MODIFIED NEH HEURISTIC ON MAKESPAN REDUCTION IN PERMUTATION FLOW SHOP PROBLEMS

by

CHONG ZHENG ZIAO

Thesis submitted in fulfilment of the requirements

for the degree of

Master of Science

July 2015

ACKNOWLEDGEMENTS

I have taken efforts throughout this duration. However, it would not have been possible without the kind support and help of many individuals and organizations. Firstly, I would like to extend my sincere thanks to my family and friends for all their support and encouragement.

Besides that, I would like to thank my supervisors, Dr Shahrul bin Kamaruddin and Co-supervisor Professor Dr Ishak bin Haji Abdul Azid for their professional supervision and advice throughout my master research. Without their advice and assistance it would be a lot tougher to complete this project. I also sincerely thanks for the time they spent in listening my problems and suggested the ideas which benefited to my research.

Next, my sincere thanks go to all the staff of the School of Mechanical Engineering and who helped me in many ways and made my research journey in USM pleasant and unforgettable. Then, I wish to acknowledge the support of the CREST grant scheme from the Government of Malaysia for funding this research.

Lastly, I would like to thanks any person who contributed to my research journey directly or indirectly. I would like to acknowledge their comments and suggestions, which was crucial for the successful completion of this project.

TABLE OF CONTENTS

CHAPTER 2 LITERATURE REVIEW

CHAPTER 3 METHODOLOGY

CHAPTER 4 MODEL DEVELOPMENT

APPENDIXES

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
AED	Average Error Deviation
B & B	Branch and bound
CDS	Campbell, Dudek and Smith
ED	Error Deviation
FIFO	First In First Out
FL	Framinan and Leisten
FSSP	Flow Shop Scheduling Problem
GA	Genetic algorithm
JIT	Just In Time
KK	Kalczynski and Kamburowski
LPT	Longest Processing Time
NEH D	NEH heuristic based on deviation
NEH	Nawaz, Enscore and Ham
NEH-M	Modified NEH
NM	Nagano and Moccellin
PFSP	Permutation Flow Shop Problem

PWB	Printed Wiring Board
RA	Rapid Access
SA	Simulated Annealing
SMT	Surface Mounting Technology
SPT	Shortest Processing Time
TS	Tabu Search
WIP	Work In Process
WY	Woo and Yim

LIST OF SYMBOLS

Symbol	Description	Unit
$\sum C_j$	Total completion time	Hour
$\sum T_j$	Total tardiness	Hour
$\sum U_j$	Total number of tardy job	-
$\sum w_j C_j$	Total weighted completion time	Hour
C_{max}	Makespan	Hour
F_m	Flow shop	-
J_m	Job shop	-
L_{max}	Lateness	Hour
O_m	Open shop	-
P_m	Parallel machine	-
Q_m	Related machines	-
R_m	Unrelated machines	-
$T\ In_{i,j}$	Time of job *j* load into machine *i*	Hour
$T\ Out_{i,j}$	Time of job *j* move out from machine *i*	Hour
d_j	Due date	-
$p_{i,j}$	Processing time of job *j* on machine *i*	Hour
r_j	Release date	-
s_{jk}	Sequence dependent setup time	Hour
w_j	Weight of job	-

1	Single machine	-
block	Blocking	-
γ	Objective functions	-
i	Machine	-
j	Job	-
m	Number of machines	-
n	Number of jobs	-
prec	Precedence constraints	-
prmp	Preemptions	-
prmu	Permutation	-
rcrc	Recirculation	-
α	Machine environment	-
β	Processing characteristic and constraints	-

PENGUBAHSUAIAN HEURISTIK NEH UNTUK MENGURANGKAN MASA SIAPAN DALAM MASALAH PERMUTATION FLOW SHOP

ABSTRAK

Masalah permutation flow shop (PFSP) merupakan salah satu persekitaran mesin yang biasa dikaji dalam masalah penjadualan. Dalam PFSP, susunan setiap proses dalam semua mesin tidak berubah. Beberapa algoritma telah dicadangkan untuk menentukan susunan kerja dan mesin untuk mengurangkan masa siapan di PFSP. Sepanjang 30 tahun yang lalu, heuristik NEH yang dicadang oleh Nawaz , Enscore dan Ham telah dianggap sebagai heuristik yang terbaik untuk meminimumkan masa siapan di PFSP. Oleh kerana penemuan ini, NEH heuristik dipilih sebagai asas kajian ini untuk meminimumkan masa siapan dan masa terbiar dalam PFSP. Pengubahsuaian dilakukan untuk meningkatkan prestasi dalam pengurangan masa siapan dan masa terbiar. Dalam kajian ini, sebanyak 109 masalah telah diselesaikan dengan bilangan mesin dan pekerjaan yang ditetapkan dalam bilangan 4 hingga 25. 100 masalah telah dilakukan dalam penilaian berangka. Masa proses pekerjaan dijana secara rawak dalam 1 hingga 10 jam dengan menggunakan excel spreadsheet. Manakala yang baki 9 set ujian telah dijalankan dalam kajian kes industri. Dalam kajian kes ini, syarikat yang terlibat merupakan sebuah syarikat yang menyediakan perkhidmatan surface mounting technology (SMT). Ia merancang jadual dengan mengguna teknik backward scheduling. Heuristik yang dicadangkan, iaitu heuristik NEH-M akan dibandingkan dengan jadual yang disedia oleh syarikat dan jadual NEH untuk mengesahkan idea yang dicadangkan. Prestasi heuristik telah dikira dengan menggunakan error deviation (ED) formula. Keputusan yang dihasilkan oleh

