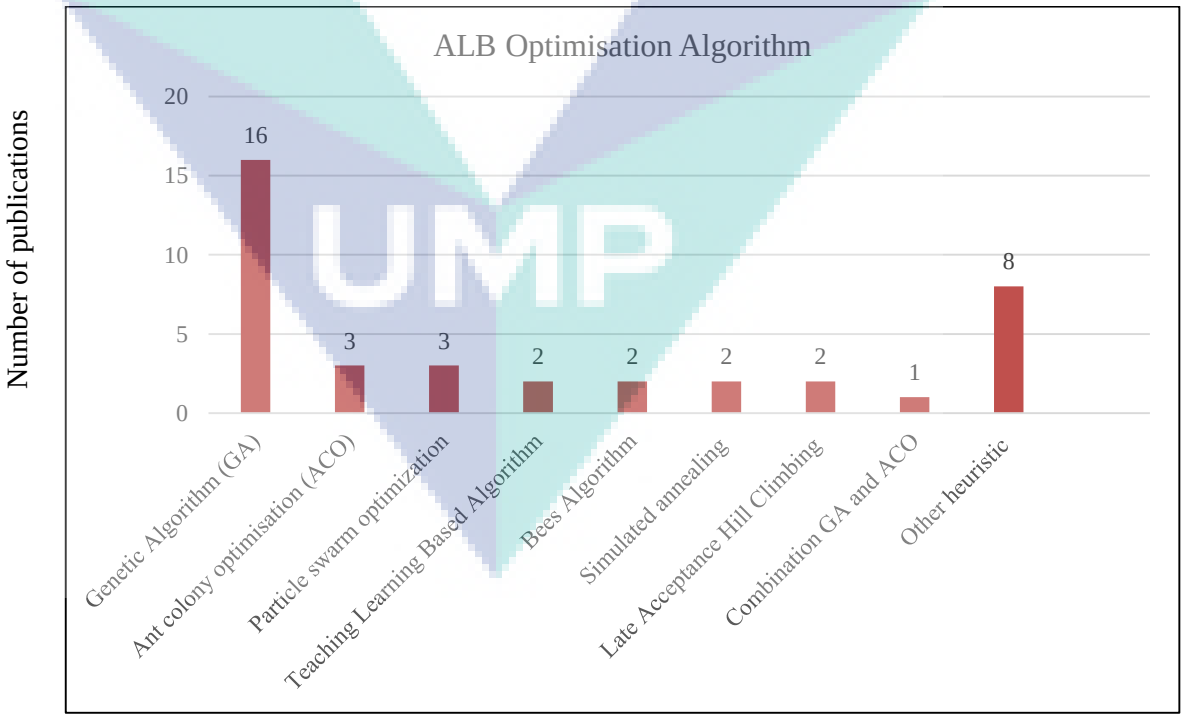


Figure 2.7 Previous research on ALB using heuristic/metaheuristic algorithm

2.3.1 Genetic Algorithm (GA)

Genetic Algorithm (GA) is one of the popular metaheuristic algorithms and received a huge number of attention from researchers compared to other type of metaheuristic optimisation approach. GA manipulating a population of solutions by randomly searching the best feasible solution in the solution space, based on the mechanism of natural selection and natural genetics (Zhang, 2018).

2.3.1.1 Process in Genetic Algorithm

In GA, every individuals or "chromosome" will be encoded to represent the solution vector. Through a number of generation, it is then being evaluated by using fitness function. From the fitness value, a number of individuals are selected to enter the mating pool in order to generate parent and off-spring individuals with the help of genetic operators. GA will continues to generate new chromosome to obtain the chromosome which provides the best solution for the problem (Mutlu et al., 2013). In general, the steps in GA are consist of :

- Initialisation
- Evaluation
- Selection
- Crossover
- Mutation
- Termination

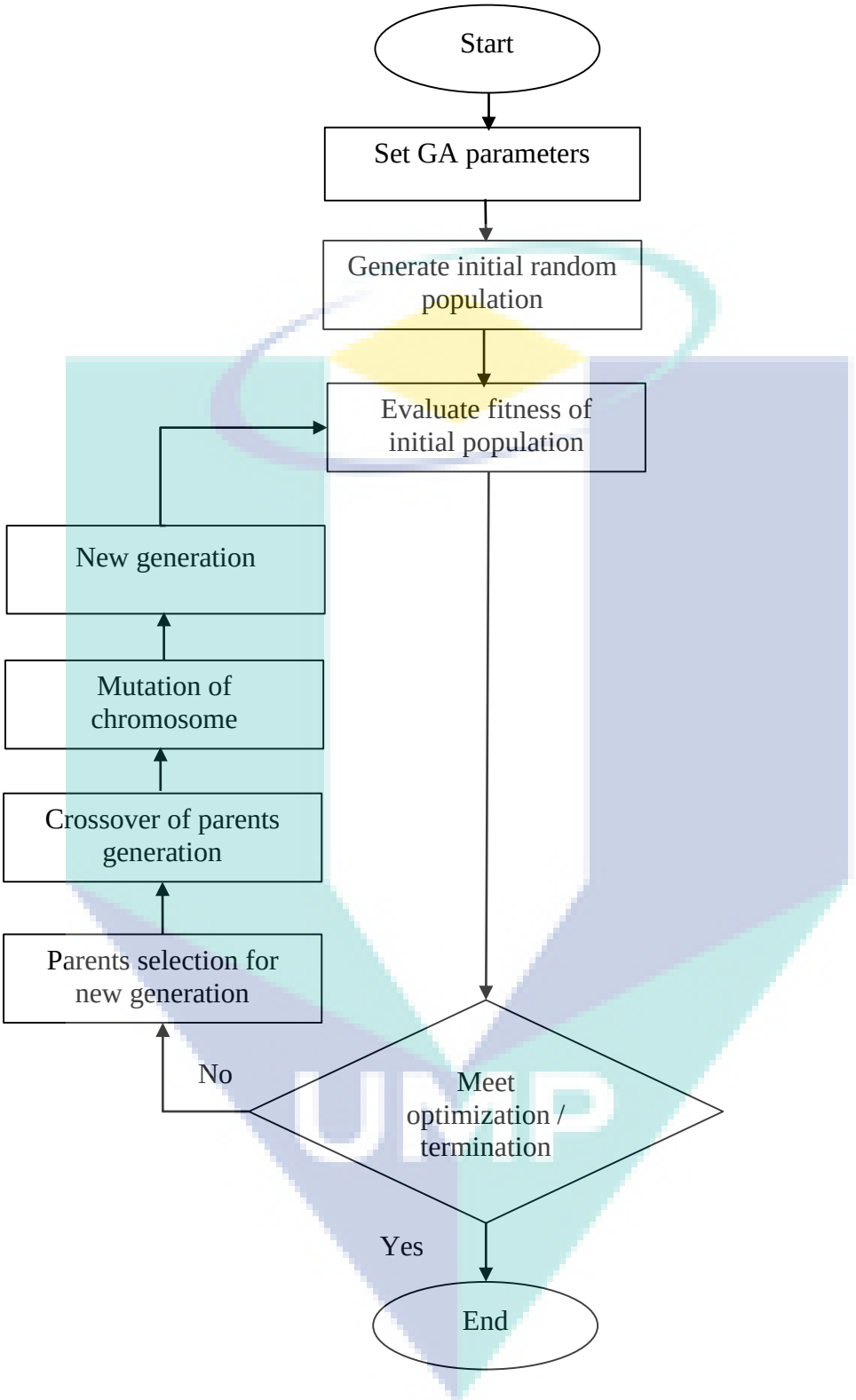


Figure 2.8 Flowchart of Genetic Algorithm

i. Initialisation

The first step in applying GA to a particular problem is to create the initial population randomly by converting the solutions (individuals) of ALB into a string type structure called chromosome. The encoding process is often the most difficult aspect of solving a problem using genetic algorithms due to the requirement in producing as much as feasible solutions to a particular problem so that it can be used in the crossover process.

The representation of solution can be binary numbers; zeros and ones {0, 1} or real numbers {1, 2, 3, ….., n}. The bit string chromosomes which consist of binary numbers; zeros and ones {0, 1} as in Figure 2.9 is the classical way to represent a solution. However, this simple binary strings in chromosomes are less suitable for the complex combinatorial problem since it usually produces infeasible solutions (Tasan & Tunali, 2008).

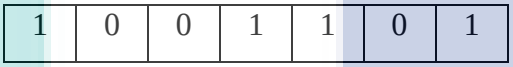


Figure 2.9 Binary representation

Real values representation is much more effective in representing the assembly line balancing problem. The most common real values representation in ALB is task-based representation where the chromosomes are represented by a feasible precedence sequences of tasks and is expressed in real number 1, 2, 3, 4, …, *n*. The length of the chromosome is defined by the total number of tasks to be arranged. For example, task-based representation for Figure 2.1 is illustrated in Figure 2.10.

Figure 2.10 Task-based representation

ii. Evaluation

The fitness of all individuals will be computed to measure how well the individuals of initial population optimise the given objective function. Fitness value will be function as a discriminator of the quality of solutions represented by the chromosomes

in a GA population. A chromosome that has a minimum fitness function value is the best chromosome and the fitness value will be used in the evolution of chromosome in the next steps of GA (Moon et al., 2009).

For multi objective optimisation problems, one of the simplest methods for combining multiple objective functions into a scalar fitness solution is by using weighted sum approach. In this approach, if there are some objective functions to be maximised, the combined fitness function *F(x)* is represented by (Triki et al., 2014):

$$F(x) = w_1 f_1(x) + w_i f_i(x) + \cdots + w_q f_q(x) \qquad 2.4$$

where,

x is a string

F(x) is a combined fitness function

$f_i(x)$ is the *i*th objective function

w_i is a constant weight for $f_i(x)$

q is the number of objective functions

iii. Selection

The selection of parent chromosomes for generating new off-springs with genetic operators are from the mating pool. Chromosomes are selected for recombination on the basis of fitness. Chromosome with higher fitness have a bigger chance of being selected more than once or even recombined with themselves compared to chromosome with lower fitness (McCall, 2005). The techniques that usually used for this purpose is roulette wheel, rank based selection and tournament methods.

In roulette wheel selection, the fitness values of the each individual will be scaled so that the total rescaled fitness values will be equal to one. The size of segment in the roulette wheel is related to the fitness value. Individuals with larger size segment in roulette wheel will have more probability of being selected (Razali & Geraghty, 2011).

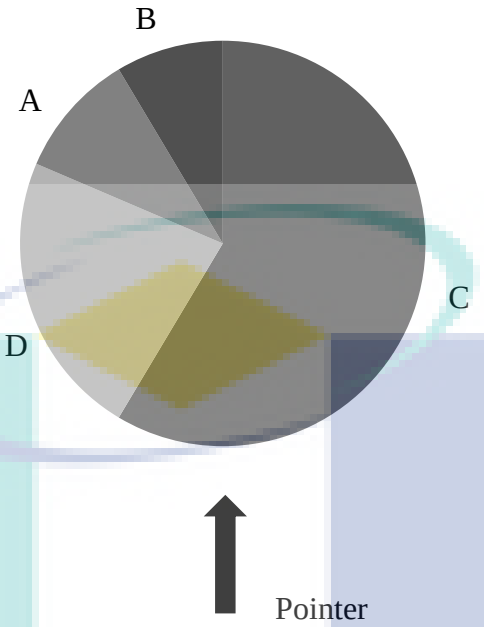


Figure 2.11 Roulette wheel

The probability of individuals selection is define as,

$$P_i = \frac{f_i}{\sum_j^n f_j} \qquad 2.5$$

where,

P_i is a probability of individuals 1, 2, 3, …, n

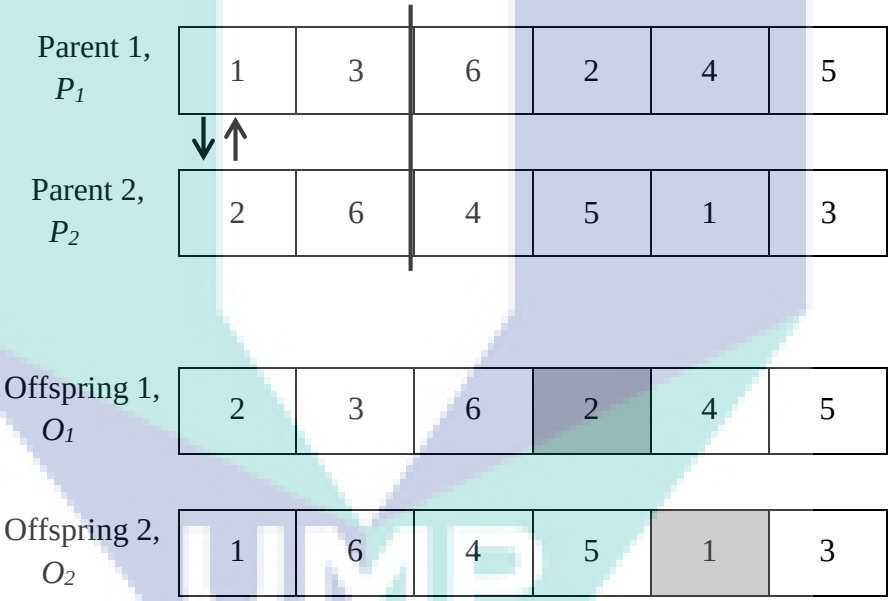
f_i is a fitness value of individuals 1, 2, 3, …, n

f_j is a total of fitness value for all individuals 1, 2, 3, …, n

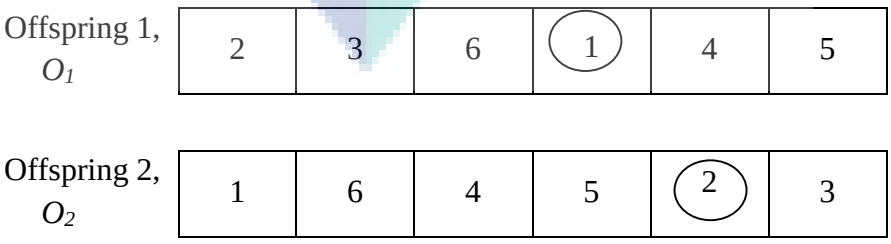
For rank selection, the individuals are sorted according to the fitness value before ranks them. Best individual gets rank 'N' and the worst one gets rank '1'. With respect to its rank, every chromosome is then allotted with the selection probability (Kumar & Member, 2012). Rank selection prevents premature convergence by uniform method of scaling across the population.

iv. Crossover

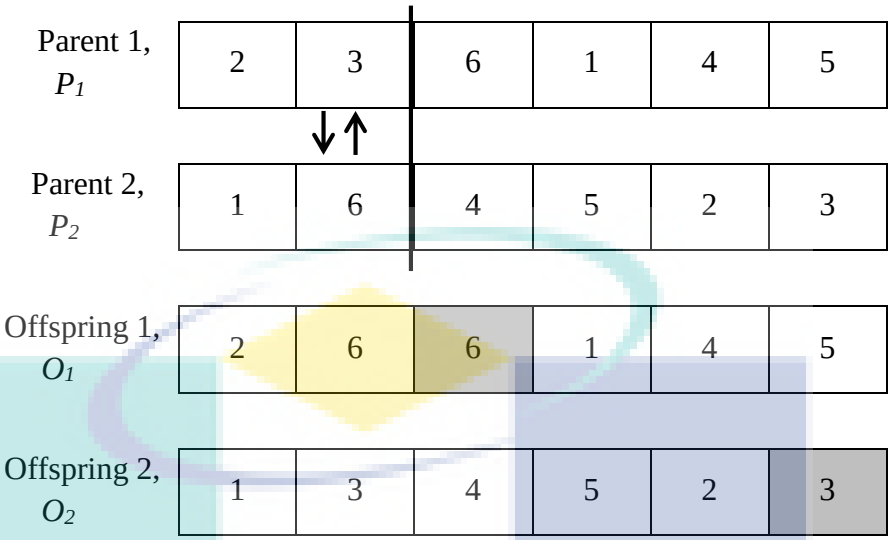
Crossover operator is used to produce two new offspring individuals (children) from two existing individuals (parents). The objective of crossover is to move the good gene from parent's chromosome to children. In SALBP, one point crossover is usually being implemented. In this type of crossover, two contiguous strings are combined by swapping the task after the crossover point with other string. The crossover must be done so that no repetition of the same task in a single string besides satisfying the precedence constraint (Barathwaj et al., 2015). For example, given Parent 1, P_1 {1, 3, 6, 2, 4, 5} and Parent 2, P_2 {2, 6, 4, 5, 1, 3}. On this two parent chromosomes, a crossover point is randomly selected after the second gene. Then, the crossover will start to swap the string at the first gene position.



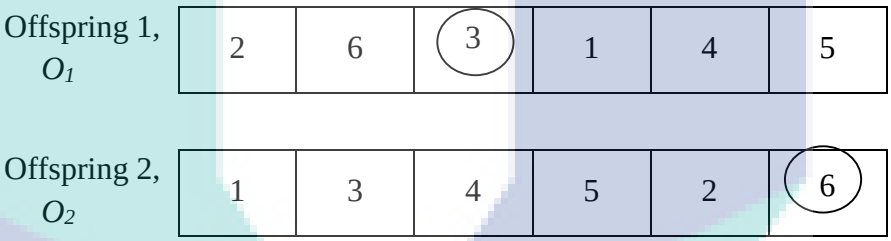
The repeated task in the string is then will be repaired.



The string values in second gene positions are swapped and repaired in similar way.



After repaired, the final offspring is



Three crossover operators Partially Matched Crossover (PMX), Ordered Crossover (OX) and Cycle Crossover (CX) are very suitable for the combinatorial optimisation problems (Yu & Yin, 2009). PMX produce new offspring from two parents by means of following procedure: (a) two random cut points is chosen for both parents. The strings of the cut point have the functions as mapped segments to be exchanged to produce new offspring, (b) two segments of the parents is exchanged, (c) second offspring is produce with the same procedure from the second parents (Chica, Cordón, & Damas, 2011).

In OX crossover, (a) two random cut point are selected. (b) gene inside the cut point remain unchanged while the gene outside the cut point is permuted. (c) The end

section of each of the two children is generated by filling in the missing elements in the order that they appear in the other parent without duplicating any of the elements within the children's chromosomes (Tasan & Tunali, 2008).

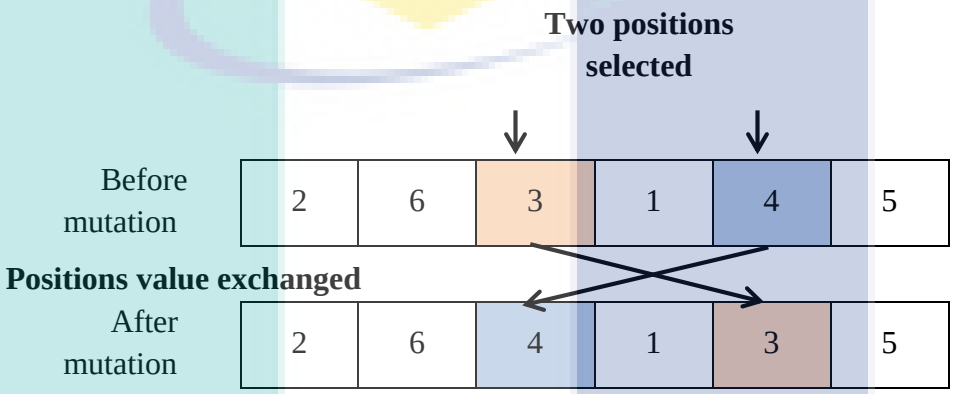
For Moon crossover, it is very similar to the change of the moon such as waxing moon , halfmoon, gibbous and full moon. The procedure of the moon crossover operator is described in figure below.