Excel menunjukkan bahawa prestasi heuristic NEH-M adalah lebih baik daripada prestasi jadual yang disedia oleh syarikat. Sebaliknya, apabila heuristic NEH-M dibandingkan dengan heuristic NEH, prestasi keseluruhan pengurangan masa siapan adalah tidak baik apabila nombor mesin dan pekerjaan semakin besar, manakala prestasi keseluruahan dalam mengurangkan masa terbiar adalah baik.

MODIFIED NEH HEURISTIC ON MAKESPAN REDUCTION IN PERMUTATION FLOW SHOP PROBLEMS

ABSTRACT

Permutation flow shop problem (PFSP) is one of the commonly reviewed machine environments in scheduling problems. The order sequence for each process remains unchanged for all machines. Few algorithms have been developed to decide the sequence of *n* jobs and *m* machines that can minimize makespan in flow shops. Throughout the past 30 years, the NEH heuristics developed by Nawaz, Enscore and Ham has been commonly regarded as the best heuristic for minimizing the makespan in permutation flow shops. Due to these findings, NEH heuristics is selected as the basis of this study. Modification is done to enhance the objectives of this study, which is makespan and idle time reduction. In this study, a total of 109 flow-shop problems were solved with the number of machines and jobs being set at a range of 4 to 25. 100 problems were carried out using numerical assessments. The process times of the jobs were randomly generated within the range of 1 to 10 using Excel spreadsheets. Whereas the remaining 9 sets of tests were carried out using real world case studies. In each case study, the company involved was provided with a surface mounting technology (SMT) service. It has the capability of planning schedules by adopting the backward scheduling technique. The proposed heuristic, NEH-M will be compared to both the historical production schedule and NEH schedule in order to verify and validate the performance of the proposed idea. The performance of the NEH-M heuristics was computed using the error deviation (ED) formula. The generated results gained through Excel modeling show that the NEH-M heuristics

outperforms the historical production schedule in all conditions. On the other hand, when the NEH-M heuristics is compared to the NEH heuristics, the overall performance of makespan reduction is underperforming while the overall performance of idle time reduction is over performing when there are large numbers of machines and jobs.

CHAPTER 1

INTRODUCTION

This chapter gives an overview of scheduling and focuses on permutation flow shop problem (PFSP). It also presents the problems that occur in permutation flow shop problem and gives the idea of the heuristic selected as the basis of model development to solve these problems. Then it summarizes the main objectives of this research. This chapter ends with the structure of the thesis.

1.1 Overview

Scheduling has been a famous topic studied by researchers in the manufacturing industry. The main concern of scheduling problem is to identify the sequence and control of the manufacturing operations of the jobs on machines to achieve one or more objective functions as according to Hejazi and Saghafian (Reza Hejazi and Saghafian, 2005). It can also be defined as a process that makes the decision of determining the sequences of jobs relevant to a repeated basic that is focused on the arrangement of resources attached to the activities with the objectives of optimizing one or more performance measures. The phrase "resources" here may refer to jobs in an assembly plant, whereas "activities" may be operations in manufacturing processes. Manufacturing processes refer to fabrication steps to transform raw materials into a final product. Modify the material into the required part involves machines. The concern of scheduling is to operate the machines at its maximum capability, no process shall be delayed for longer time and to complete the entire processes in the shortest time (Kishor and Goyal, 2013). In other words, the main

concern of scheduling is to reduce the completion time of all activities, avoid the tardiness of activities that cannot be completed before the due date, complete the most important activities on time and maximize the number of activities completed in a fixed period.

Depends on the situation, the objectives of scheduling can vary from one to another. One objective may be on minimizing the makespan, while another objective may be on reducing the number of late jobs. A schedule that is able to fulfill its objectives is referred as good scheduling. A good scheduling algorithm can lower the manufacturing cost in manufacturing process and simultaneously enable the company to stay competitive. At the same time, it allows assorted jobs to be accomplished systematically, and also able to prevent resources conflict.

For different classes of scheduling problems, the approaches developed are quite different. Graham (Graham et al., 1979) proposed a three-field notation ($\alpha \mid \beta \mid \gamma$) for scheduling problems classification. α refers to machine environment, β refers to processing characteristic and constraints, and γ refers to objective functions. Machine environment can be classified into single machine (1), open shop (O_m), parallel machine (P_m), job shop (J_m) and flow shop (F_m). Machine environment refers to the arrangement of the machines in the shop floor and how the jobs are passed from one machine to the other.