Procedure: Moon crossover
Begin
Initialization: $osp \leftarrow$ null, $k \leftarrow 0$
Select two random chromosomes p_a and p_b, $p_a = g_1 g_2 \ldots g_J$ and $p_b = q_1 q_2 \ldots q_J$;
Select two genes from p_a at random.
$osp \leftarrow$ the substring between g_i and g_j selected from p_a.
if the length of $osp = J$ **then end**
else $sub\_p_b \leftarrow$ the remaining substring from the deleting genes which are
already selected from p_a
end if
while (length of $osp \neq J$) **do**
if $i = 1$ **then** $i = J+1$;
$i \leftarrow i - 1$;
$k \leftarrow k + 1$, $k = 1, 2, \ldots,$ length of sub_p_b;
if $g_i \neq q_k$, then $osp = <osp, g_i, q_k>$;
else $g_i = q_k$, then $osp = <osp, g_i,>$;
else if $j = J$ **then**
$i \leftarrow i - 1$;
$k \leftarrow k + 1$, $k = 1, 2, \ldots,$ length of sub_p_b;
if $g_i \neq q_k$, then $osp = < q_k, g_i, osp>$;
else $g_i = q_k$, then $osp = < g_i, osp>$;
else
$i \leftarrow i - 1$;
$k \leftarrow k + 1$, $k = 1, 2, \ldots,$ length of sub_p_b;
if $g_i \neq q_k$, then $osp = < g_i, osp, q_k>$;
else $g_i = q_k$, then $osp = < g_i, osp>$;
end if
end while
end procedure

Figure 2.12 Procedure of Moon crossover (Adopted from Moon et al., (2009))

v. Mutation

Mutations make random changes to the offspring generated by the crossover and disturb the original genetic information (Simaria & Vilarinho, 2004). The purpose of mutation is to ensure diversity among individuals and therefore introduce randomness to the search. This will prevent the solution keep falling into a local optimum and the search space of the algorithm is diversify into uncovered search space (Mutlu et al., 2013a ; Y. Zhang et al., 2018).



vi. Termination

The process of crossover and mutation to forms a new population will be continue until termination criteria is met. A fixed number of generations, time, fitness and some form of convergence are typical termination criteria (Tasan & Tunali, 2008). During solution convergence, if there is no substantial change in fitness function after several iterations, then the algorithm can be terminated.

2.3.1.2 Genetic Algorithm for ALB

ALBP fall into NP-hard categories due to complexity of the combinatorial optimisation problems. Researchers previously have studied the ALBP and solved the problem by using optimum seeking methods such as Genetic Algorithms (GA) since it provides an alternative to traditional optimisation methods. GA has two advantages in solving ALB problem: (i) GA searches a solution in a large population rather than a single point and the algorithm will not be trapped in local optimum since many solutions are

considered concurrently, and (ii) GA fitness function may take any form and several fitness functions can be utilised simultaneously (Tasan & Tunali, 2008).

There are many research that implement genetic algorithm technique to the assembly line balancing problem since it able to locate optimum solutions in complex landscapes. Rajakumar et al. (2006) proposed a methodology based on GA to solve the parallel machine scheduling problems in an assembly line with precedence constraints with the objective of minimizing the workflow among the operators. The result shows that GA are able to search for better solution. However, the result obtained by this research is not validated by comparing the results with other meta-heuristic algorithms to find out the robustness of the results obtained by the GA.

Moon et al. (2009) address the problem of assigning multi-functional workers to tasks in assembly line while minimising the overall costs by using a mathematical model and genetic algorithm as an integrated optimisation. The results shows that GA are able to found optimal solutions for the small and medium-sized test problems more rapidly than the mathematical programming. However, the performance of GA in this research is not being validated with other types of meta-heuristic algorithms.

Purnomo et al. (2013) used a GA to solve two-sided assembly line balancing problem type 2 with assignment restriction. The assignment restriction that is considered in this paper is synchronous, distance constraint and zoning constraint. The result of GA is compared with iterative first-fit rules where the results shows that GA performs better for small and big benchmark problems, while the iterative first-fit rules perform better for medium problems.

Triki et al. (2014) studied on assembly line resource assignment and balancing problem of type 2. A new version of multi-objective genetic algorithm (MOGA) called hybrid MOGA (HMOGA) is introduced to solve a set of literature problems by minimizing the cycle time and cost of resources. The numerical results show that the HMOGA had a superior performance in comparison with Modified Weighted Pareto-Based Multi-Objective Genetic Algorithm (MWPMOGA) and Strength Pareto Evolutionary Algorithm 2 (SPEA2).

Alavidoost et al. (2015) proposed hybrid multi-objective genetic algorithm for single model straight and U-shaped assembly line balancing problems with fuzzy task

processing times. The algorithm is then tested on different benchmarks problems. The results of this research shows better performance for the proposed algorithm and the algorithm are able to find the optimal solution for the set objective functions.

GA have a nature of dynamic adaptive calculation. However, in standard GA there are high probabilities of having the non-optimal solution and trapped in local optima due to static circumstances in the evolution process (crossover and mutation). A twist in crossover can produce a larger solution space and reduce the possibility of stopping the searching at the non-optimal solution (Yu & Yin, 2009) . Sivasankaran & Shahabudeen, (2014) design four different genetic algorithm (GA)-based heuristics with different crossover methods to group the tasks in the single model assembly line balancing problem. The objective function of this research is to minimise the number of workstations for a given cycle time so that the efficiency of the line is maximised. The experimental results show that the proposed crossover method in GA surpassed the performance of the existing heuristics and the standard GA. However, these research were not considering the resource constraint which may occur in assembly line.

2.3.2 Ant Colony Optimisation (ACO)

Ant Colony Optimisation (ACO) is one of metaheuristic approach that is inspired from a natural process of the capability of ant in finding the good and shortest path in the sub-colony by using pheromone trail in communicating with each other. ACO use this concept in finding the best feasible solution on weighted graph to the optimization problem according to the problem's objective function (Akpınar et al., 2013). According to Shuang et al. (2008) ACO can produce premature convergence where the unfit solution might be just a few iteration away from the best optimum solution.

2.3.3 Particle Swarm Optimisation (PSO)

Particle Swarm Optimisation (PSO) is inspired by a behaviour of animal swarm in searching for food such as birds flocking or fish schooling. The PSO algorithm is initialized with a population of random solutions where the bird/fish is representing the individual potential solution, the flock/school is the population, the area is the search

space and the food is representing the optimum solution for the problem (Chutima & Kid-Arn, 2013). The issue with PSO in applying to ALB problem is, this algorithm originally is designed for continuous problem, the solution obtained is in real value space while solution for ALB is in discrete integer space (Lv & Lu, 2010).

2.4 Discrete Event Simulation

There are several numbers of simulation tools for manufacturing process that can help in decision making in the industry. Simulation tools can be used to evaluate the changes and performance of current system or to determine the improved new system behaviour under different kind of settings. Of all simulation techniques that can be used as simulation tools, a discrete event simulation is a method to models the operation of a real life system as a discrete sequence of events in time (Sharma, 2015).

Witness is a discrete-event simulation software from Lanner Group that has an object-oriented modelling environment that is mostly used in simulation of actual status of assembly lines in manufacturing industry. Witness simulation can also be used to simulate the improvement opportunities through the simulation statistical analysis. The concept of Witness involves five common elements in manufacturing industry which is parts, machines, conveyors, buffers and labours. (P. Semanco and D. Marton, 2013) .

In Witness simulation, forklifts and machines in the simulation functions can read the status of the test manufacturing system during simulation. The status is:

- Idle: the object is inactive;
- Busy: the object is working;
- Blocked: the object is not able to manage the missions due to the high workload (Briano et al., 2010).

Jaffrey & Mohamed (2018) use Witness Simulation to design the production of a manufacturing company with the objective functions is to optimize workers constraint in order to increase the production rate and line efficiency of the assembly line. The problems of the current assembly line and the performance of the improved assembly line is identified by analysing through Witness Simulation. The analysed simulation output in

term of average busy, idle and blocked percentage data are able to provide enough information regarding the assembly line performance.

Jusop & Rashid (2017) conducted a case study on electronic manufacturing company on assembly line balancing problem type-E with the objective function is to minimise number of workstation by considering resource constraint. Witness simulation is used to simulate the assembly line regarding on the average busy, idle, block and number of output on the existing layout, after optimization and after validation. The findings from the simulation shows that Witness simulation are able to provide information that the result of optimization have improvement in achieving the objectives function compared to the actual layout.

Hamzas et al., (2017) did a case study on a two sided assembly motorcycle assembly plant. The overall performance of the existing assembly line including machine and worker is evaluated by using Witness Simulation. The performance of the current layout is important for planning on the future improvement on the assembly line and resources used in the assembly line. The results from the simulations shows that the simulation data is valid and can be used as reference for future planning on the improvement of the assembly line.

2.5 Summary

The issue of line balancing with the minimum or limited number of resources (machines and workers) has always been a serious problem in the industry (Ağpak et al., 2005). However, in the majority of the previous works, researchers make assumptions where any of assembly tasks can be processed or assembled in any workstations. This is certainly true for the product which only requires a common or simple tool to be assembled. However, when the complexity of a product increased, it requires a special tool, machine or highly skilled labour to assemble that particular component. Therefore, the limitation of resources will be another constraint for the industry. Besides that, the current research on ALB-RC optimisation is limited to one type of resource in simple assembly line balancing problem.

Since ALB-RC is categorised as NP-hard problem, a suitable and reliable algorithm to optimise this problem is crucial to ensure the selected algorithm can produce a high quality feasible solutions in reasonable computational time. In previous ALB research works, GA was proved to produce better solutions compared to other algorithms in reasonable computational time. However, GA has a limitation in term of exploitation the best chromosome during the reproduction process in crossover and the generated number may violate the precedence relation in assembly line. Therefore, a newly proposed crossover can be introduced to tackle this issue.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This research is conducted in three main phases as in Figure 3.1. In the first phase, ALBP-RC representation is developed to fulfil the optimisation requirement and enable the integration of objective functions for SALBP-1. The procedure to establish a model of ALBP-RC is demonstrated by using an assembly example and followed by establishing the evaluation procedure by using mathematical equation. This evaluation procedure is used to combine the objective functions for optimisation purpose.

The second phase of this research is the development of an algorithm in order to optimise ALBP-RC. The activities in this phase started with establishing the solution procedure in finding the best solution for ALBP-RC. Once the general solution procedure is established, the algorithm flow is construct by referring to the solution procedure. Next, the algorithm is coded into a computer program. In this activity, MATLAB software is used for the coding purpose. The algorithm in MATLAB code later is tested and verified using the benchmark problems and the result will be compared with other types of crossovers.

In the third phase of this research, an industrial case study is conducted with the purpose to validate the proposed methodology (including representation, modelling procedure and evaluation procedure) and also the optimisation algorithm. In this step, the related assembly data such as assembly task, precedence constraint, resources and assembly time is collected. The selected problem later is modelled using the approach that developed earlier. Next, the problem will be optimise using the proposed algorithm.

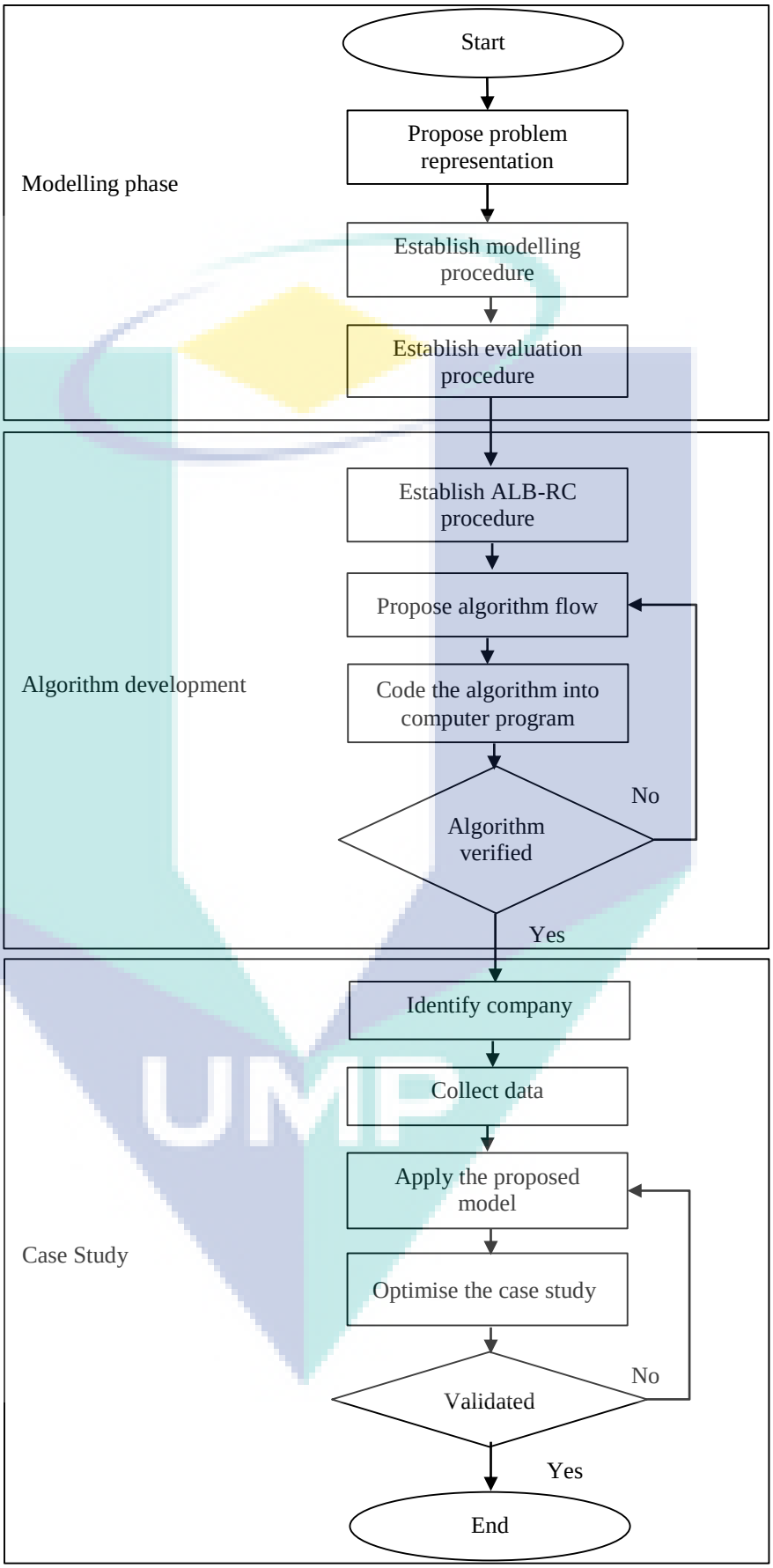


Figure 3.1 Research methodology

3.2 ALB with Resource Constraints Problem Modelling

The purpose of problem modelling is to integrate ALB-RC problems using a single representation scheme. This representation scheme will consider three optimisation objectives which is to minimise number of workstation, tool/machine and worker. The proposed approach will be developed based on assembly task and combine the precedence graph and matrices. An integrated representation scheme for ALB-RC enable all objectives function to be optimised together.

3.2.1 Mathematical equation

Firstly, the optimal solutions of the first, second and third objectives are obtained by solving each objective function separately using mathematical equation. The first objective function which is (1) to minimise the number of workstations for a given cycle time is subjected to (i) Assignment constraint, which ensures that each task is assigned only once; (ii) cycle time constraint, which ensure that the total times in each workstation does not exceed the given cycle time; (iii) precedence relation constraint, which guarantees that precedence relation among tasks is not violated; and (iv) workstation constraint, which guarantees that a workstation is utilised if the task(s) is/are assigned to it.

$$f_1 = \min \sum_{s=1}^{S} A_s \qquad 3.1$$

The second objective function is (2) to minimise the number of machines used which subject to resource availability constraints, which ensure that the total number of resources in the workstation is not more than the number of available machines.

$$f_2 = \sum_{m=1}^{r} y_{ms} \qquad 3.2$$

The third objective function is (3) to minimise the total number of workers used in an assembly line with the restriction that only one worker to be assigned to exactly one workstation depending upon his/her skills.

$$f_3 = \sum_{w=1}^{h} z_{ws} \tag{3.3}$$

As a final point, a weighted sum approach is used to combine the objective functions for optimisation purpose. However, due to different unit and ranges for objective functions, its need to be normalised into a similar range. For this purpose, the following formula is used:

$$\bar{f}_i = \frac{(f_i - f_{i\,min})(\bar{f}_{i\,max} - \bar{f}_{i\,min})}{(f_{i\,max} - f_{i\,min})} + \bar{f}_{i\,min} \tag{3.4}$$

where,

$\bar{f}_i$ - normalised value for *i*th objective function

f_i - value for *i*th objective function

$f_{i\,min}$ - minimum value for *i*th objective function

$f_{i\,max}$ - maximum value for *i*th objective function

$\bar{f}_{i\,max}$ - normalised maximum value

$\bar{f}_{i\,min}$ - normalised minimum value.

All the objective functions are normalised into 0 to 10 range $\bar{f}_i \in (1,10)$.

The minimum and maximum value of each objective functions are calculated as follow:

$$f_1 \max = round\ up\ \left(\frac{\sum_{i=1}^{n} t_i}{t_i max}\right) \tag{3.5}$$

$$f_1 \min = round\ up\ \left(\frac{\sum_{i=1}^{n} t_i}{ct_{max}}\right) \tag{3.6}$$

$$f_2 \max = f_1 \max \times no.of\ machine\ type \tag{3.7}$$

$$f_2 \min = no.of\ machine\ type \quad (3.8)$$

$$f_3 \max = n \quad (3.9)$$

$$f_3 \min = f_1\ min \quad (3.10)$$

Then, a fitness function is employed to minimise the summation of normalised objective functions.

$$F = w_1\bar{f}_1 + w_2\bar{f}_2 + w_3\bar{f}_3 \quad (3.11)$$

Where;

$$w_1 = w_2 = w_3 = 0.33$$

w_i represent the weights of objectives and $\bar{f}_1$, $\bar{f}_2$ and $\bar{f}_3$ represent respectively the normalised values derived from the equation 3.4.

3.2.2 Data presentation

The ALB problem is represented using a precedence graph. The number inside the node represents the assembly task. The directed edge means the precedence between task i and j. It shows that a task can be performed only after its predecessor tasks are performed. The precedence relation between each task needs to be examining the alternative routes from one node to another. In ALB, the assembly tasks need to be assigned into workstations, so that the workstation time is almost equal. Figure 3.2 shows a precedence diagram that represents an assembly process.

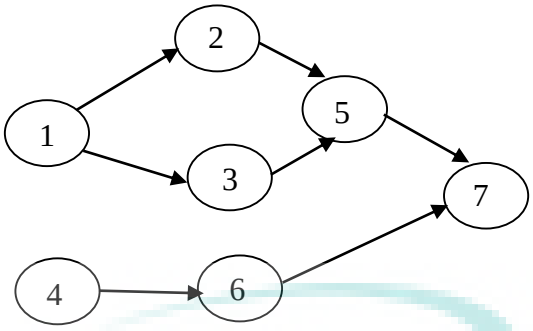


Figure 3.2 Example of precedence diagram

Table 3.1 presents the precedence relation for the above precedence diagram in a matrix form. In this matrix, when task *i* have precedence relation with task *j*, '1' will be put into the matrix, otherwise it will be '0'. This matrix presentation is important in transmitting the precedence relation between task into a computational language for optimisation or evaluation purpose.

Table 3.1 Precedence matrix

i	***j***						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	*0*	*1*	*1*	*0*	*0*	*0*	*0*
2	0	0	0	0	1	0	0
3	0	0	0	0	1	0	0
4	0	0	0	0	0	1	0
5	0	0	0	0	0	0	1
6	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0

Table 3.2 presents the ability of worker to perform task in a matrix form. Refer to researchers which conduct research in workers constraint in ALB (I. Moon et al., 2009; Kara, Özgüven, Yalçin, & Atasagun, 2011; Mutlu et al., 2013; Jayaswal & Agarwal, 2014), this researchers makes assumption on the capability of workers in performing the tasks given. In this matrix, when worker *h* have the ability to perform task *k*, '1' will be put into the matrix, otherwise it will be '0'.

Table 3.2 Worker matrix

k	*h*						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	*1*	*0*	*1*	*1*	*0*	*1*	*1*
2	0	1	0	1	0	1	0
3	1	1	1	0	1	0	1
4	0	0	1	0	1	1	1
5	1	0	0	1	0	1	0
6	0	1	1	0	1	0	0
7	1	1	0	1	0	0	1

Table 3.3 meanwhile shows the assembly information which includes the task time, machine and also worker. The machine type information are also based on the assumption made. The worker columns with tick mark meaning that the worker is able to conduct a specific assembly task. To assemble an assembly task, only one worker is required.

Table 3.3 Assembly information

Task	**Precedence relation**	**Time**	**Machine**	**Worker**						
				1	**2**	**3**	**4**	**5**	**6**	**7**
1	-	18	A	/		/	/		/	/
2	1	22	B		/		/		/	
3	1	9	B	/	/	/		/		/
4	-	7	A			/		/	/	/
5	2, 3	12	A	/			/		/	
6	4	6	B		/	/		/		
7	5, 6	20	A	/	/		/			/

3.2.3 Problem description

SALBP-1 is a simple assembly line with a number of tasks are carried in the designated workstation with given cycle time and have the objective to minimize number of workstation. The task is assigned to the workstation without violating the maximum cycle time on a serial line layout as illustrated in Figure 3.3.

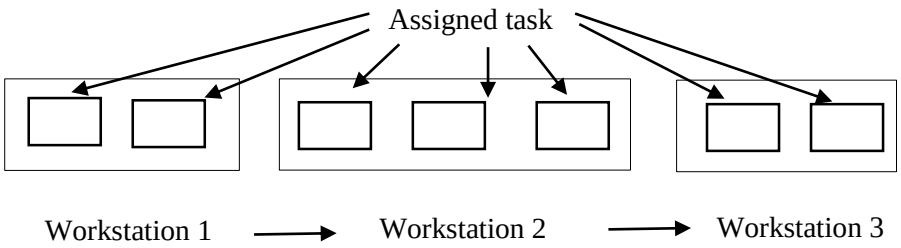


Figure 3.3 Simple assembly line

The problem of SALBP-1 with resource constraint is formulated by using several assumptions as follow:

a) The precedence relationships are known.
b) Task can be assigned to workstation without violating the precedence relation.
c) A worker can be assigned to task depending upon skills.
d) A worker can only be assigned to one workstation.
e) Machine required to process one task may be more than one type.
f) Tasks using the same type of machine can share the machine.
g) The line is balanced for a single product.
h) The processing time is deterministic.

3.2.4 Problem Evaluation

For the assembly information in Table 3.3, there are a few random feasible sequence can be derived such as sequence s_1 = [1 4 3 2 6 5 7] and s_2 = [4 1 2 3 5 6 7]. In evaluating ALB-RC, the assembly information for each sequence which consist of the task time, types of machines needed and the worker that can conduct the task is gathered as in Table 3.4 and Table 3.5.

Table 3.4 Assembly information for feasible assembly sequence s_1

Sequence	1	4	3	2	6	5	7
Time	18	7	9	22	6	12	20
Tool	A	A	B	B	B	A	A
Worker	1,3,4,6,7	3,5,6,7	1,2,3,5,7	2,4,6	2,3,5	1,4,6	1,2,4,7

Table 3.5 Assembly information for feasible assembly sequence s_2

Sequence	4	1	2	3	5	6	7
Time	7	18	22	9	12	6	20
Tool	A	A	B	B	A	B	A
Worker	3,5,6,7	1,3,4,6,7	2,4,6	1,2,3,5,7	1,4,6	2,3,5	1,2,4,7

For SALBP-1, the maximum cycle time, ct_{max} are given. For each workstation, the total processing time cannot exceed the ct_{max} or otherwise, the demand for the product cannot be fulfilled. The example of the assignment of assembly task to workstation for s_1 with the given maximum cycle time, ct_{max} is 34 time unit is presented in Figure 3.4.

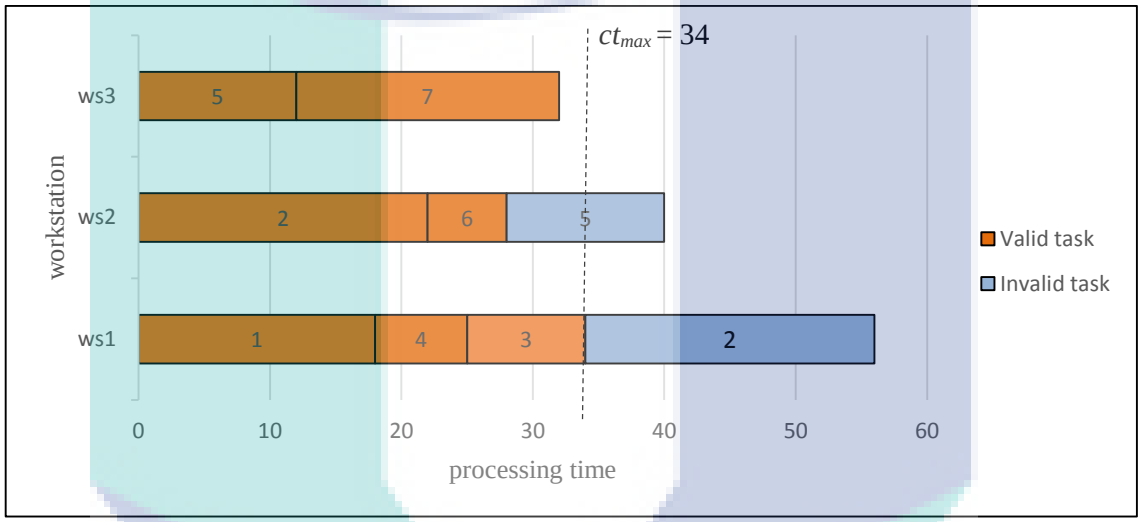


Figure 3.4 Assembly task assignment for s_1

Based on Figure 3.4, for workstation 1 (ws_1) in s_1, the total assembly time for 1, 4, and 3 is 34 time units. If the assembly task 2 is also included in ws_1, the total assembly time will become 56 time units, which exceeds the ct_{max}. Therefore, the assembly task 2 is assigned into ws_2. This same procedure is also applied to the subsequent tasks and workstations. The results of assembly tasks assignment for s_1 and s_2 are presented in Table 3.6 and Table 3.7.

The station time row shows cumulative time to conduct assembly process for all tasks in a specific station. The types of machine required in conducting task in every workstation is represented in machine row. Meanwhile, the worker selection is made based on the number of workers frequency in a workstation. For example, assembly information in Table 3.4 shows that in workstation 1, workers 3 and 7 have the highest frequency in conducting any task given. In this case, the worker is select randomly.

Table 3.6 Assembly task and workstation assignment s_1

Workstation	1	2	3
Task	1, 4, 3	2, 6	5, 7
Station time	34	28	32
Machine	A,B	B	A
Worker	3	2	1

Table 3.7 Assembly task and workstation assignment s_2

Workstation	1	2	3	4
Task	4, 1	2, 3	5, 6	7
Station time	34	31	18	20
Machine	A	B	A, B	A
Worker	6	2	1, 3	4

Based on the presented approach, the fitness value for the stated three objective functions in ALB-RC problem can be calculated as below:

- Maximum and minimum value for f_1, f_2 and f_3 is calculated from equation 3.5 - 3.10.

$$f_1 \max = \frac{\sum t_i}{t_{max}} = \frac{94}{22} = 4.27 \approx 5 \text{ stations}$$

$$f_1 \min = \frac{\sum t_i}{ct_{max}} = \frac{94}{34} = 2.76 \approx 3 \text{ stations}$$

$$f_2 \max = f_1 \max \times no.of\ machine\ type = 5 \times 2 = 10\ machines$$

$$f_2 \min = no.of\ machine\ type = 2\ machines$$

$$f_3 \max = n = 7\ workers$$

$$f_3 \min = f_1 \min = 3\ workers$$

- Normalised value $\bar{f_i}$ for f_1, f_2 and f_3 can be calculated using equation 3.4 as follows.

Feasible sequence s_1

$$\bar{f}_i = \frac{(f_i - f_{i\,min})(\bar{f}_{i\,max} - \bar{f}_{i\,min})}{(f_{i\,max} - f_{i\,min})} + \bar{f}_{i\,min}$$

$$\bar{f}_1 = \frac{(3-3)(10-1)}{(5-3)} + 1 = 1$$

$$\bar{f}_2 = \frac{(4-2)(10-1)}{(10-2)} + 1 = 3.25$$

$$\bar{f}_3 = \frac{(3-3)(10-1)}{(7-3)} + 1 = 1$$

Feasible sequence s_2

$$\bar{f}_i = \frac{(f_i - f_{i\,min})(\bar{f}_{i\,max} - \bar{f}_{i\,min})}{(f_{i\,max} - f_{i\,min})} + \bar{f}_{i\,min}$$

$$\bar{f}_1 = \frac{(4-3)(10-1)}{(5-3)} + 1 = 5.5$$

$$\bar{f}_2 = \frac{(5-2)(10-1)}{(10-2)} + 1 = 4.375$$

$$\bar{f}_3 = \frac{(5-3)(10-1)}{(7-3)} + 1 = 5.5$$

- Fitness function for is calculated by using equation 3.11.

Feasible sequence s_1

$$F = w_1\bar{f}_1 + w_2\bar{f}_2 + w_3\bar{f}_3$$

$$F_1 = (0.33 \times 1) + (0.33 \times 3.25) + (0.33 \times 1) = 1.733$$

Feasible sequence s_2

$$F = w_1\bar{f}_1 + w_2\bar{f}_2 + w_3\bar{f}_3$$

$$F_2 = (0.33 \times 5.5) + (0.33 \times 4.375) + (0.33 \times 5.5) = 5.074$$

Therefore, the result for test problems as presented in Table 3.8.

Table 3.8 Result for test problem

Sequence	Workstation	Tool	Worker	Fitness value
s_1	**3**	**4**	**3**	**1.733**
s_2	4	5	5	5.074

Table 3.8 presents the optimisation objective value in term of number of workstation, tool and worker obtained by using different feasible sequence for test problem. A weighted sum approach is then used for combining those three objective functions into a scalar fitness solution. The fitness value is determined the best feasible sequence that will produce the best solution for the problem given. In Table 3.8, sequence s_1 produce fitness value 1.733 compared to sequence s_2 with fitness value 5.074. From literature review, a chromosome that has a minimum fitness function value is the best chromosome.

3.3 Genetic Algorithm with Rank Based Crossovers

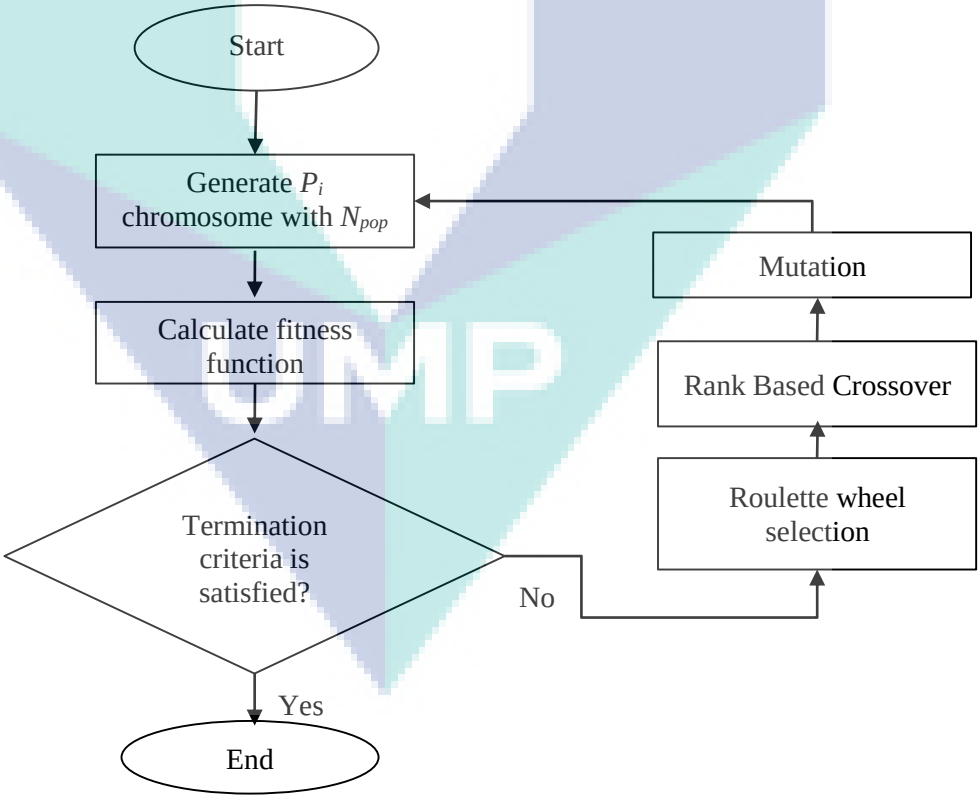


Figure 3.5 Flowchart of GA with Rank Based Crossovers

Genetic algorithm is an optimisation technique that mimics the survival for the fitness concept. Solutions with better fitness have larger possibilities to remain in the population, while the solution with bad fitness will be eliminated from the population. In general, GA consists of five main steps; Initialization, Evaluation, Selection, Crossover and Mutation.

Step 1: Initialization

In the initialization step, the GA parameter such as population size, n_{pop}, maximum generation, gen_{max} and GA coefficients are set up. For the chromosome representation, S, a set of permutation integer is used. In this case, the integer is directly representing the assembly task. For example, the first chromosome in the assembly problem with six tasks as in Figure 3.6 might be, S1 = [3 4 1 6 2 5]. Besides that, the initial population, pop that consist of n_{pop} of chromosomes are randomly generated. These sequences normally did not satisfy the precedence constraint.

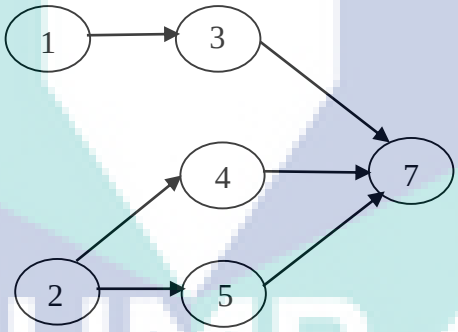


Figure 3.6 Example of ALBP-RC

Step 2: Decoding

Before the solution can be evaluated using the fitness function, the chromosome must be decoded into feasible solution, *f*. Feasible solution refers to the assembly sequence that fulfils the constraint. In ALB problem, the compulsory constraint is the precedence constraint. For the given example chromosome in the previous step, S_1 = [3 4 1 6 2 5], the sequence in the S_1 violate the precedence constraint. In this example, the assembly task 3 only can be started after task 1 is completed (Figure 3.6).

A topological sort is implemented to decode the solution. The topological sort is started by identifying the candidate tasks. The candidate tasks refer to the task/s without the precedence. For the example in Figure 3.6 the candidate tasks are task 1 and 2. Since the candidate tasks is more than one, a selection based on the earliest task appearance in the chromosome is applied. Referring to the S_1 = [3 4 1 6 2 5], task 1 appears earlier than task 2. Therefore, task 1 is selected and stored in feasible solution f_1 = [1]. Next, the selected task is removed from the precedence graph. So now, the new candidate tasks are task 2 and 3. In this case, task 3 appeared earlier than task 2 in S_1. Therefore, the feasible solution become f_1 = [1 3]. These steps are repeated until all the assembly tasks are selected and stored in f_1. For this example, the feasible solution decoded using this procedure is f_1 = [1 3 2 4 5 6].

Step 3: Evaluation

In this step, the encoded feasible sequence is evaluated by using predefined objective functions. The objective functions are calculated using procedures and formulas in Equation 3.4 – 3.11.

Step 4: Selection

The purpose of selection step is to choose the chromosome to be placed in the mating pool. The selected chromosome will be the parent of the children in a new generation. The selection process is conducted using a Roulette wheel selection (RWS) mechanism. In the RWS, the chromosome with better fitness will have a larger portion of the space on the Roulette wheel. Therefore, the solution with better fitness will have a larger chance to be chosen as the parent for a new generation. In the RWS, there is a possibility where the same solution is chosen more than once.

Step 5: Crossover

In the reproduction of children, crossover operator plays an important rule. In general, crossover works by manipulating two parent chromosomes to produce two new

children. In this work, we introduced two crossover operators, named Rank based crossover type I and II (RBC-I and RBC-II). The proposed crossovers are compared with popular crossover operator for permutation problem, i.e. ordered crossover (OX), partially matched crossover (PMX) and Moon crossover (C. Moon, Kim, Choi, & Seo, 2002).

Both of the proposed crossovers taken into account the best chromosome from the population in the reproduction process. In both of the proposed crossover, each of the assembly tasks will be given a rank according to their position in the chromosome. Then the rank for parent and best chromosome will be summed up to form a new rank. The child solution will be generated based on the new rank. By using this approach, the new child will inherit the gene from their parent and also the best solution. The detail explanation on the proposed crossover as in section 3.3.1 and 3.3.2.

Step 6: Mutation

Mutation is exploration mechanism in GA. The purpose of mutation is to explore the search space to obtain a better solution. Besides that, mutation is also function to avoid the solution from trapped in local optima. For this purpose, a single point swapped mechanism is used. For a child that produced from the crossover, a random cutting point is selected. Then, the chromosome is swapped the position between front and rear.

Step 7: Termination

Termination criteria determine how long the optimisation will run. There are a few termination criteria used to determine the optimisation run. One of the termination criteria is stall generation. Stall generation means that the optimisation result did not converge for a particular number of generations. When the generation is stall, the optimisation run will automatically stop. In this work, we used the maximum number of generation to stop the optimisation.

3.3.1 Rank based crossover type-I (RBC-I)

In RBC-I, one parent (P_1) is mated with the best chromosome (X_{best}) to produce one new offspring. Firstly, gene in P_1 and X_{best} is ranked according to the position in the string. Next, P_1 and X_{best} is accendingly sorted to produce new rank for sorted rank P_1 (R'_1) and sorted rank X_{best} (R'_{best}). Then, R'_1 and the R'_{best} is added to form sorted offspring rank (R'_{O1}). Finally, the P'_1 is sorted back according to the R'_{O1} to generate offspring solution, O_1. In the case where the rank is tied, the selection is made randomly. This offspring solution will be used for mutation in the next steps of the process in GA to find the optimal solution. The numerical procedure for RBC-I is presented in Figure 3.7.

Step 1: Assign rank to the P_1 and X_{best}

P_1	4	2	5	3	1
R_1	1	2	3	4	5

X_{best}	2	3	1	4	5
R_{best}	1	2	3	4	5

Step 2: Sort rank according to P'_1 and X'_{best}

P'_1	1	2	3	4	5
R'_1	5	2	4	1	3

X'_{best}	1	2	3	4	5
R'_{best}	3	1	2	4	5

Step 3: Sum up R'_1 and R'_{best} as R'_{O1}

P'_1	1	2	3	4	5
R'_1	5	2	4	1	3
R'_{best}	3	1	2	4	5
R'_{O1}	8	3	6	5	8

Step 4: Sort P_1 according to the sum rank to generate O_1

O_1	2	4	3	1	1
R_{O1}	3	5	6	8	8

Figure 3.7 Numerical procedure for RBC-I

A pseudo-code of the RBC-I is given in Procedure for RBC-I below.

Procedure for RBC-I

Begin
Identify the best chromosome from Evaluation, X_{best}
Assign rank, R_{best} to X_{best}
Sort X_{best} in ascending order, X'_{best}
Sort R_{best} according to X'_{best}, R'_{best}
Do
For each parent, P_i
Assign rank, R_i to P_i
Sort P_i in ascending order, P'_i
Sort R_i according to P'_i, R'_i
Calculate sorted offspring rank, R'_{Oi}
$R'_{Oi} = R'_i + R'_{best}$
New offspring, O_i → Sort P'_i according to R'_{Oi} in ascending order
End

3.3.2 Rank based crossover type-II (RBC-II)

The RBC-II applied the same rank concept as in RBC-I, but this crossover considers two parents to mate with one best chromosome. The early steps where the rank is assigned and sorted is the same with RBC-I as shown in Figure 3.8. However, to calculate the rank for offspring solutions (R'_O), the following formula is used in RBC-II.

$$R'_O = C_{best}(R'_{best}) + C_1(R'_1) + C_2(R'_2) \quad 3.12$$

C_{best}, C_1 and C_2 are the coefficients for the best chromosome (X_{best}), parent 1 (P_1) and parent 2 (P_2) respectively. The C_{best} is fixed at 0.2 where the offspring solution will only inherit 20% from the best chromosome. Meanwhile, the C_1 and C_2 coefficient is depend on the offspring. To generate offspring 1 (O_1), the C_1 and C_2 are as follow.

$$C_1 = 0.7(1 - C_{best}) \quad 3.13$$

$$C_2 = 0.3(1 - C_{best}) \quad 3.14$$

On the other hand, to generate offspring 2 (O_2), the following coefficients are used.

$$C_1 = 0.3(1 - C_{best}) \quad 3.15$$

$$C_2 = 0.7(1 - C_{best}) \quad 3.16$$

Next, after the rank for offspring is calculated, the offspring solutions (O_1 and O_2) are generated by sorting the rank for offspring (R'_O) in the ascending orders. As in RBC-I, in the event of tie rank, the selection is made randomly. The numerical example for RBC-II is shown in Figure 3.8.

A pseudo-code of the RBC-II is given in Procedure for RBC-II below.

Procedure for RBC-II

Begin

Identify the best chromosome from Evaluation, X_{best}

Assign rank, R_{best} to X_{best}

Sort X_{best} in ascending order, X'_{best}

Sort R_{best} according to X'_{best}, R'_{best}

Do

For a pair of parent, P_i and P_j

Assign rank, R_i to P_i and R_j to P_j

Sort P_i and P_j in ascending order, P'_i, P'_j

Sort R_i according to P'_i, R'_i

Sort R_j according to P'_j, R'_j

Calculate sorted offspring rank, R'_{Oi}

For coefficient, Cbest = 0.2

$R'_{Oi} = C_{best}(R'_{best}) + C_1(R'_i) + C_2(R'_j)$

$C_1 = 0.7(1 - C_{best})$

$C_2 = 0.3(1 - C_{best})$

$R'_{Oj} = C_{best}(R'_{best}) + C_1(R'_i) + C_2(R'_j)$

$C_1 = 0.3(1 - C_{best})$

$C_2 = 0.7(1 - C_{best})$

Generate new offsprings,

$O_i \rightarrow$ Sort P'_i according to R'_{Oi} in ascending order

$O_j \rightarrow$ Sort P'_j according to R'_{Oj} in ascending order

End

Step 1: Assign rank to the P_1 and X_{best}

P_1	4	2	5	3	1
R_1	1	2	3	4	5

P_2	1	5	2	4	3
R_2	1	2	3	4	5

X_{best}	2	3	1	4	5
R_{best}	1	2	3	4	5

Step 2: Sort rank according to P_1 , P_2 and X_{best}

P'_1	1	2	3	4	5
R'_1	5	2	4	1	3

P'_2	1	2	3	4	5
R'_2	1	3	5	4	5

X'_{best}	1	2	3	4	5
R'_{best}	3	1	2	4	5

Step 3: Calculate rank for offspring

O'_1	1	2	3	4	5
$R'O_1$	3.64	2.04	3.84	2.32	3.16

O'_2	1	2	3	4	5
$R'O_2$	2.36	2.36	4.16	3.28	2.48

Step 4: Sort O' according to RO to generate offspring

O_1	2	4	5	1	3
R_{O1}	2.04	2.32	3.6	3.64	3.84

O_2	1	2	5	4	3
R_{O2}	2.36	2.36	2.48	3.28	4.16

Figure 3.8 Numerical example of RBC-II

3.4 Computational Experiment

A computational experiment was carried out in order to evaluate the effectiveness and the performance of RBC-I and RBC-II on the problem of SALBP-1 with tool and worker constraints. For this purpose, a set of ALB benchmark problem by Scholl is used (Scholl, 1993). This benchmark data sets have been used for testing and comparing solution procedures in SALBP by Scholl et al. (2008), Taylor et al. (2012), Saif et al. (2014) and Sikora et al. (2016). The instances in the SALBP-1 benchmark problem contain a wide range of values of the cycle time (from 6 to 17,067 units of time) and number of tasks (from 7 to 297 tasks).

- From 47 instances for SALBP-1 in the benchmark problem, 17 problems that varies in term of the size is selected. The benchmark test problem is divided into three categories; small ($n \leq 20$ task), medium ($20 < n \leq 70$) and large ($n > 70$).
- The mathematical programming language for the problems of SALBP-1 with tools and workers constraint is coded into MATLAB Version 7.0 with a HP Intel Core i5 at 2.5 GHz and with 4 GB of RAM.
- For the computational experiment, the population size is set to 30, maximum generation is 300, probability of crossover is 0.7 and probability of mutation is 0.2.

In order to reduce pseudorandom effect in finding the optimal solution for the problem, the optimisation is run for five times. The best solution will be selected for comparison purpose. For comparison purpose, the RBC-I and RBC-II are compared with popular crossover operators for the combinatorial problem. The comparison crossovers are the ordered crossover (OX), partially matched crossover (PMX) and Moon crossover. The OX and PMX are among popular crossover operator for the combinatorial problem. Meanwhile, the Moon crossover is used since it was claimed to be the best crossover for the combinatorial problem (C. Moon et al., 2002).

3.5 Case Study

An industrial case study has been conducted in an electronics manufacturing company. The purpose of the industrial case study is to evaluate the proposed ALB-RC model and the GA using RBC to optimise the problem. The industrial case study started with understanding the assembly process for the studied product. Then, the assembly task for the studied product is identified. In this case, the assembly tasks are directly defined based on the work elements used by the company.

Next, the precedence constraints are defined according to the engineer's input and the assembly process observation. Then, the assembly data collection is made. For the assembly task time, five repetitions are made and the average time is calculated. Besides the assembly time, the main machine or equipment used to conduct the assembly task is also recorded. In addition, the details of worker skills and requirements are gathered. The precedence graph of the studied product is shown in Figure 3.9.

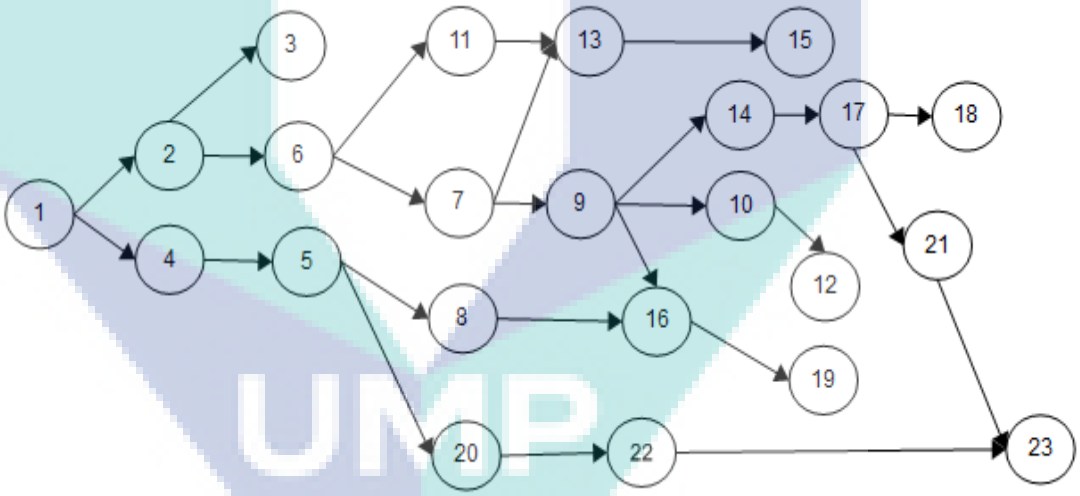


Figure 3.9 Precedence diagram for the case study

The assembly data for this problem is presented in Table 3.9. For the optimisation purpose, the average time will be used. Table 3.9 also presents the required machine or tool for the equipment. Since the proposed model allows one machine per assembly task, only the main machine or equipment is considered. It should be noted that some of the assembly tasks requires more than one tools to be completed. The last column in Table 3.9 shows the worker that needs to conduct a particular assembly task. Currently, 10

workers are needed to perform the assembly process. According to the line supervisor, the workers can conduct any assembly task with minimum training required. Therefore, it is assumed that any assembly task can be assigned to any worker.

Table 3.9 Assembly data for case study problem

Task	Task time (s)						Machine	Worker
	1	2	3	4	5	Average		
1	35.2	37.3	38.1	40.4	43.1	38.82	M1	W1
2	27.6	34.5	33.7	30.2	29.7	31.14	M1	W1
3	43.9	50.2	51.8	47.7	46.2	47.96	M2	W1
4	84.2	92.1	91.4	90.7	85.9	88.86	M2	W2
5	29.1	27.7	25.2	25.6	29.0	27.32	M3	W2
6	42.8	46.7	53.2	51.7	44.3	47.74	M2	W3
7	78.8	76.4	66.8	69.7	74.0	73.14	M3	W3
8	20.5	16.9	24.9	23.4	18.8	20.90	M2	W4
9	64.3	63.9	62.1	62.5	68.7	64.30	M3	W4
10	14.2	12.3	10.6	10.7	11.6	11.88	M4	W4
11	14.6	17.4	13.6	14.3	15.8	15.14	M4	W4
12	69.6	61.1	57.7	67.1	68.4	64.78	M5	W5
13	63.3	55.5	58.9	55.1	56.8	57.92	M6	W5
14	62.5	63.6	55.2	56.4	61.7	59.88	M5	W6
15	38.5	46.3	47.8	48.7	43.4	44.94	M5	W6
16	29.3	30.4	34.2	29.1	36.0	31.80	M7	W7
17	28.3	24.7	26.2	31.8	25.3	27.26	M6	W7
18	65.4	73.1	64.1	71.7	74.8	69.82	M6	W8
19	25.2	20.8	22.6	19.5	23.3	22.28	M7	W8
20	42.7	46.3	39.6	47.4	43.5	43.90	M4	W9
21	17.7	17.6	22.4	18.2	20.9	19.36	M8	W9
22	16.8	20.1	24.3	15.5	23.7	20.08	M7	W9
23	36.5	36.5	34.8	29.1	32.2	33.82	M8	W10

The efficiency of the assembly line configuration can be measured using the objective function as explained in section 3.2.3. In addition to the optimisation objective function, the following indicators are used to measure the efficiency of the assembly line.

Smoothness index (*SI*):

$$SI = \sqrt{\sum_{k=1}^{K} (ct - pt_k)^2} \quad 3.17$$

Line efficiency (*LE*):

$$LE = \frac{\sum_{i=1}^{n} t_i}{nws \times ct} \quad 3.18$$

SI measure how balance the workload assignment between workstation. The smaller SI represent better workload balance. This will smoothen the flow of the assembled product in the assembly line. Meanwhile, the LE shows the level of value added time utilization in assembly line. The higher LE indicated the lower wasting time in the assembly line.

3.6 Witness Simulation

A discrete event simulation has been conducted for an industrial case study problem. Simulation is a tool to mimic a real-world situation, without disturbing the physical system. In assembly line, simulation is also a tool to validate the changes or improvement achieved by a proposed solution, without changing the real assembly process. In this research, the purpose of discrete event simulation is to measure the changes in the assembly line when the assembly task configuration is changed. To be more specific, this simulation will measure the assembly line before and after the optimisation using the proposed crossovers. This activity is also a method to validate the ALB-RC model and the proposed assembly task configuration by new crossovers.

For simulation purpose, Witness simulation software by Lanner Group is used. In the Witness simulation software, the Machine element is defined as a workstation for the assembly line. The elements set up in Witness software is presented in Figure 3.10.

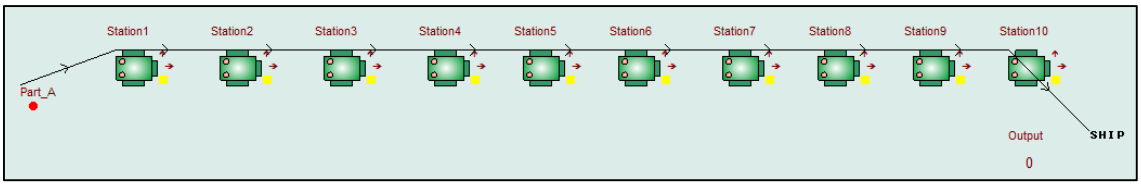


Figure 3.10 Witness basic elements setup

In the simulation mode, one part element is set to represent the product. The detailed setup for the part element is shown in Figure 3.11.

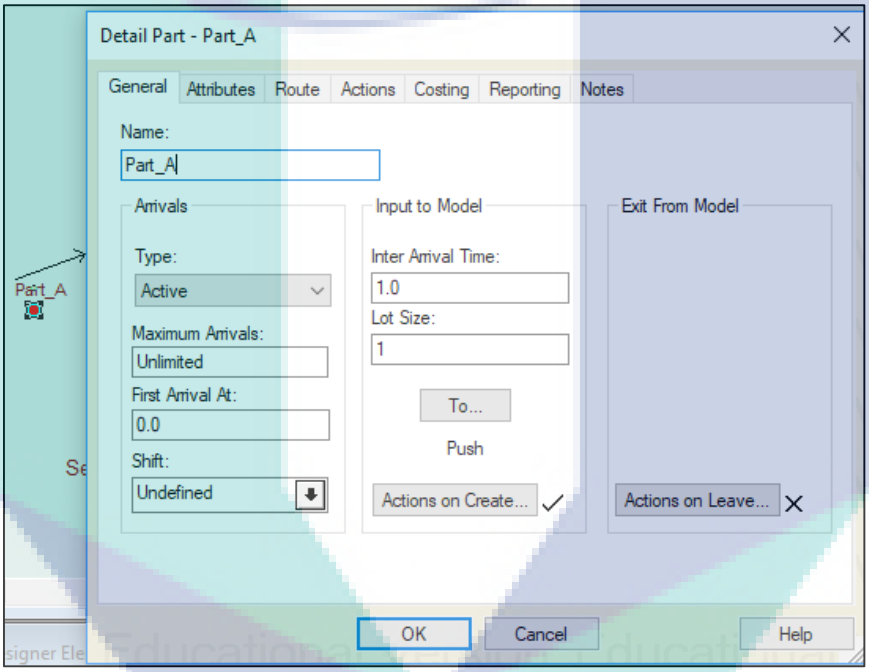


Figure 3.11 Detail element setup

For simulation purpose, the part arrival is assumed as follow:

Inter-arrival time = 1.0 second

Maximum arrival = Unlimited

First arrival = 0.0 second

In this model, the push system is used to transfer the part from one station to another station as in Figure 3.12.

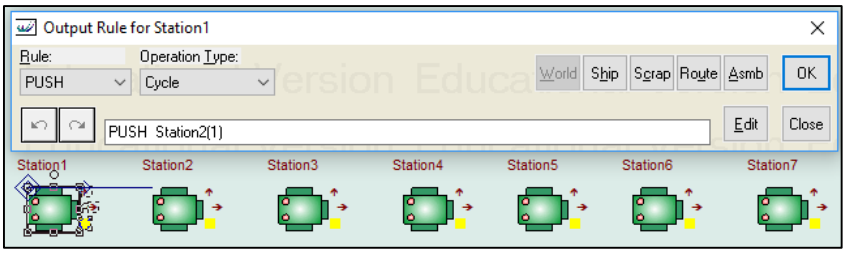


Figure 3.12 Witness push system setup

Meanwhile, the processing time for each workstation then is assumed as normally distributed. This time is defined from the cumulative task time in a particular workstation. In the Detail Machine element, the average time is used for the duration as shown in Figure 3.13. Besides that, the user also needs to specify the standard deviation and also the pseudo-random seed.

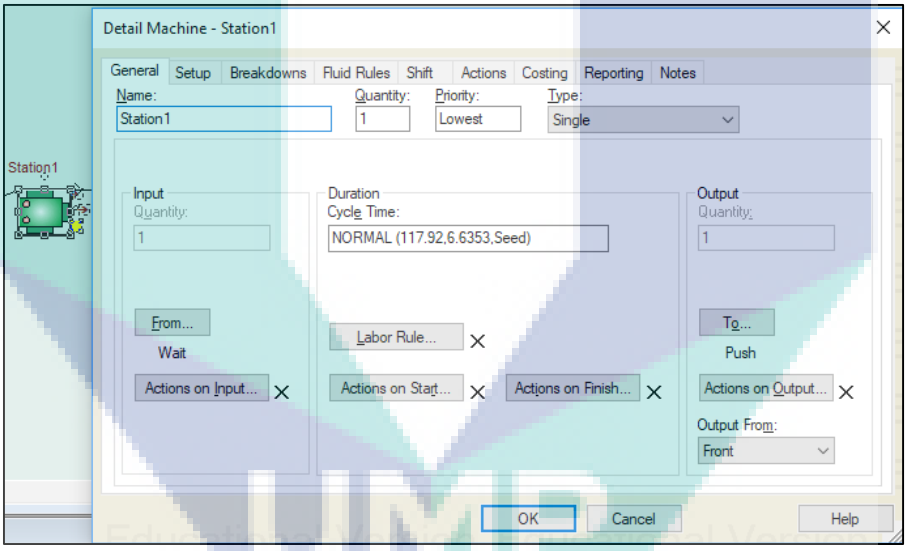


Figure 3.13 Detail machine element setup

In this model, the pseudo-random random seed is defined on the action on creating part element as in Figure 3.14.

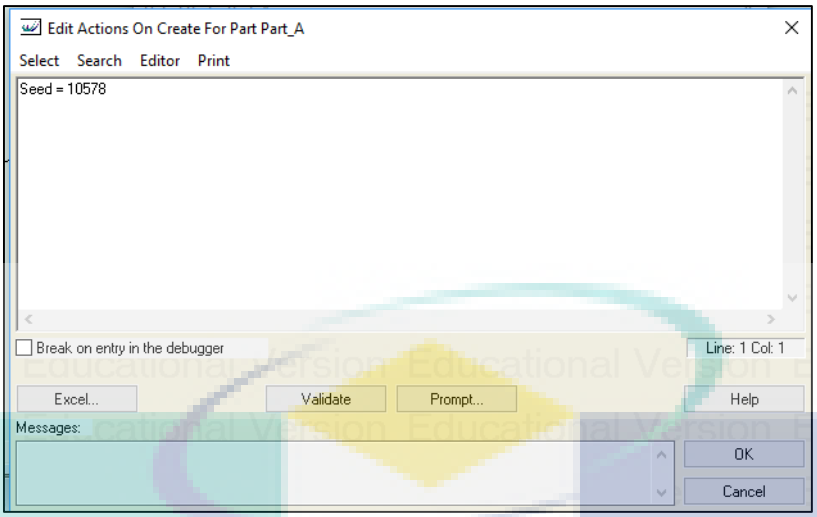


Figure 3.14 Pseudo-random seed control

The simulation is conducted for a period of eight working hours. For this purpose, the simulation duration is set at the simulation control panel, as shown in Figure 3.15.

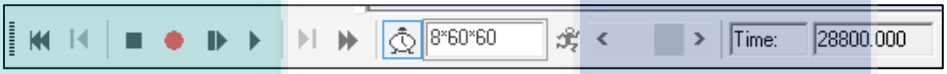


Figure 3.15 Simulation control panel

Simulation is repeated for ten times, with different pseudo-random seeds. In this simulation, four output are measured as follow:

i. Average idle percentage

ii. Average busy percentage

iii. Average block percentage

iv. Number of output per simulation duration

The example of Witness simulation output is presented in Figure 3.16.

Name	% Idle	% Busy	% Filling	% Emptying	% Blocked	% Cycle Wait Labor	% Setup	% Setup Wait Labor	% Broken	% Repair Wait	No. Of Operation
Station1	0.41	95.51	0.00	0.00	4.08	0.00	0.00	0.00	0.00	0.00	234
Station2	1.66	93.81	0.00	0.00	4.53	0.00	0.00	0.00	0.00	0.00	233
Station3	1.13	97.33	0.00	0.00	1.54	0.00	0.00	0.00	0.00	0.00	232
Station4	4.06	89.92	0.00	0.00	6.02	0.00	0.00	0.00	0.00	0.00	231
Station5	2.28	97.71	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	230
Station6	16.40	83.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	229
Station7	52.96	47.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	228
Station8	27.05	72.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	228
Station9	34.13	65.87	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	227
Station10	73.43	26.57	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	227

Name	Part_A
No. Entered	235
No. Shipped	227
No. Scrapped	0
No. Assemble	0
No. Rejected	28566
W.I.P.	8
Avg W.I.P.	7.86
Avg Time	963.87
Sigma Rating	6.00

Figure 3.16 Example of Witness simulation output

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Optimisation Results for Benchmark Problem

In the computational experiment for the benchmark problem, the optimal solution for the problem is depends on the fitness function. In this problem, the fitness function consists of the following optimisation objectives; (1) minimise number of workstation, (2) minimise number of machine, and (3) minimise number of worker. These optimisation objectives are then combined using a weighted sum approach as presented in section 3.2.

Table 4.1 presents the best fitness values (minimum fitness function) obtained by GA using different crossovers strategies tested on 17 benchmark problems range from 7 to 148 number of tasks. In Table 4.1, the bolded values show the best fitness for a particular problem. Based on the results, all the crossovers were able to search for an optimum solution for the small size problems (Problem 1 to 5). The fitness value shows the same value when tested using different crossovers.

However, when the problem size is increased from small to medium and large size problem, the RBC-I and RBC-II have shown a better performance compared with other crossovers. In medium-size problem (Problem 6 to 11), the RBC-II has better fitness in 83% (5 out of 6) of the benchmark problems, except in Hahn problem. Meanwhile, in large size problem (Problem 12 to 17), the RBC-I and II individually has better fitness in 50% of the problem. Other type of crossover are not able to obtain the best fitness values in large size benchmark problem.

Table 4.1 Optimisation results

No	Problem	No. of Task	Given Cycle Time	Crossover type				
				OX	PMX	Moon	RBC-I	RBC-II
1	Mertens	7	8	8.7500	8.7500	8.7500	8.7500	8.7500
2	Bowman	8	20	8.5000	8.5000	8.5000	8.5000	8.5000
3	Jaeschke	9	18	3.5000	3.5000	3.5000	3.5000	3.5000
4	Mansoor	11	48	2.9286	2.9286	2.9286	2.9286	2.9286
5	Jackson	11	13	2.2857	2.2857	2.2857	2.2857	2.2857
6	Buxey	29	54	4.0457	4.0576	4.0517	3.7789	3.5181
7	Sawyer	30	75	1.6800	1.6800	1.6800	1.8000	1.4400
8	Gunther	35	69	2.7952	2.8952	2.7810	2.6738	2.5881
9	Kilbridge	45	69	3.8831	3.9642	3.9599	3.8789	3.7999
10	Hahn	53	2004	5.6467	5.5190	5.5815	5.6495	5.6440
11	Warnecke	58	111	3.4024	3.5080	3.8168	3.5628	3.1858
12	Wee Mag	75	56	4.1857	4.1303	4.0446	4.0053	3.8964
13	Arc83	83	6540	2.4743	2.5198	2.5879	2.4320	2.5132
14	Lutz 2	89	19	3.0760	2.9237	3.0062	2.5626	2.9492
15	Mukherje	94	263	3.2866	3.3370	3.2747	3.4274	3.0903
16	Arc111	111	6540	6.7874	6.6003	6.5334	6.4993	5.5394
17	Barthol2	148	170	3.2254	3.1278	3.1912	2.9669	3.1034

Table 4.2 until Table 4.4 presents the optimisation objective value in term of the number of workstation, machine and worker for small, medium and large size problem respectively. For the small size problem in Table 4.2, all crossovers came out with the same objective values. This result is directly related with the size of search space for this problem. For small size problem, the search space is relatively small compared with medium and large size problem. Therefore, the chances for the algorithm to find the best solution is much better in this problem.

Table 4.2 Optimisation objective value for small size problem

Problem	Objective	OX	PMX	Moon	RBC-I	RBC-II
Martens	Workstation	5	5	5	5	5
	Machine	8	8	8	8	8
	Worker	6	6	6	6	6
Bowman	Workstation	2	2	2	2	2
	Machine	4	4	4	4	4
	Worker	4	4	4	4	4
Jaeschke	Workstation	3	3	3	3	3
	Machine	6	6	6	6	6
	Worker	7	7	7	7	7
Mansoor	Workstation	4	4	4	4	4
	Machine	10	10	10	10	10
	Worker	5	5	5	5	5

Table 4.2 Continued

Problem	Objective	OX	PMX	Moon	RBC-I	RBC-II
Jackson	Workstation	4	4	4	4	4
	Machine	7	7	7	7	7
	Worker	7	7	7	7	7

In Table 4.3, the minimum objective values mostly found in RBC-II, as presented in the fitness function value in Table 4.1. The best fitness function value (Table 4.1) however not necessary to have the best value in all objectives. For example in the Gunther problem, the best fitness function value is attained by RBC-II. However, in Table 4.3, the RBC-II only have the minimum value in two objectives (i.e. Station and Machine), while for the number of worker, the best objective is found in Moon crossover. This is because of the normalizing effect as explained in section 3.2 in chapter 3. For the optimisation objective with a smaller range, small changes in the objective value give larger effect on the fitness function, compared with optimisation objective with a larger range.

Table 4.3 Optimisation objective value for medium size problem

Problem	Objective	OX	PMX	Moon	RBC-I	RBC-II
Buxey	Workstation	7	7	7	7	7
	Machine	21	23	22	20	20
	Worker	20	20	19	19	18
Sawyer	Workstation	5	5	5	5	5
	Machine	17	17	17	17	15
	Worker	9	9	9	11	8
Gunther	Workstation	8	8	8	8	8
	Machine	22	23	24	24	21
	Worker	12	17	11	15	12
Kilbridge	Workstation	9	9	9	9	9
	Machine	28	28	30	30	26
	Worker	23	26	22	25	24
Hahn	Workstation	8	8	8	8	8
	Machine	23	22	22	23	22
	Worker	20	18	19	19	19
Warnecke	Workstation	15	15	16	16	15
	Machine	49	45	46	44	43
	Worker	28	29	32	35	32

Table 4.4 shows the optimisation objective value for large size problem. For this class of problem, the minimum value is more scattered throughout the crossover types. Base on Table 4.1, the minimum fitness was found in RBC-I and II. Here, the normalizing

effect again contributed to the fitness value preference in RBC-I and II. For example in Mukherje problem, both Moon and RBC-II have two minimum optimisation objective value. However, the fitness for RBC-II is better than Moon crossover because the range of the machine number is larger than the range of worker number. Therefore, small changes in the number of worker contributed to better fitness compared with the number of machine.

Table 4.4 Optimisation objective value for large size problem

Problem	Objective	OX	PMX	Moon	RBC-I	RBC-II
Wee Mag	Workstation	31	31	31	31	31
	Machine	65	71	71	69	66
	Worker	47	50	48	47	44
Arc83	Workstation	13	13	13	13	13
	Machine	44	43	48	44	45
	Worker	33	33	32	31	37
Lutz 2	Workstation	28	28	28	27	28
	Machine	58	55	56	51	53
	Worker	52	44	52	47	45
Mukherje	Workstation	17	17	17	18	17
	Machine	66	69	63	65	68
	Worker	36	37	37	38	34
Arc111	Workstation	26	25	26	26	25
	Machine	103	126	104	99	105
	Worker	52	52	48	56	49
Barthol2	Workstation	27	27	27	27	27
	Machine	95	95	98	92	94
	Worker	68	64	70	70	66

To have better view from the computational experiment result, a standard competition ranking approach is used. The crossover with the best fitness will be given rank 1, followed by the next as rank 2, etc. If the crossover performance is equivalent, the following rank is ignored. The summary of the standard competition ranking is presented in Table 4.5. The value in this table indicated the frequency of the problem being rank in a specific ranking.

Table 4.5 Summary of standard competition ranking

Crossover	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Average rank
OX	5	3	3	2	4	2.8235
PMX	6	2	2	4	3	2.7647
Moon	5	3	3	4	2	2.7058
RBC-I	8	5	0	1	3	2.1764
RBC-II	13	1	3	0	0	1.4117

Based on Table 4.5, the RBC-II was most frequently ranked as 1, followed by RBC-I and PMX. Meanwhile, the OX has the most frequently ranked as 5. The average rank for each of crossover is then calculated. Based on the average rank the best crossover is RBC-II. The RBC-II is only ranked from 1 until 3. In the meantime, RBC-I is in the second best according to the average rank. For RBC-I, besides ranked as 1 and 2, this crossover was also ranked as 5 in three cases. On the other hand, the OX is the worst crossover based on the average rank.

The RBC-I and II have shown better performance because of the involvement of the best chromosome in the reproduction process. This makes the search direction is more guided compared with other crossovers. In the OX, PMX and Moon crossovers, the reproduction process solely depend on the parents. Even though the parents were selected among the best, the variation in the chromosomes makes the search direction become too diverse.

Meanwhile, in the comparison between RBC-I and II, the RBC-I is too dependent on the best solution because a single parent is mated with the best solution for the regeneration. This makes the chance for the chromosome to trap in local optima is slightly higher. In RBC-II, the regeneration process involved a pair of parents and the best solution. Two chromosomes from parents make the regeneration is not too relied on the best solution. Furthermore, the generated offspring only inherit 20% of the gene from the best solution (since C_{best} = 0.2). This makes the RBC-II able to generate more varied offspring but in the guided mode.

4.2 Optimisation Results for Case Study

Optimisation for the case study problem has been conducted using Genetic Algorithm with different crossovers as in section 4.1. For the case study optimisation, ten optimisation runs with different pseudo-random seeds have been done. The number of maximum generation is set to 300, while the probability of crossover (p_c) and mutation (p_m) are 0.8 and 0.2 respectively. Table 4.6 presents the fitness value for the case study problem from ten different runs.

Table 4.6 Fitness value for the case study problem

No.	Crossover type				
	PMX	**Moon**	**OX**	**RBC-I**	**RBC-II**
Run 1	2.61	2.40	2.00	2.40	2.20
Run 2	2.40	2.60	2.60	2.01	2.20
Run 3	2.40	2.60	2.60	2.20	2.00
Run 4	2.60	2.60	2.41	2.20	2.41
Run 5	2.40	2.40	2.40	2.60	2.40
Run 6	2.40	2.40	2.60	2.60	2.40
Run 7	2.40	2.40	2.80	2.21	2.00
Run 8	2.40	2.40	2.40	2.00	2.20
Run 9	2.40	2.40	2.20	2.40	2.20
Run 10	2.60	2.60	2.40	2.60	2.00
Min	2.40	2.40	2.00	2.00	2.00
Max	2.61	2.60	2.80	2.60	2.41
Average	2.4614	2.4800	2.4414	2.3229	2.2014

Based on the optimisation result for the case study problem in Table 4.6, the minimum fitness is 2.00, while the maximum fitness is 2.80. For the minimum fitness, three crossovers able to search for this solution. These crossovers were OX, RBC-I and RBC-II. Based on the average fitness value, the best crossover is the RBC-II. This is followed by RBC-I, OX, PMX and finally the Moon crossover. The details of solution found by the algorithms are in Appendix B.

The optimisation result for the case study problem indicated that the PMX and Moon crossovers have almost similar performance. Both crossovers are incapable to converge to minimum fitness. The OX crossover on the other hand able to search for optimum solution. However, the obtained fitness range was too diverged since the largest fitness value was also obtained by OX crossover. Meanwhile, the RBC-II has the best performance with 2.20 average fitness. In comparison with other crossover types, the RBC-II was also obtained the smallest maximum fitness value.

Figure 4.1 shows the average convergence of different crossover for the case study problem. According to the plot, the Moon crossover has the slowest convergence in the first 50 generations. Then the convergence was slowly occurred 50 to 200 generations. Meanwhile, the PMX and OX crossovers have moderate convergence in the early generation. Then a similar trend as found in Moon crossover was also observed from the 50 to the end of the generation. The RBC-I and RBC-II on the other hand converge rapidly in the first 20 generations. Then the convergence of RBC-I and RBC-II were almost

equivalent between 100 to 200 generations. The RBC-II continued to converge until the end of the total generation.

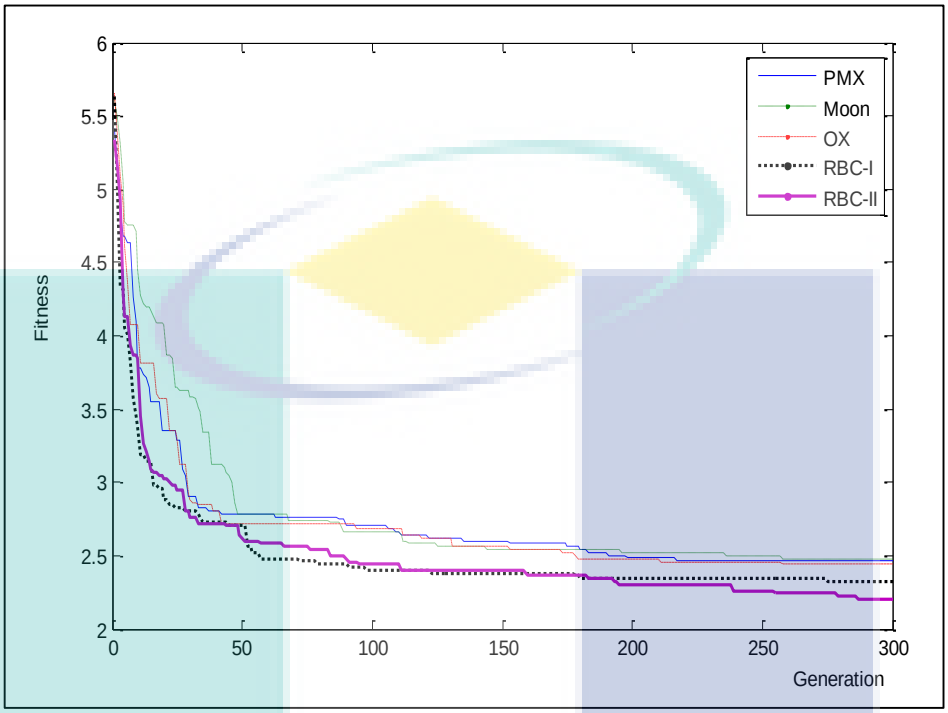


Figure 4.1 Convergence plot for different crossovers

The existing assembly process in the studied line is conducted in ten workstations and required ten workers. The existing assembly task assignment is presented in Table 4.7 Configuration of existing assembly layout. For the existing layout, 20 machines were needed to complete all assembly tasks.

Table 4.7 Configuration of existing assembly layout

Workstation	Assembly Task	Task Time (s)	Workstation time (s)	Machine	Worker
1	1	38.82	117.92	M1, M2	W1
	2	31.14			
	3	47.96			
2	4	88.86	116.18	M2,M3	W2
	5	27.32			
3	6	47.74	120.88	M2,M3	W3
	7	73.14			
4	8	20.90	112.22	M2,M3,M4	W4
	9	64.30			
	10	11.88			
	11	15.14			

Table 4.7 Continued

Workstation	Assembly Task	Task Time (s)	Workstation time (s)	Machine	Worker
5	12	64.78	122.70	M5,M6	W5
	13	57.92			
6	14	59.88	104.82	M5	W6
	15	44.94			
7	16	31.80	59.06	M6,M7	W7
	17	27.26			
8	18	69.82	92.10	M6,M7	W8
	19	22.28			
9	20	43.90	83.34	M4,M7,M8	W9
	21	19.36			
	22	20.08			
10	23	33.82	33.82	M5	W10

Table 4.8 meanwhile shows the best optimisation result obtained by the GA with RBC-II. For the same work content, the suggested solution by RBC-II can be completed in nine workstations that operated by nine workers. On the other hand, the number of required machine was reduced to 17.

Table 4.8 Configuration of optimised assembly layout

Workstation	**Assembly Task**	**Task Time (s)**	**Workstation time (s)**	**Machine**	**Worker**
1	1	38.82	117.92	M1,M2	W1
	2	31.14			
	3	47.96			
2	4	88.86	116.18	M2,M3	W2
	5	27.32			
3	6	47.74	83.78	M2,M4	W3
	8	20.90			
	11	15.14			
4	20	43.90	117.04	M3,M4	W4
	7	73.14			
5	9	64.30	118.38	M3,M7	W5
	16	31.80			
	19	22.28			
6	22	20.08	78.00	M6,M7	W6
	13	57.92			
7	14	59.88	104.82	M5	W7
	15	44.94			
8	17	27.26	108.96	M4,M6	W8
	18	69.82			
	10	11.88			
9	12	64.78	117.96	M5,M8	W9
	21	19.36			
	23	33.82			

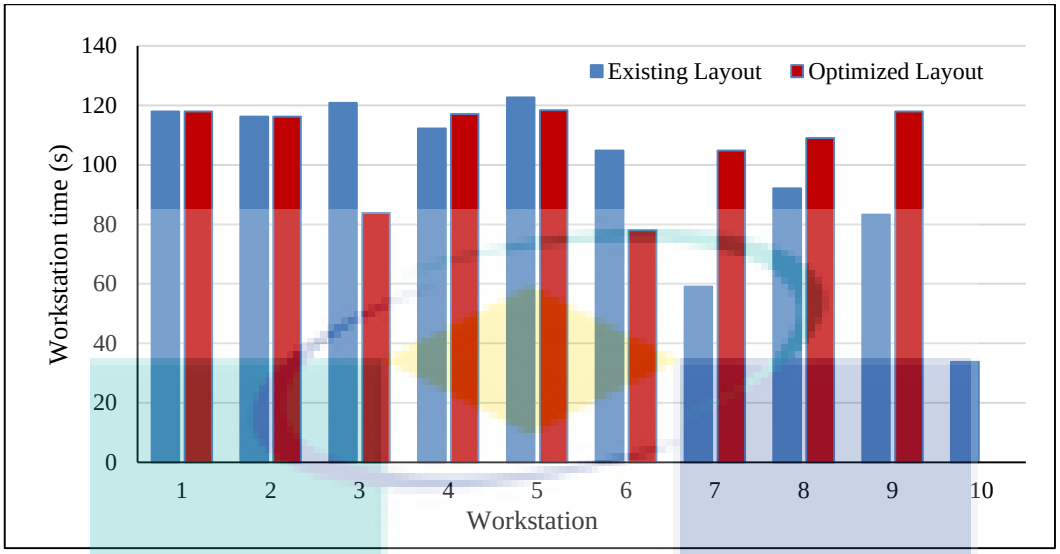


Figure 4.2 Comparison of workstation time

Figure 4.2 shows a comparison of workstation time for existing and optimised layout. Based on the bar graph, the optimised layout is more balanced within a smaller gap between minimum and maximum time compared with the existing layout. It indicated that the optimised layout has better (smaller) idle time compared with the existing layout. Based on this observation, the percentage of busy in the optimise layout will increase.

Besides that, the maximum workstation time for the optimised layout is slightly smaller than the existing layout. In the assembly line, the maximum workstation time is also known as cycle time for the assembly. The cycle time will control the production pace for the whole assembly line. With a smaller cycle time, the optimised layout is predicted to produce more output compared with the existing layout.

Besides the objective function in the optimisation that measure the number of workstation, machine and worker, a few other indicators to measure the line balance and efficiency can be used to compare the assembly layout before and after optimisation. The comparison of existing and optimised assembly layout indicators is shown in Table 4.9.

Table 4.9 Comparison of existing and optimised assembly layout indicators

Indicator	Existing Layout	Optimised Layout
Number of workstation	10	9
Number of machine	20	17
Number of worker	10	9
Smoothness index	122.20	55.74
Line efficiency (%)	78.48	90.39

Based on the comparison of the existing and optimised layout, the number of workstation and worker were reduced about 10% from the existing configuration. Meanwhile, the number of machine also was reduced from 20 to 17 units. This is about 15% reduction from the existing layout. Besides that, the smoothness index and line efficiency for the optimised layout were also better than the existing. This is because the number of workstation and also the cycle time for the optimised layout is lower than the current layout. The cycle time refers to the maximum workstation time among all workstations. In the existing layout, the cycle time is 122.7 seconds, while in the optimised layout, the cycle time was reduced to 118.38 seconds. The cycle time will control the production pace in assembly line.

4.3 Simulation of Case Study Problem

This section presents simulation results for the case study problem to validate the optimisation output. Simulation has been conducted using Witness simulation software, by Lanner Group with computation time is one (1) hour. The simulation is conducted for the assembly line before and after optimisation.

Table 4.10 and Table 4.11 presents the workstation time before and after optimisation. The workstation time is the cumulative time for all tasks in a particular workstation. For example, the first data for workstation 1 (106.7 seconds) is the summation of time for task 1, 2 and 3 (since task 1-3 are in workstation 1) for the first reading (Refer to Table 4.7 and Table 4.8). Then the standard deviation for each time is calculated. In the simulation model, the average time is inserted, with the calculated standard deviation.

Table 4.10 Workstation time before optimisation

Workstation	Workstation time (s)					Average time	Standard deviation
	1	2	3	4	5		
1	106.7	122	123.6	118.3	119	117.92	6.63
2	113.3	119.8	116.6	116.3	114.9	116.18	2.41
3	121.6	123.1	120	121.4	118.3	120.88	1.81
4	113.6	110.5	111.2	110.9	114.9	112.22	1.93
5	132.9	116.6	116.6	122.2	125.2	122.70	6.80
6	101	109.9	103	105.1	105.1	104.82	3.31
7	57.6	55.1	60.4	60.9	61.3	59.06	2.65
8	90.6	93.9	86.7	91.2	98.1	92.10	4.23
9	77.2	84	86.3	81.1	88.1	83.34	4.32
10	36.5	36.5	34.8	29.1	32.2	33.82	3.17

Table 4.11 Workstation time after optimisation

Workstation	Workstation time (s)					Average time	Standard deviation
	1	2	3	4	5		
1	106.7	122	123.6	118.3	119	117.92	6.65
2	113.3	119.8	116.6	116.3	114.9	116.18	2.41
3	77.9	81	91.7	89.4	78.9	83.78	6.33
4	121.5	122.7	106.4	117.1	117.5	117.04	6.43
5	118.8	115.1	118.9	111.1	128	118.38	6.26
6	80.1	75.6	83.2	70.6	80.5	78.00	4.96
7	101	109.9	103	105.1	105.1	104.82	3.31
8	107.9	110.1	100.9	114.2	111.7	108.96	5.06
9	123.8	115.2	114.9	114.4	121.5	117.96	4.37

Table 4.12 and Table 4.13 presents the simulation results for the existing and optimised layout. In this study, four simulation outputs are considered; the average idle, busy and block percentages, and the number of output.

Table 4.12 Simulation result for existing layout

Repetition	Average Idle %	Average Busy %	Average Block %	No. of Output
1	21.442	76.855	1.703	226
2	21.487	76.976	1.536	227
3	21.547	76.547	1.905	225
4	21.413	76.704	1.884	225
5	21.495	76.59	1.916	225
6	21.897	76.574	1.527	225
7	21.664	76.622	1.713	225
8	21.59	76.703	1.709	226
9	21.482	76.694	1.824	225
10	21.351	77.031	1.618	227

The company has targeted to produce 6000 unit of product per month with average daily output is 240 units per day. If compared the existing daily output in the simulation, number of output was 225 units to 227 units. This simulation result is acceptable since the percentage of error is only 5-6%. Referring to Baril et al. (2016) this percentage of error is acceptable since the margin of error related to the statistics curves and the error induced by a simplified process is ±10%.

Table 4.13 Simulation result for optimised layout

Repetition	Average Idle %	Average Busy %	Average Block %	No. of Output
1	9.453	86.470	4.078	229
2	9.204	86.880	3.916	230
3	9.279	86.788	3.932	229
4	8.847	87.069	4.084	230
5	8.939	86.792	4.269	229
6	8.896	86.608	4.497	228
7	9.132	86.332	4.538	228
8	8.840	86.953	4.206	230
9	8.967	86.707	4.327	229
10	8.969	86.887	4.144	229

According to the observation from Table 4.12 and Table 4.13, the average idle percentage for optimised layout has reduced compared with the original layout. Meanwhile, the average busy, average block and number of daily output for optimised layout have increased compared with the existing layout. This observation has been expected based on the line efficiency and smoothness index in Table 4.9, except for the average block percentage.

Based on the detail simulation result for optimised layout (Appendix C2), the highest blockage occurred at station 3. This is because station 3 has lower workstation time compared with station 4. For the time difference is about 40%, the task in station 3 will be completed faster than station 4. This makes the part from station 3 cannot be transferred into station 4 since the buffer is not used in the model.

In order to measure the simulation results before and after optimisation, a two-sample *t*-test is conducted. At 95% confidence interval, the *t*-test for each of simulation output is conducted with the following hypothesis.

H0: The mean of the two samples are equal

H1: The mean of the two samples are different

For the simulation result with ten repetitions, the degree of freedom is 17, while the critical *t* (*t**) value is 2.1098. The summary of *t*-test is presented in Table 4.14.

Table 4.14 Summary of t-test

Simulation indicator	P value	t value
Average Idle %	3.7599E-28	153.4502
Average Busy %	7.4634E-26	112.3768
Average Block %	1.49608E-15	30.2615
No. of Output	1.0791E-08	9.8775

Based on Table 4.14, all *P* values are smaller than $\alpha = 0.05$, for 95% confidence interval. In the same time, all the *t* values are larger than critical *t*. This result means that the null hypothesis needs to be rejected. Therefore, the *t*-test indicated that there is a significant difference between the simulation result before and after optimisation. In other words, the optimised results have shown significant improvement for the average idle percentage, average busy percentage and the number of daily output. The result also means that the average blockage percentage is significantly increased in the optimised layout.

The result from industrial case study shows that the proposed ALB-RC model able to reduce the assembly resources in actual assembly environment without ignoring the standard indicators in assembly line balancing such as number of workstation, cycle time, smoothness index and line efficiency. This result also verified that the proposed RBC-II crossovers capable to perform well compared with comparison crossovers for the actual industrial data. The simulation result has validated the proposed assembly task configuration from optimisation.

4.4 Comparison Results for Benchmark Problem and Case Study

In order to make a comparison on the result obtained in benchmark problem and case study, average fitness value in medium size problem for benchmark problem

(Problem 6 to 11 in Table 4.1) is calculated and ranked as in Table 4.15. This is to make a valid comparison since the size problem of the case study is a medium size problem.

Table 4.15 Rank for benchmark medium size problem

	OX	PMX	Moon	RBC-I	RBC-II
Average	3.5755	3.6040	3.6452	3.5573	3.3627
Rank	3	4	5	2	1

The rank for benchmark problem and case study is then compared as in Table 4.16. The rank result shows that RBC-II is rank as 1 followed by RBC-I, OX, PMX. Meanwhile, Moon is the worst compared to other types of crossover. This result can validate the effectiveness of the proposed crossover in optimizing ALB-RC.

Table 4.16 Comparison of crossover for medium size problem

	OX	PMX	Moon	RBC-I	RBC-II
Benchmark	3	4	5	2	1
Case Study	3	4	5	2	1

4.5 Summary of the Results

This chapter presents the result of this research. In section 4.1, the computational experiment results were discussed. The computational experiment result indicated that the proposed Rank Based Crossover type I and II (RBC-I and RBC-II) have better overall performance. This finding answered the second research objective: to propose an improved algorithm to optimise Simple Assembly Line Balancing Problem Type 1 with resource and worker constraints.

Section 4.2 meanwhile presents the finding for the case study problem. For the optimisation of the case study problem, the RBC-II came out with the best solution. In comparison with the existing layout configuration, the proposed solution by RBC-II is predicted to reduce the number of workstation, machine and worker. At the same time, the assembly line efficiency is also increased. The optimisation result for case study problem is then being simulated using Witness software as presented in Section 4.3. The *t*-test for simulation results validated that the output from optimisation is capable to improve the assembly line efficiency.

CHAPTER 5

CONCLUSIONS

5.1 Summary of the Research

In this research, problems of assembly line balancing with resource constraints (ALB-RC) were studied. The problem is a non-deterministic polynomial (NP) hard class problem due to the complexity of the problems. The existing methods such as exact methods have the limitation of size to solve ALB-RC. Meanwhile, the heuristic approaches have a limitation in term of solution quality.

Metaheuristic approach is more practical for large problems and can locate optimum solution. In metaheuristic approach, a wide range of algorithm can be used to solve different types of problems. In this research, Genetic Algorithm was used to solve ALB-RC due to the capability of the algorithm to find the best feasible solution in different applications. In order to strengthen the algorithm to find the best solution for ALB-RC, two proposed crossovers were introduced. The algorithm with the proposed new crossovers were successfully developed using the rank-based mechanism. These crossovers are known as Rank Based Crossover Type I and II (RBC-I and RBC-II).

In this research, the optimisation objectives are to minimise the number of workstation, number of machine/tool and number of worker while at the same time balancing and optimizing the assembly line. The proposed algorithm with the new crossovers are searching for the best feasible arrangement of task, worker and machine without violating the constraints and limitation which may have in the assembly line.

Numerical experiment was conducted to evaluate and verified the performance of the proposed RBC-I and II for ALB-RC before it is applied and tested on an industrial case study problem. The numerical experiment was conducted by using different sizes of

benchmark problem range from small to large size. For comparison purpose, three popular crossovers for line balancing problem were also implemented. These crossovers are Ordered crossover (OX), Partially Matched crossover (PMX) and Moon crossover. The numerical experiment and the industrial case study results show that the proposed RBC-I and RBC-II is more efficient and capable in generating the optimal solution for ALB-RC compared to OX, PMX and Moon crossover. Meanwhile, when comparing RBC-I and RBC-II, RBC-II has the best performance.

Later, an industrial case study problem was used to validate the proposed model and algorithm for ALB-RC. This problem was optimised using GA with different crossovers as mention earlier. The optimisation result indicated that the best solution provided by RBC-II able to reduce the number of workstation, machine and worker. For validation purpose, a discrete event simulation was conducted using Witness simulation software. The simulation results concluded that the optimised layout by RBC-II has a significant improvement compared with the existing line in term of idle and busy percentage, and also in term of the number of daily output.

5.2 Research Contributions

The general contribution of this research as outlined in the research aim is the establishment of a methodology and algorithm for optimisation of ALB-RC. In order to achieve the research aim, this research has delivered a number of specific contributions to knowledge.

The proposed ALB resource constraint model was developed based on the assembly task. The proposed representation scheme clearly defines the resource constraints; machine and worker on assembly tasks, which has not been done before. This contribution is important because without clear definition of assembly line with resource constraint on the representation, the implementation of this scheme to real-world problems would be impossible.

The second contribution is the proposed Rank-based crossovers for Genetic Algorithm to optimise ALB-RC. This algorithm has been tested using a different range of problem from the benchmark test problem and have performed well in finding the

optimal solutions. The Rank-based crossovers for Genetic Algorithm performance has been validated using the benchmark problem, while the optimisation results (optimised layout) has been validated using simulation. The proposed formulation allows manufacturers to gain benefits such as cost saving for production line where they can reduce the number of machine and worker needed to perform task in an assembly line.

5.3 Research Conclusions

In relation to the research objectives;

i. This research had successfully proposed a model of Assembly Line Balancing with resource constraints (ALB-RC) for SALBP-1. The resources considered in this research were the machines and workers.

ii. An improved version of Genetic Algorithm was successfully proposed using new Rank-Based Crossovers for ALB-RC. The performance was validated through computational experiment using benchmark problem.

iii. The proposed ALB-RC model and RBC-II result had successfully being validated through industrial case study problem.

Based on the accomplishment of research objectives, it can be concluded that this research has established a methodology and algorithm for generating optimal solution for ALB-RC. Therefore, the aim of this research has successfully been achieved.

5.4 Limitations and Recommendations for Future Works

Although this research had successfully been done, there are a few limitations that observed.

- The proposed model is limited to simple assembly line balancing type I (SALBP-I) only. Besides SALBP-I there are many other types of assembly line problem used in industry.

- The algorithm comparison is limited to Genetic Algorithm only. Recently, there are various type of metaheuristic algorithms have been introduced for optimisation.

Based on the limitations of this research, several recommendations for future research are proposed. The future directions of the research are summarised as follows.

- Extend the application for other types of assembly line problem. For instant, this research is limited to the simple assembly line resource constraints problem type 1 which only can be applied if the assembly line produce a single product on a single assembly line.
- Compare the performance of the proposed algorithm with other algorithms that have good potential such as Simulated Annealing, Memetic Algorithm or hybrid algorithms.

REFERENCES

Ağpak, K., Gökçen, H., Ağpak, K., & Gökçen, H. (2005). Assembly line balancing: Two resource constrained cases. *International Journal of Production Economics*, *96*(1), 129–140. https://doi.org/10.1016/j.ijpe.2004.03.008

Akpinar, S., Elmi, A., & Bektaş, T. (2017). Combinatorial Benders cuts for assembly line balancing problems with setups. *European Journal of Operational Research*, *259*, 527–537. https://doi.org/http://dx.doi.org/10.1016/j.ejor.2016.11.001

Akpınar, S., Mirac Bayhan, G., & Baykasoglu, A. (2013). Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. *Applied Soft Computing*, *13*(1), 574–589. https://doi.org/10.1016/j.asoc.2012.07.024

Al-Ahmari, A., Ameen, W., Abidi, M. H., & Mian, S. H. (2018). Evaluation of 3D printing approach for manual assembly training. *International Journal of Industrial Ergonomics*, *66*, 57–62. https://doi.org/https://doi.org/10.1016/j.ergon.2018.02.004

Alavidoost, M. H., Tarimoradi, M., & Zarandi, M. H. F. (2015). Fuzzy adaptive genetic algorithm for multi-objective assembly line balancing problems. *Applied Soft Computing*, *34*, 655–677.

Amin, M. A., & Karim, M. A. (2013). A time-based quantitative approach for selecting lean strategies for manufacturing organisations. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2012.693639

Azizoğlu, M., & İmat, S. (2018). Workload smoothing in simple assembly line balancing. *Computers & Operations Research*, *89*, 51–57. https://doi.org/https://doi.org/10.1016/j.cor.2017.08.006

Barathwaj, N., Raja, P., & Gokulraj, S. (2015). Optimization of assembly line balancing using genetic algorithm. *Journal of Central South University*, *22*(10), 3957–3969. https://doi.org/10.1007/s11771-015-2940-9

Baril, C., Gascon, V., Miller, J., & Côté, N. (2016). Use of a discrete-event simulation in a Kaizen event : A case study in healthcare. *European Journal of Operational Research*, *249*(1), 327–339. https://doi.org/10.1016/j.ejor.2015.08.036

Bautista, J., & Pereira, J. (2007). Ant algorithms for a time and space constrained assembly line balancing problem. *European Journal of Operational Research*, *177*(3), 2016–2032. https://doi.org/10.1016/j.ejor.2005.12.017

Baybars, I. (1986). An efficient heuristic method for the simple assembly line balancing problem. *International Journal of Production Research*, *24*(1), 149–166.

Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, *168*(3), 694–715. https://doi.org/10.1016/j.ejor.2004.07.023

Borba, L., & Ritt, M. (2014). A heuristic and a branch-and-bound algorithm for the assembly line worker assignment and balancing problem. *Computers and Operations Research*, *45*, 87–96. https://doi.org/10.1016/j.cor.2013.12.002

Boysen, N., Fliedner, M., & Scholl, A. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research*, *183*(2), 674–693. https://doi.org/10.1016/j.ejor.2006.10.010

Boysen, N., Fliedner, M., & Scholl, A. (2008). Assembly line balancing: Which model to use when? *International Journal of Production Economics*, *111*(2), 509–528. https://doi.org/10.1016/j.ijpe.2007.02.026

Briano, E., Caballini, C., Mosca, R., & Revetria, R. (2010). Using WITNESS Simulation software as a validation tool for an industrial plant layout. *SYSTEM SCIENCE and SIMULATION in ENGINEERING*, 201–206.

Che, Z. H. (2017). A multi-objective optimization algorithm for solving the supplier selection problem with assembly sequence planning and assembly line balancing. *Computers & Industrial Engineering*, *105*, 247–259. https://doi.org/https://doi.org/10.1016/j.cie.2016.12.036

Chica, M., Cordón, Ó., & Damas, S. (2011). Tackling the 1/3 variant of the time and space assembly line balancing problem by means of a multiobjective genetic algorithm. *2011 IEEE Congress of Evolutionary Computation, CEC 2011*, *61*(1), 1367–1374. https://doi.org/10.1109/CEC.2011.5949775

Chutima, P., & Kid-Arn, S. (2013). PSONK: Particle Swarm Optimization With Negative Knowledge For Multi-Objective U-Shaped Assembly Lines Balancing With Parallel Workstations. *Journal of Advanced Manufacturing Systems*, *12*(1), 15–41. https://doi.org/10.1142/S0219686713500029

Corominas, A., Ferrer, L., & Pastor, R. (2011). Assembly line balancing: General resource-constrained case. *International Journal of Production Research*, *49*(12), 3527–3542. https://doi.org/10.1080/00207543.2010.481294

Desale, S., Rasool, A., Andhale, S., & Rane, P. (2015). Heuristic and Meta-Heuristic Algorithms and Their Relevance to the Real World: A Survey. *International Journal of Computer Engineering in Research Trends*, *351*(5), 2349–7084.

Dolgui, A., & Proth, J. (2013). Assembly Line Balancing : Conventional Methods and Extensions. In *IFAC Proceedings Volumes* (Vol. 46, pp. 43–48). IFAC. https://doi.org/10.3182/20130619-3-RU-3018.00644

Dong, J., Zhang, L., & Xiao, T. (2018). A hybrid PSO/SA algorithm for bi-criteria stochastic line balancing with flexible task times and zoning constraints. *Journal of Intelligent Manufacturing*, *29*(4), 737–751. https://doi.org/10.1007/s10845-015-1126-5

Erdal Erel & Subhash C. Sarin. (1998). A survey of the assembly line balancing procedures. *Production Planning & Control*, *9*(5), 414–434. https://doi.org/10.1080/095372898233902

Esmaeilbeigi, R., Naderi, B., & Charkhgard, P. (2015). The type E simple assembly line balancing problem: A mixed integer linear programming formulation. *Computers & Operations Research*, *64*, 168–177. https://doi.org/10.1016/j.cor.2015.05.017

Fathi, M. (2011). A New Heuristic Method Based on CPM in SALBP. *Journal of Industrial Engineering International*, *7*(13), 1–11.

Fathi, M., Fontes, D. B. M. M., Moris, M. U., & Ghobakhloo, M. (2018). Assembly line balancing problem: A comparative evaluation of heuristics and a computational assessment of objectives. *Journal of Modelling in Management*, *13*(2), 455–474. https://doi.org/10.1108/JM2-03-2017-0027

Gansterer, M., & Hartl, R. F. (2018). One- and two-sided assembly line balancing problems with real-world constraints. *International Journal of Production Research*, *56*(8), 3025–3042. https://doi.org/10.1080/00207543.2017.1394599

Grzechca, W., & Foulds, L. R. (2015). The assembly line balancing problem with task splitting: A case study. *IFAC- Papers On Line*, *28*(3), 2002–2008. https://doi.org/10.1016/j.ifacol.2015.06.382

Hamta, N., Fatemi Ghomi, S. M. T., Jolai, F., & Bahalke, U. (2011). Bi-criteria assembly line balancing by considering flexible operation times. *Applied Mathematical Modelling*, *35*(12), 5592–5608. https://doi.org/10.1016/j.apm.2011.05.016

Hamzas, M. F. M. A., Bareduan, S. A., Zakaria, M. Z., Tan, W. J., & Zairi, S. (2017). Validation of X1 motorcycle model in industrial plant layout by using WITNESSTM simulation software, *20182*, 20182. https://doi.org/10.1063/1.5002376

Jaffrey, V., & Mohamed, N. M. Z. N. (2018). Assembly Line Efficiency Improvement by Using WITNESS Simulation Software Assembly Line Efficiency Improvement by Using WITNESS Simulation Software, (2581). https://doi.org/10.1088/1757-899X/319/1/012004

Janardhanan, M. N., Li, Z., Nielsen, P., & Tang, Q. (2018). Artificial bee colony algorithms for two-sided assembly line worker assignment and balancing problem. *Advances in Intelligent Systems and Computing*, *620*, 11–18. https://doi.org/10.1007/978-3-319-62410-5_2

Jayaswal, S., & Agarwal, P. (2014). Balancing U-shaped assembly lines with resource dependent task. *Journal of Manufacturing Systems*. https://doi.org/10.1016/j.jmsy.2014.05.002

Jusop, M., & Ab. Rashid, M. (2017). Optimization of Assembly Line Balancing with Resource Constraint using NSGA-II : A Case Study. *International Journal of Applied Engineering Research*, *12*(7), 1421–1426.

Jusop, M., & Ab. Rashid, M. F. F. (2016). Optimisation of assembly line balancing type-E with resource constraints using NSGA-II. *Key Engineering Materials*, *701*, 195–199. https://doi.org/10.4028/www.scientific.net/KEM.701.195

Kao, H.-H., Yeh, D.-H., Wang, Y.-H., & Hung, J.-C. (2010). An optimal algorithm for type-I assembly line balancing problem with resource constraint. *African Journal of Business Management*, *4*(10), 2051–2058.

Kara, Y., Özgüven, C., Yalçin, N., & Atasagun, Y. (2011). Balancing straight and U-shaped assembly lines with resource dependent task times. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2010.535039

Koltai, T., & Tatay, V. (2013). Formulation of workforce skill constraints in assembly line balancing models. *Optimization and Engineering*, *14*(4), 529–545. https://doi.org/10.1007/s11081-013-9230-x

Kovalev, S., Delorme, X., Dolgui, A., & Oulamara, A. (2017). Minimizing the number of stations and station activation costs for a production line. *Computers and Operations Research*. https://doi.org/10.1016/j.cor.2016.10.007

Kucukkoc, I., & Zhang, D. Z. (2015). Type-E parallel two-sided assembly line balancing problem: Mathematical model and ant colony optimisation based approach with optimised parameters. *Computers and Industrial Engineering*. https://doi.org/10.1016/j.cie.2014.12.037

Kucukkoc, I., & Zhang, D. Z. (2016). Mixed-model parallel two-sided assembly line balancing problem: A flexible agent-based ant colony optimization approach. *Computers and Industrial Engineering*, *97*, 58–72. https://doi.org/10.1016/j.cie.2016.04.001

Kumar, R., & Member, S. (2012). Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms. *International Journal of Machine Learning and Computing*, *2*(4), 365–370.

Li, D., Zhang, C., Shao, X., & Lin, W. (2016). A multi-objective TLBO algorithm for balancing two-sided assembly line with multiple constraints. *Journal of Intelligent Manufacturing*, *27*(4), 725–739. https://doi.org/10.1007/s10845-014-0919-2

Li, Z., Tang, Q., & Zhang, L. P. (2017). Two-sided assembly line balancing problem of type I: Improvements, a simple algorithm and a comprehensive study. *Computers and Operations Research*. https://doi.org/10.1016/j.cor.2016.10.006

Liu, S. B., Ng, K. M., & Ong, H. L. (2006). Branch-and-bound algorithms for simple assembly line balancing problem. *The International Journal of Advanced Manufacturing Technology*, *36*(1–2), 169–177. https://doi.org/10.1007/s00170-006-0821-y

Lv, H., & Lu, C. (2010). An assembly sequence planning approach with a discrete particle swarm optimization algorithm. *The International Journal of Advanced Manufacturing Technology*, *50*(5–8), 761–770. https://doi.org/10.1007/s00170-010-2519-4

McCall, J. (2005). Genetic algorithms for modelling and optimisation. *Journal of Computational and Applied Mathematics*, *184*(1), 205–222. https://doi.org/10.1016/j.cam.2004.07.034

Moon, C., Kim, J., Choi, G., & Seo, Y. (2002). An efficient genetic algorithm for the traveling salesman problem with precedence constraints. *European Journal of Operational Research*, *140*(3), 606–617. https://doi.org/10.1016/S0377-2217(01)00227-2

Moon, I., Logendran, R., & Lee, J. (2009). Integrated assembly line balancing with resource restrictions. *International Journal of Production Research*, *47*(19), 5525–5541. https://doi.org/10.1080/00207540802089876

Moreira, M. C. O., Pastor, R., Costa, A. M., & Miralles, C. (2017). The multi-objective assembly line worker integration and balancing problem of type-2. *Computers & Operations Research*, *82*, 114–125. https://doi.org/10.1016/j.cor.2017.01.003

Mura, M. D., & Dini, G. (2016). Worker Skills and Equipment Optimization in Assembly Line Balancing by a Genetic Approach. *Procedia CIRP*, *44*, 102–107. https://doi.org/https://doi.org/10.1016/j.procir.2016.02.033

Mutlu, Ö., Polat, O., & Supciller, A. A. (2013). An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-II. *Computers and Operations Research*, *40*(1), 418–426. https://doi.org/10.1016/j.cor.2012.07.010

Özbakir, L., & Tapkan, P. (2011). Bee colony intelligence in zone constrained two-sided assembly line balancing problem. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2011.03.089

Özdemir, R. G., & Ayağ, Z. (2011). An integrated approach to evaluating assembly-line design alternatives with equipment selection. *Production Planning & Control*, *22*(2), 194–206. https://doi.org/10.1080/09537281003790515

P. Semanco and D. Marton. (2013). Simulation Tools Evaluation using Theoretical Manufacturing Model. *Acta Polytechnica Hungarica*, *10*(2), 193–204.

Pape, T. (2015). Heuristics and lower bounds for the simple assembly line balancing problem type 1: Overview, computational tests and improvements. *European Journal of Operational Research*, *240*(1), 32–42. https://doi.org/10.1016/j.ejor.2014.06.023

Purnomo, H. D., Wee, H. M., & Rau, H. (2013). Two-sided assembly lines balancing with assignment restrictions. *Mathematical and Computer Modelling*, *57*(1–2), 189–199. https://doi.org/10.1016/j.mcm.2011.06.010

Rada-vilela, J., Chica, M., Cordón, Ó., & Damas, S. (2013). A comparative study of Multi-Objective Ant Colony Optimization algorithms for the Time and Space Assembly Line Balancing Problem. *Applied Soft Computing Journal*, *13*(11), 4370–4382. https://doi.org/10.1016/j.asoc.2013.06.014

Raj, A. S. V., Mathew, J., Jose, P., & Sivan, G. (2016). Optimization of Cycle Time in an Assembly Line Balancing Problem. *Procedia Technology*. https://doi.org/10.1016/j.protcy.2016.08.231

Rajakumar, S., Arunachalam, V. P., & Selladurai, V. (2006). Workflow balancing in parallel machine scheduling with precedence constraints using genetic algorithm. *Journal of Manufacturing Technology Management*, *17*(2), 239–254. https://doi.org/10.1108/17410380610642296

Ramezanian, R., & Ezzatpanah, A. (2015). Modeling and solving multi-objective mixed-model assembly line balancing and Worker Assignment Problem. *Computers & Industrial Engineering*, *87*, 74–80. https://doi.org/10.1016/j.cie.2015.04.017

Rashid, M. F. F., Hutabarat, W., & Tiwari, A. (2012). A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. *The International Journal of Advanced Manufacturing Technology*, *59*(1–4), 335–349. https://doi.org/10.1007/s00170-011-3499-8

Razali, N. M., & Geraghty, J. (2011). Genetic Algorithm Performance with Different Selection Strategies in Solving TSP. In *World Congress on Engineering* (Vol. II, pp. 4–9). London.

Roshani, A., & Giglio, D. (2015). A simulated annealing approach for multi-manned assembly line balancing problem type II. *15th IFAC Symposium on Information Control in Manufacturing*, *48*(3), 2367–2372. https://doi.org/10.1016/j.ifacol.2015.06.430

Saif, U., Guan, Z., Liu, W., Zhang, C., & Wang, B. (2014). Pareto based artificial bee colony algorithm for multi objective single model assembly line balancing with uncertain task times. *Computers and Industrial Engineering*. https://doi.org/10.1016/j.cie.2014.07.009

Scholl, A. (1993). Benchmark Data Sets by Scholl. Retrieved from http://assembly-line-balancing.mansci.de/salbp/benchmark-data-sets-1993/

Scholl, A., Boysen, N., & Fliedner, M. (2008). The sequence-dependent assembly line balancing problem. *OR Spectrum*. https://doi.org/10.1007/s00291-006-0070-3

Scholl, A., Boysen, N., & Fliedner, M. (2011). The assembly line balancing and scheduling problem with sequence-dependent setup times: problem extension, model formulation and efficient heuristics. *OR Spectrum*, *35*(1), 291–320. https://doi.org/10.1007/s00291-011-0265-0

Sharma, P. (2015). Discrete-Event Simulation. *International Journal of Scientific & Technology Research*, *4*(4), 136–140. https://doi.org/10.1007/978-1-4471-5634-5_10

Shuang, B., Chen, J., & Li, Z. (2007). Microrobot based micro-assembly sequence planning with hybrid ant colony algorithm. *The International Journal of Advanced Manufacturing Technology*, *38*(11–12), 1227–1235. https://doi.org/10.1007/s00170-007-1165-y

Sikora, C. G. S., Lopes, T. C., & Magatão, L. (2017). Traveling worker assembly line (re)balancing problem: Model, reduction techniques, and real case studies. *European Journal of Operational Research*, *259*(3), 949–971. https://doi.org/10.1016/j.ejor.2016.11.027

Sikora, C. G. S., Lopes, T. C., Schibelbain, D., & Magatão, L. (2016). Integer Based Formulation for the Simple Assembly Line Balancing Problem with Multiple Identical Tasks. *Computers & Industrial Engineering, in review*, 134–144. https://doi.org/10.1016/j.cie.2016.026

Simaria, A. S., & Vilarinho, P. M. (2004). A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II. *Computers & Industrial Engineering*, *47*(4), 391–407. https://doi.org/10.1016/j.cie.2004.09.001

Sivasankaran, P., & Shahabudeen, P. (2014). Study and Analysis of GA-Based Heuristic Applied to Assembly Line Balancing Problem. *Journal of Advanced Manufacturing Systems*, *13*(2), 113–131. https://doi.org/10.1142/S0219686714500085

Sternatz, J. (2014). Enhanced multi-Hoffmann heuristic for efficiently solving real-world assembly line balancing problems in automotive industry. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2013.11.005

Tang, Q., Li, Z., Zhang, L., & Zhang, C. (2017). Balancing stochastic two-sided assembly line with multiple constraints using hybrid teaching-learning-based optimization algorithm. *Computers & Operations Research*, *82*, 102–113. https://doi.org/10.1016/j.cor.2017.01.015

Tapkan, P., Ozbakir, L., & Baykasoglu, A. (2012). Modeling and solving constrained two-sided assembly line balancing problem via bee algorithms. *Applied Soft Computing*, *12*(11), 3343–3355. https://doi.org/10.1016/j.asoc.2012.06.003

Tasan, S. O., & Tunali, S. (2008). A review of the current applications of genetic algorithms in assembly line balancing. *Journal of Intelligent Manufacturing*, *19*(1), 49–69. https://doi.org/10.1007/s10845-007-0045-5

Taylor, P. (2010). International Journal of Production Assembly line balancing : general resource-constrained case, (August 2014), 37–41. https://doi.org/10.1080/00207543.2010.481294

Taylor, P., Kim, Y. K., Kim, Y., & Kim, Y. J. (2010). Production Planning & Control : The Management of Operations Two-sided assembly line balancing : A genetic algorithm approach Two-sided assembly line balancing : a genetic algorithm approach, (November 2014), 37–41. https://doi.org/10.1080/095372800232478

Taylor, P., Mamun, A. A., Khaled, A. A., Ali, S. M., & Chowdhury, M. M. (2012). A heuristic approach for balancing mixed-model assembly line of type I using genetic algorithm, (November 2014), 37–41. https://doi.org/10.1080/00207543.2011.643830

Triki, H., Mellouli, A., & Masmoudi, F. (2014). A multi-objective genetic algorithm for assembly line resource assignment and balancing problem of type 2 (ALRABP-2), *2*. https://doi.org/10.1007/s10845-014-0984-6

Tuncel, G., & Aydin, D. (2014). Two-sided assembly line balancing using teaching-learning based optimization algorithm. *Computers and Industrial Engineering*, *74*(1), 291–299. https://doi.org/10.1016/j.cie.2014.06.006

Vilà, M., & Pereira, J. (2013). A branch-and-bound algorithm for assembly line worker assignment and balancing problems. *Computers and Operation Research*, *44*, 105–114. https://doi.org/10.1016/j.cor.2013.10.016

Wang, B., Guan, Z., Li, D., Zhang, C., & Chen, L. (2014). Two-sided assembly line balancing with operator number and task constraints: a hybrid imperialist competitive algorithm. *The International Journal of Advanced Manufacturing Technology*, *74*(5–8), 791–805. https://doi.org/10.1007/s00170-014-5816-5

Xu, W., & Xiao, T. (2009). Robust balancing of mixed model assembly line. *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, *28*(6), 1489–1502. https://doi.org/10.1108/03321640910992038

Yeh, D. H., & Kao, H. H. (2009). A new bidirectional heuristic for the assembly line balancing problem. *Computers and Industrial Engineering*, *57*(4), 1155–1160. https://doi.org/10.1016/j.cie.2009.05.004

Yin, Y., Stecke, K. E., & Li, D. (2018). The evolution of production systems from Industry 2.0 through Industry 4.0. *International Journal of Production Research*, *56*(1–2), 848–861. https://doi.org/10.1080/00207543.2017.1403664

Yu, J., & Yin, Y. (2009). Assembly line balancing based on an adaptive genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, *48*(1–4), 347–354. https://doi.org/10.1007/s00170-009-2281-7

Yuan, B., Zhang, C., & Shao, X. (2013). A late acceptance hill-climbing algorithm for balancing two-sided assembly lines with multiple constraints. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-013-0770-x

Yuguang, Z., Bo, A., & Yong, Z. (2016). A PSO algorithm for multi-objective hull assembly line balancing using the stratified optimization strategy. *Computers and Industrial Engineering*, *98*, 53–62. https://doi.org/10.1016/j.cie.2016.05.026

Zacharia, P. T., & Nearchou, A. C. (2016). A population-based algorithm for the bi-objective assembly line worker assignment and balancing problem. *Engineering Applications of Artificial Intelligence*, *49*, 1–9. https://doi.org/10.1016/j.engappai.2015.11.007

Zhang, H. (2018). An immune genetic algorithm for simple assembly line balancing problem of type 1. *Assembly Automation, InPress*. https://doi.org/10.1108/AA-08-2017-101

Zhao, X., Hsu, C. Y., Chang, P. C., & Li, L. (2016). A genetic algorithm for the multi-objective optimization of mixed-model assembly line based on the mental workload. *Engineering Applications of Artificial Intelligence*. https://doi.org/10.1016/j.engappai.2015.03.005

APPENDIX A
MATLAB CODE

Appendix A1: Fitness Function Code

```
function [ res ] = obj_func3(n,max_ct,seq)

aspdata;

for i=1:n
    test_time(1, i)=data_mat(seq(1,i),1);
    test_mach(1, i)=data_mat(seq(1,i),2);
end

ws=1;
pt=0;

for i=1:n
    if pt+test_time(1,i)<=max_ct
        test_ws(1,i)=ws;
        pt=pt+test_time(1,i);
    else
        ws=ws+1;
        pt=test_time(1,i);
        test_ws(1,i)=ws;
    end
end
no_ws=test_ws(1,n);
mach_counter=0;

for i=1:no_ws
    nc=0;
    for j=1:n
        if test_ws(1,j)==i
            nc=nc+1;
            req_m(1,nc)=test_mach(1,j);
        end
    end
    mach_counter=length(unique((req_m(1,:))))+mach_counter;
end

%disp('Workstation Machine Worker')
res=[no_ws mach_counter no_w];

end
```

Appendix A2: Worker Assignment Code

```
%to calculate/assign worker
%1. establish table rw
nworker=length(worker_mat(1,:));
w_mat=worker_mat;
w_assign=zeros(1,n);
for i=1:n
    s=[];sel_task=[];av_worker=[];
    sum_w=sum(w_mat')';
    s1=nonzeros(sum_w);
    if isempty(s1)==1
        %disp('stop')
        worker_assignment;
        break
    end

    s2=min(s1);
    s=find(sum_w==s2);
    sel_task=s(randi(length(s)));

    c2=0;
    av_worker=[];
    for j=1:nworker
        if w_mat(sel_task,j)==1
            c2=c2+1;
            av_worker(1,c2)=j;
        end
    end
    sel_worker=av_worker(randi(length(av_worker)));

    w_assign(1,sel_task)=sel_worker;

    w_mat(:,sel_worker)=0;
    w_mat(sel_task,:)=0;
    if i==n
        break
    end

end
w_assign;
zz=zeros(1,n);
no_w=0;
worker_task=[seq; zz];
for i=1:no_ws
    nc=0;
    req_task=[];
    new_worker_mat=[];
    for j=1:n
        if test_ws(1,j)==i
            nc=nc+1;
            req_task(1,nc)=seq(1,j);
        end
    end
    req_task;

    for task=1:length(req_task)
        new_worker_mat(task,:)=worker_mat(req_task(1,task),:);
    end
    if length(new_worker_mat(:,1))>1
        summatrix=sum(new_worker_mat);
    else
        summatrix=new_worker_mat;
    end
    as2=zeros(3,length(req_task));
    for worker=1:length(req_task)
        as2(1,worker)=req_task(1,worker);    %req task- sequence in station
```

```
        as2(2,worker)=w_assign(1,req_task(1,worker)); %assigned worker for
particular task
        as2(3,worker)=summatrix(1,as2(2,worker));

    end
    as2;
    cm=0;
    for i2=1:length(as2(1,:))
        if as2(3,i2)>cm
            sel2=as2(2,i2);
            cm=as2(3,i2);
        end
    end
    for i2=1:length(as2(1,:))
        if worker_mat(as2(1,i2),sel2)==1
            %count=count+1;
            %worker_task(2,count)=sel2;
            as2(2,i2)=sel2;
        end
    end
    ww=length(unique(as2(2,:)));
    no_w=no_w+ww;
    tp=length(nonzeros(worker_task(2,:)));

    worker_task(2,tp+1:tp+length(as2(1,:)))=as2(2,:);
end
worker_task;
no_w;
```

Appendix A3: Crossovers Code

```
%Specify Crossover type
%1=PMX Crossover; 2=Moon Crossover; 3=OX Crossover; 4=RBC-I; 5=RBC-II

if crover_type==1;
%=============CROSSOVER PMX====================================
    pair_sel=randperm(npop);
    p1=[];p2=[];offspring=[];
    for pair=1:npop/2;
        p1=sel_parent(pair_sel(1,pair*2-1),:);
        p2=sel_parent(pair_sel(1,pair*2),:);

        unqran=0;
        while unqran==0
            cp1=randi([2,n-1],1,2);
            if cp1(1)~=cp1(2)
                unqran=1;
            end
        end
        cp1=sort(cp1);
        c1=zeros(1,n);c2=zeros(1,n);
        c1=[p1(1:cp1(1)-1) p2(cp1(1):cp1(2)) p1(cp1(2)+1:end)];
%(cp1(1):cp1(2))=%p2((cp1(1)):cp1(2))
        c2=[p2(1:cp1(1)-1) p1(cp1(1):cp1(2)) p2(cp1(2)+1:end)];

        tempc1=c1;tempc2=c2;
        tempc1(cp1(1):cp1(2))=0;
        tempc2(cp1(1):cp1(2))=0;

        exc=[];
        exc(1,:)=p2(cp1(1):cp1(2));
        exc(2,:)=p1(cp1(1):cp1(2));

        % Check crossover probability
        if rand>pc      %if random number > prob crossover
            c1=p1;
        else            %if random no < prob crossover
            %Regenerate c1
            exc_loop=0;
            while exc_loop==0
                for i=1:n
                    findmatch=[];
                    findmatch=find(exc(1,:)==tempc1(i));
                    is_empty_match=isempty(findmatch);
                    if is_empty_match==0
                        tempc1(i)=exc(2,findmatch);
                    end
                end
                redun_check=ismember(tempc1,exc(1,:));
                if nnz(redun_check)==0
                    exc_loop=1;
                end
            end
            c1=[tempc1(1:cp1(1)-1) exc(1,:) tempc1(cp1(2)+1:end)];
        end

        if rand>pc          %if random number > prob crossover
            c2=p2;
        else
            %Regenerate c2
            exc_loop=0;
            while exc_loop==0
                for i=1:n
                    findmatch=[];
                    findmatch=find(exc(2,:)==tempc2(i));
                    is_empty_match=isempty(findmatch);
                    if is_empty_match==0
```

```
                        tempc2(i)=exc(1,findmatch);
                    end
                end
                redun_check=ismember(tempc2,exc(2,:));
                if nnz(redun_check)==0
                    exc_loop=1;
                end
            end
            c2=[tempc2(1:cp1(1)-1) exc(2,:) tempc2(cp1(2)+1:end)];
        end
        offspring(pair*2-1,:)=c1;
        offspring(pair*2,:)=c2;
    end
    offspring;

elseif crover_type==2
%==================MOON CROSSOVER==============================

    pair_sel=1:npop;
    p1=[];p2=[];offspring=[];
    for pair=1:npop/2;
        Pa=sel_parent(pair_sel(1,pair*2-1),:);
        Pb=sel_parent(pair_sel(1,pair*2),:);
        if rand<=pm
            moon_cross;
            offspring(pair*2-1,:)=child1;
            offspring(pair*2,:)=child2;
        else
            offspring(pair*2-1,:)=Pa;
            offspring(pair*2,:)=Pb;
        end
    end
    offspring;

elseif crover_type==3;
%=============CROSSOVER OX====================================

    pair_sel=randperm(npop);

    %pair_sel=1:npop
    %return
    p1=[];p2=[];offspring=[];
    for pair=1:npop/2;
        p1=sel_parent(pair_sel(1,pair*2-1),:);
        p2=sel_parent(pair_sel(1,pair*2),:);
        ch1=zeros(1,n);ch2=zeros(1,n);
        unqran=0;
        while unqran==0
            cp1=randi([2,n-1],1,2);
            if cp1(1)~=cp1(2)
                unqran=1;
            end
        end
        cp1=sort(cp1);
        %Insert gene from cp1 of p2 into ch1
        ch1(1,cp1(1):cp1(2))=p2(1,cp1(1):cp1(2));
        for i=1:n
            if ch1(1,i)==0
                for j=1:n
                    if ismember(p1(1,j),ch1)==0
                        ch1(1,i)=p1(1,j);
                        break
                    end
                end
            end
        end
        ch1;
        %Process ch2
```

```
        ch2(1,cp1(1):cp1(2))=p1(1,cp1(1):cp1(2));
        for i=1:n
            if ch2(1,i)==0
                for j=1:n
                    if ismember(p2(1,j),ch2)==0
                        ch2(1,i)=p2(1,j);
                        break
                    end
                end
            end
        end
        ch2;
        offspring(pair*2-1,:)=ch1;
        offspring(pair*2,:)=ch2;
    end
    offspring;
%=============RBC-I=====================================
elseif crover_type==4;
    alpha_w=[alpha_pop;1:n];
    [Y,I]=sort(alpha_w(1,:));
    alpha_inv=alpha_w(:,I);
        pair_sel=randperm(npop);
        for i=1:npop
            parent_pop=sel_parent(i,:);
            parent_w=[parent_pop;1:n];
            [D,S]=sort(parent_w(1,:));
            par_inv=parent_w(:,S);
            add_alpha_parent=[1:n;alpha_inv(2,:)+par_inv(2,:)];
            [DD,SS]=sort(add_alpha_parent(2,:));
            new_ospring=add_alpha_parent(:,SS);
            offspring(i,:)=new_ospring(1,:);
        end

%=============RBC-II====================================
elseif crover_type==5;
    w_alpha=0.3;
    alpha_w=[alpha_pop;1:n];
    [Y,I]=sort(alpha_w(1,:));
    alpha_inv=alpha_w(:,I);
    pair_sel=randperm(npop);
    vy=0;
    for pair=1:npop/2;
        p1=sel_parent(pair_sel(1,pair*2-1),:);
        p2=sel_parent(pair_sel(1,pair*2),:);

        p1_w=[p1;1:n];
        [D1,S1]=sort(p1_w(1,:));
        p1_inv=p1_w(:,S1);

        p2_w=[p2;1:n];
        [D2,S2]=sort(p2_w(1,:));
        p2_inv=p2_w(:,S2);

        par_wmat=[p1_inv;p2_inv];

        for j=1:2
            vy=vy+1;
            if j==1
                w_p1=(1-w_alpha)*0.7;
                w_p2=(1-w_alpha)*0.3;
            else
                w_p1=(1-w_alpha)*0.3;
                w_p2=(1-w_alpha)*0.7;
            end

add_prime_w=w_alpha*(alpha_inv(2,:))+w_p1*(p1_inv(2,:))+w_p2*(p2_inv(2,:));
            apr=[1:n;add_prime_w];
            [DD,SS]=sort(apr(2,:));
```

```
            new_ospring=apr(:,SS);
            offspring(vy,:)=new_ospring(1,:);

        end

    end
end
```

APPENDIX B
CASE STUDY OPTIMISATION RESULTS

Appendix B1 : Case study optimisation results using PMX crossover

	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
Fitness	2.614286	2.4	2.4	2.6	2.4	2.4	2.4	2.4	2.4	2.6
Assembly Sequence	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2
	4	3	3	3	4	6	3	3	3	6
	5	4	4	4	5	4	4	4	4	7
	20	5	5	5	8	7	5	5	5	11
	6	6	8	8	20	5	20	6	8	13
	8	11	6	20	6	8	8	11	20	15
	11	20	7	6	22	3	6	20	6	4
	22	22	20	11	11	20	11	22	11	5
	7	8	22	22	7	22	22	8	22	20
	13	7	11	7	9	9	7	7	7	22
	15	13	13	9	16	16	9	9	9	8
	3	15	15	16	19	19	10	14	13	3
	9	9	9	19	14	14	14	16	15	9
	16	14	16	13	17	17	13	19	10	14
	19	16	19	15	21	18	17	13	16	16
	14	19	14	10	23	10	18	17	19	19
	17	10	17	12	18	12	16	21	14	17
	18	17	18	14	13	21	19	23	17	18
	21	18	10	17	3	23	12	18	18	10
	23	12	21	21	10	11	21	10	12	21
	10	21	23	23	12	13	23	12	21	23
	12	23	12	18	15	15	15	15	23	12

Appendix B2: Case study optimisation results using Moon crossover

Fitness	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
	2.4	2.6	2.6	2.6	2.4	2.4	2.4	2.4	2.4	2.6
Assembly Sequence	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2
	6	6	3	3	6	3	6	3	6	3
	4	4	4	4	7	4	11	4	4	4
	5	5	5	5	11	5	7	5	5	5
	8	20	8	8	9	8	13	20	20	8
	3	8	20	20	3	20	3	8	8	20
	11	3	6	6	4	6	4	6	3	6
	7	7	11	7	5	11	5	11	22	11
	20	9	22	11	8	22	8	22	7	22
	22	16	7	13	16	7	9	7	11	7
	9	19	13	15	13	9	10	9	9	9
	16	11	15	22	15	13	16	10	14	16
	19	13	9	9	14	14	19	12	16	19
	10	15	10	16	17	17	14	16	19	13
	12	10	12	19	18	18	12	19	13	15
	13	12	16	14	19	10	15	13	17	10
	15	22	19	17	20	12	17	14	21	12
	14	14	14	18	22	16	18	17	23	14
	17	17	17	10	10	19	20	18	18	17
	21	21	21	12	12	21	22	15	10	21
	23	18	23	21	21	23	21	21	12	18
	18	23	18	23	23	15	23	23	15	23

Appendix B3: Case study optimisation results using OX crossover

	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
Fitness	2	2.6	2.6	2.414286	2.4	2.6	2.8	2.4	2.2	2.4
Assembly Sequence	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2
	6	3	6	6	6	6	6	6	3	6
	4	4	11	4	4	7	4	11	4	7
	5	5	7	5	5	11	7	7	5	11
	8	6	13	8	20	4	5	13	20	4
	7	8	15	3	3	5	20	3	8	5
	9	20	4	20	8	8	11	4	6	20
	3	22	5	22	22	3	13	5	7	8
	16	11	20	11	7	20	15	8	11	3
	19	7	22	7	9	22	9	9	9	9
	14	13	8	9	14	9	14	10	16	16
	17	15	3	13	16	16	10	16	19	22
	11	9	9	16	19	19	17	19	10	19
	13	10	16	19	10	10	3	14	22	10
	15	12	19	10	17	12	18	17	13	14
	18	16	14	15	18	14	12	18	15	17
	20	19	17	14	21	13	21	12	14	18
	22	14	18	17	23	15	22	15	17	13
	10	17	10	18	12	17	23	21	18	15
	12	21	12	21	11	21	8	20	12	21
	21	23	21	23	13	23	16	22	21	23
	23	18	23	12	15	18	19	23	23	12

Appendix B4 : Case study optimisation results using RBC-I crossover

Fitness	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
	2.4	2.014286	2.2	2.2	2.6	2.6	2.214286	2	2.4	2.6
Assembly Sequence	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2
	4	4	3	4	6	6	4	4	4	3
	5	5	4	5	11	4	5	5	5	4
	3	8	5	20	7	5	8	3	8	5
	8	3	8	8	13	11	6	6	3	8
	6	6	6	6	15	20	3	7	6	20
	7	7	20	22	4	3	7	20	7	6
	20	9	11	11	5	22	9	22	20	7
	22	16	22	7	20	8	16	8	11	22
	11	19	7	13	8	7	19	9	13	11
	9	14	9	15	3	9	20	14	15	13
	14	10	13	3	22	16	11	16	22	15
	17	20	16	9	9	14	14	19	9	9
	21	22	19	16	16	17	17	10	16	10
	23	17	10	19	19	21	21	17	14	12
	18	18	14	14	14	18	18	18	17	16
	10	11	15	17	17	23	13	21	21	19
	16	13	17	18	18	19	15	23	23	14
	19	15	18	10	10	10	10	12	18	17
	12	12	12	12	12	12	12	11	19	21
	13	21	21	21	21	13	22	13	10	23
	15	23	23	23	23	15	23	15	12	18

Appendix B5 : Case study optimisation results using RBC-I crossover

Fitness	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
	2.2	2.2	2	2.414286	2.4	2.4	2	2.2	2.2	2
Assembly Sequence	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2
	6	3	6	3	3	4	3	3	4	4
	4	4	4	4	4	5	4	4	5	5
	5	5	5	5	5	3	5	5	8	3
	8	8	20	20	20	8	6	8	6	6
	3	6	11	8	8	6	8	6	3	7
	11	20	3	6	6	7	11	11	7	20
	7	11	8	7	22	20	20	7	20	22
	20	7	7	9	11	22	7	20	22	8
	22	9	9	16	7	11	9	22	11	9
	9	10	16	19	9	9	16	9	13	16
	14	14	19	10	10	14	19	16	15	19
	16	16	22	11	13	17	22	19	9	14
	19	19	13	13	16	21	13	14	16	17
	10	17	15	15	19	23	14	17	19	18
	17	21	14	14	12	18	15	21	14	10
	18	13	17	17	15	10	17	23	17	12
	21	15	18	21	14	16	18	13	18	21
	23	18	10	18	17	19	10	18	10	23
	13	12	12	12	21	12	12	10	12	11
	15	22	21	22	18	13	21	12	21	13
	12	23	23	23	23	15	23	15	23	15

APPENDIX C
SIMULATION OUTPUT

Appendix C1: Simulation Output for Existing Layout

Name	% Idle	% Busy	% Blocked	No. Of Operations	Name	Part_A
RUN 1						
Station1	0.37	95.25	4.39	233	No. Entered	234
Station2	1.48	93.62	4.89	232	No. Shipped	226
Station3	1.21	96.94	1.85	231	No. Scrapped	0
Station4	4.15	89.96	5.89	230	No. Assembled	0
Station5	2.24	97.75	0.01	229	No. Rejected	28567
Station6	16.51	83.49	0	228	W.I.P.	8
Station7	53.33	46.67	0	228	Avg W.I.P.	7.86
Station8	27.18	72.82	0	227	Avg Time	966.88
Station9	34.53	65.47	0	226	Sigma Rating	6
Station10	73.42	26.58	0	226		
RUN 2						
Station1	0.41	95.67	3.92	234	No. Entered	235
Station2	1.78	93.58	4.63	233	No. Shipped	227
Station3	1.22	97.4	1.39	232	No. Scrapped	0
Station4	4.5	90.07	5.42	231	No. Assembled	0
Station5	2.4	97.6	0	230	No. Rejected	28566
Station6	16.87	83.13	0	229	W.I.P.	8
Station7	53.13	46.87	0	228	Avg W.I.P.	7.85
Station8	26.84	73.16	0	228	Avg Time	962.2
Station9	34.23	65.77	0	227	Sigma Rating	6
Station10	73.49	26.51	0	227		
RUN 3						
Station1	0.42	94.74	4.84	232	No. Entered	233
Station2	1.5	93.4	5.1	231	No. Shipped	225
Station3	1.21	96.62	2.17	230	No. Scrapped	0

Appendix C1: Continued

Name	% Idle	% Busy	% Blocked	No. Of Operations	Name	Part_ A
Station4	3.59	89.49	6.91	229	No. Assembled	0
Station5	2.26	97.71	0.03	228	No. Rejected	28568
Station6	16.96	83.04	0	227	W.I.P.	8
Station7	53.58	46.42	0	227	Avg W.I.P.	7.85
Station8	27.62	72.38	0	226	Avg Time	969.72
Station9	34.87	65.13	0	225	Sigma Rating	6
Station10	73.46	26.54	0	225		
RUN 4						
Station1	0.41	94.36	5.23	233	No. Entered	233
Station2	1.44	93.67	4.89	232	No. Shipped	225
Station3	1.05	96.9	2.06	231	No. Scrapped	0
Station4	3.85	89.5	6.65	230	No. Assembled	0
Station5	2.34	97.65	0.01	228	No. Rejected	28568
Station6	16.68	83.32	0	228	W.I.P.	8
Station7	53.2	46.8	0	227	Avg W.I.P.	7.86
Station8	27.17	72.83	0	226	Avg Time	971.38
Station9	34.56	65.44	0	226	Sigma Rating	6
Station10	73.43	26.57	0	225		
RUN 5						
Station1	0.39	94.7	4.9	232	No. Entered	233
Station2	1.34	93.3	5.36	231	No. Shipped	225
Station3	1.14	96.71	2.16	230	No. Scrapped	0
Station4	3.76	89.51	6.74	229	No. Assembled	0
Station5	2.19	97.81	0	228	No. Rejected	28568
Station6	17.03	82.97	0	227	W.I.P.	8
Station7	53.3	46.7	0	227	Avg W.I.P.	7.85
Station8	27.55	72.45	0	226	Avg Time	970.36
Station9	34.57	65.43	0	225	Sigma Rating	6
Station10	73.68	26.32	0	225		

Appendix C1: Continued

Name	% Idle	% Busy	% Blocked	No. Of Operations	Name	Part_ A
RUN 6						
Station1	0.39	96.2	3.4	232	No. Entered	233
Station2	2.42	93.39	4.19	231	No. Shipped	225
Station3	1.62	96.7	1.67	230	No. Scrapped	0
Station4	4.67	89.33	6	229	No. Assembled	0
Station5	2.49	97.5	0.01	228	No. Rejected	28568
Station6	17.23	82.77	0	227	W.I.P.	8
Station7	53.58	46.42	0	227	Avg W.I.P.	7.81
Station8	27.54	72.46	0	226	Avg Time	965.38
Station9	35.31	64.69	0	225	Sigma Rating	6
Station10	73.72	26.28	0	225		
RUN 7						
Station1	0.41	95.48	4.11	232	No. Entered	233
Station2	1.73	93.34	4.92	231	No. Shipped	225
Station3	1.3	96.68	2.02	230	No. Scrapped	0
Station4	4.35	89.63	6.02	229	No. Assembled	0
Station5	2.36	97.58	0.06	228	No. Rejected	28568
Station6	17.34	82.66	0	227	W.I.P.	8
Station7	53.27	46.73	0	227	Avg W.I.P.	7.83
Station8	27.82	72.18	0	226	Avg Time	968.27
Station9	34.51	65.49	0	225	Sigma Rating	6
Station10	73.55	26.45	0	225		
RUN 8						
Station1	0.43	95.24	4.34	233	No. Entered	234
Station2	1.6	93.66	4.75	232	No. Shipped	226
Station3	1.27	96.95	1.79	231	No. Scrapped	0
Station4	4.16	89.65	6.19	230	No. Assembled	0
Station5	2.51	97.46	0.02	229	No. Rejected	28567
Station6	16.89	83.11	0	228	W.I.P.	8

Appendix C1: Continued

Name	% Idle	% Busy	% Blocked	No. Of Operations	Name	Part_ A
Station7	53.36	46.64	0	227	Avg W.I.P.	7.84
Station8	27.48	72.52	0	227	Avg Time	965.06
Station9	34.74	65.26	0	226	Sigma Rating	6
Station10	73.46	26.54	0	226		
RUN 9						
Station1	0.4	95.3	4.3	232	No. Entered	233
Station2	1.58	93.25	5.17	231	No. Shipped	225
Station3	1.22	96.69	2.09	230	No. Scrapped	0
Station4	3.78	89.55	6.67	229	No. Assembled	0
Station5	2.15	97.84	0.01	228	No. Rejected	28568
Station6	16.87	83.13	0	227	W.I.P.	8
Station7	53.4	46.6	0	227	Avg W.I.P.	7.85
Station8	26.97	73.03	0	226	Avg Time	970.51
Station9	34.62	65.38	0	225	Sigma Rating	6
Station10	73.83	26.17	0	225		
RUN 10						
Station1	0.41	95.51	4.08	234	No. Entered	235
Station2	1.66	93.81	4.53	233	No. Shipped	227
Station3	1.13	97.33	1.54	232	No. Scrapped	0
Station4	4.06	89.92	6.02	231	No. Assembled	0
Station5	2.28	97.71	0.01	230	No. Rejected	28566
Station6	16.4	83.6	0	229	W.I.P.	8
Station7	52.96	47.04	0	228	Avg W.I.P.	7.86
Station8	27.05	72.95	0	228	Avg Time	963.87
Station9	34.13	65.87	0	227	Sigma Rating	6
Station10	73.43	26.57	0	227		

Appendix C2: Simulation Output for Optimised Layout

Name	% Idle	% Busy	% Blocked	No. Of Operations	Name	Part_ A
RUN 1						
Station1	0.4	97.18	2.42	236	No. Entered	237
Station2	3.02	95	1.98	235	No. Shipped	229
Station3	9.69	67.7	22.61	235	No. Scrapped	0
Station4	1.55	94.97	3.49	233	No. Assembled	0
Station5	3.84	96.16	0	232	No. Rejected	28564
Station6	36.78	62.73	0.49	232	W.I.P.	8
Station7	14.22	84.12	1.66	231	Avg W.I.P.	8.15
Station8	9.32	86.63	4.05	230	Avg Time	990.29
Station9	6.26	93.74	0	229	Sigma Rating	6
RUN 2						
Station1	0.4	97.64	1.96	237	No. Entered	238
Station2	3.11	95.55	1.34	236	No. Shipped	230
Station3	10.48	68.68	20.84	235	No. Scrapped	0
Station4	1.51	95.33	3.16	234	No. Assembled	0
Station5	4.42	95.58	0	233	No. Rejected	28563
Station6	35.97	62.95	1.08	232	W.I.P.	8
Station7	12.79	84.61	2.6	232	Avg W.I.P.	8.17
Station8	8.25	87.49	4.26	231	Avg Time	988.83
Station9	5.91	94.09	0	230	Sigma Rating	6
RUN 3						
Station1	0.4	97.52	2.08	237	No. Entered	238
Station2	3.3	95.36	1.34	236	No. Shipped	229
Station3	8.47	68.82	22.7	235	No. Scrapped	0
Station4	1.3	95.15	3.55	234	No. Assembled	0
Station5	3.67	96.33	0	233	No. Rejected	28563
Station6	37.03	62.69	0.28	232	W.I.P.	9
Station7	14.4	84.16	1.44	231	Avg W.I.P.	8.16
Station8	9.06	86.94	4	230	Avg Time	988.02
Station9	5.88	94.12	0	229	Sigma Rating	6
RUN 4						
Station1	0.41	96.85	2.73	238	No. Entered	239
Station2	2.78	95.87	1.35	237	No. Shipped	230
Station3	8.91	68.98	22.11	236	No. Scrapped	0
Station4	1.18	95.26	3.56	235	No. Assembled	0
Station5	3.87	96.13	0	234	No. Rejected	28562
Station6	36.2	63.43	0.37	233	W.I.P.	9
Station7	12.96	85	2.05	233	Avg W.I.P.	8.2

Appendix C2: Continued

Name	% Idle	% Busy	% Blocked	No. Of Operations	Name	Part_ A
Station8	7.89	87.52	4.59	232	Avg Time	988.58
Station9	5.42	94.58	0	230	Sigma Rating	6
RUN 5						
Station1	0.43	97.33	2.24	237	No. Entered	238
Station2	3.22	95.57	1.21	236	No. Shipped	229
Station3	10.64	67.96	21.4	235	No. Scrapped	0
Station4	1.46	95.13	3.42	234	No. Assembled	0
Station5	4.14	95.83	0.03	233	No. Rejected	28563
Station6	34.61	62.86	2.53	232	W.I.P.	9
Station7	12.59	84.5	2.91	231	Avg W.I.P.	8.2
Station8	7.96	87.35	4.68	231	Avg Time	991.72
Station9	5.4	94.6	0	229	Sigma Rating	6
RUN 6						
Station1	0.4	96.31	3.29	236	No. Entered	237
Station2	2.42	94.98	2.6	235	No. Shipped	228
Station3	5.66	68.24	26.1	234	No. Scrapped	0
Station4	1.16	95.11	3.73	233	No. Assembled	0
Station5	3.82	96.18	0	232	No. Rejected	28564
Station6	36.61	63.18	0.21	231	W.I.P.	9
Station7	14.4	84.24	1.36	230	Avg W.I.P.	8.2
Station8	9.11	87.71	3.18	230	Avg Time	996.39
Station9	6.48	93.52	0	228	Sigma Rating	6
RUN 7						
Station1	0.41	96.34	3.24	236	No. Entered	237
Station2	2.62	94.93	2.46	235	No. Shipped	228
Station3	5.46	68.17	26.37	234	No. Scrapped	0
Station4	1.22	95.59	3.19	233	No. Assembled	0
Station5	4.2	95.8	0	232	No. Rejected	28564
Station6	37.29	62.31	0.41	231	W.I.P.	9
Station7	14.97	83.52	1.51	230	Avg W.I.P.	8.18
Station8	9.6	86.75	3.66	229	Avg Time	993.81
Station9	6.42	93.58	0	228	Sigma Rating	6
RUN 8						
Station1	0.39	97.29	2.32	238	No. Entered	239
Station2	2.55	95.76	1.68	237	No. Shipped	230
Station3	7.69	68.69	23.62	236	No. Scrapped	0
Station4	1.3	95.74	2.96	235	No. Assembled	0
Station5	4.02	95.98	0	234	No. Rejected	28562

Appendix C2: Continued

Name	% Idle	% Busy	% Blocked	No. Of Operations	Name	Part_ A
Station6	36.5	62.99	0.51	233	W.I.P.	9
Station7	13.35	84.49	2.16	232	Avg W.I.P.	8.2
Station8	8.18	87.22	4.6	231	Avg Time	988.62
Station9	5.58	94.42	0	230	Sigma Rating	6
RUN 9						
Station1	0.41	96.69	2.9	236	No. Entered	237
Station2	2.87	95.04	2.09	235	No. Shipped	229
Station3	6.8	68.83	24.37	234	No. Scrapped	0
Station4	1.24	95.61	3.15	233	No. Assembled	0
Station5	4.19	95.81	0	232	No. Rejected	28564
Station6	36.74	62.83	0.43	232	W.I.P.	8
Station7	13.87	84.1	2.03	231	Avg W.I.P.	8.19
Station8	8.66	87.37	3.97	230	Avg Time	995.61
Station9	5.92	94.08	0	229	Sigma Rating	6
RUN 10						
Station1	0.41	97.81	1.78	237	No. Entered	238
Station2	3.51	95.19	1.3	236	No. Shipped	229
Station3	9.01	68.65	22.34	235	No. Scrapped	0
Station4	1.28	95.35	3.37	234	No. Assembled	0
Station5	4.31	95.69	0	233	No. Rejected	28563
Station6	35.86	63.37	0.77	232	W.I.P.	9
Station7	12.84	84.43	2.73	231	Avg W.I.P.	8.19
Station8	7.69	87.3	5.01	230	Avg Time	991.4
Station9	5.81	94.19	0	229	Sigma Rating	6