The simplest of all possible machine environment is the single machine, and it contains only one machine. In this case, the machine environment consists of only one machine, and at a time, the machine can process only one job (Li et al., 2011). Sometimes, processes have to undergo the operation of all jobs according to the same sequence, which implies that the jobs have to follow the identical path. This type of machine environment is names as flow shop scheduling, where the machines are

arranged in series (Pinedo, 2008). However, job-shop problem is different with the flow shop problem, each of the jobs will follow its particular flow pattern. With the job shop problem, the jobs undergo its operation on any machine for only once, unless it is designed to allow for more than one visits to machines (Kleeman and Lamont, 2007). Lastly is the open shop. Open shop involves *m* machines and there are *n* jobs. A job can be handled by maximum one process at a time and maximum by one machine at a time. A process can be done in any order as long as it is the same job (Brucker et al., 1993). Based on the paper written by Haibo and Chunming (Haibo and Chunming, 2010), it is stated that flow shop scheduling problems (FSSP) is a commonly reviewed machine environment in scheduling problems. This sub-class has taken up about 25% of assembly lines, manufacturing system and information service facilities.

Generally, the well-known methods to solve scheduling problems can be categorized into heuristics, simulation and optimization methods. This research focuses on heuristic method. The reasons of selecting heuristic method in solving the problem are discussed in detail in Chapter 2. Heuristics method is built up by two categories, which are improvement heuristics and constructive heuristics. Constructive heuristic builds up a solution piece by piece. It determines the ordered sequence of jobs and works by constructing a solution step by step. This heuristic approach has done according to some specific conditions or decisions to come out with a feasible solution and improvement by adopting intelligent search techniques to look for a high-quality solution for given specified objectives. Constructive heuristics are able to provide good quality results but may not achieve the best possible solution (Burke et al., 2007). For constructive heuristics, there are few heuristics that have been introduced in relatively early decades, for example, the heuristics by

Nawaz, Enscore and Ham, denoted by NEH (Nawaz et al., 1983) and the pioneer's work of Johnson (Johnson, 1954).

The second category is improvement heuristics. It takes a solution and it is then changed to form of the structure to find a better solution. By adopting specific rules, improvement usually starts from an current feasible solution that is generally proposed by one of the existing heuristics and comes out with a better solution (Ancău, 2012). A better solution is usually obtained by exploring the neighborhood (Omatu, 2014). Improvement heuristics are mainly metaheuristics, such as simulated annealing (SA), genetic algorithm (GA) and tabu search (TS) algorithm. All the listed heuristics are designed for minimizing makespan.

Improvement methods are not recommended to solve large-scale problems as they are quite time consuming. Conversely, constructive heuristics are mainly simple heuristics. Certain strategies or priority rules are usually adopted in constructive heuristics. Therefore, enhancement can also be expected when such approaches or directions are used in metaheuristics. Due to this reason, the strategies and priority rules are commendable to be explored (Dong et al., 2006). In the paper of Solimanpur et al. (Solimanpur et al., 2004), the author stated that as the flow shop scheduling problem has been identified as NP-hard, the branch and bound method is not recommended for solving large size problems. Due to this constraint, it has provide the researchers proposing heuristics development. The summary of differences between constructive heuristics and improvement heuristics are shown in Table 1. 1**.**

Table 1. 1: Comparisons of Constructive and Improvement Heuristics

Constructive Heuristics	Improvement Heuristics
Build up a solution piece by piece.	Generated from a solution and change to form of the structure to find a better solution.
Solution may be *local*.	Solution may be *optimum*.
Able to develop solution in short period.	Time consuming.
Heuristics Johnson's Rule and Nawaz, Enscore and Ham heuristic (denoted by NEH)	Mainly metaheuristics, such as simulated annealing (SA), genetic algorithm (GA)and tabu search (TS) algorithm.

1.2 Problem Statement

One of the active studied topics in the literature of scheduling is the permutation flow shop problem (PFSP) (Ruiz and Stützle, 2007). It refers to the determination of the order of *n* jobs on *m* machines while all jobs have similar machine sequence. In the middle of the desired objectives, minimization of makespan has gained the most thoughtfulness (Ruiz and Maroto, 2005). With the same processing time of jobs in each machine, different jobs arrangement will generate different makespan. Therefore, the method used in planning a production schedule plays an important rule.

In real world industry, backward scheduling is usually adopted in planning the schedule. Backward scheduling starts at the requirement date. This means that based on the committed shipment date that is agreed by the customer, the production will schedule backwards to determine the time required for each operation (Childe, 1996). The scheduling technique is adopted as it is able to receive the ordered material in the latest time possible. This helps in reducing the holding cost in inventory. However, if there is any delay in the production, the planned schedule will finish late (Summers, 1998). Unfortunately, if one is concerned with makespan reduction or increasing of utilization, backward scheduling is not a good choice.

The aim of this research is to reduce the makespan and idle time of a PFSP with a new proposed algorithm. The algorithm is developed based on a selected superior solution. The superior solution is selected by referring to literature findings. New rules are proposed in generating the initial job sequence. To maintain the job insertion phase, the number of enumeration will still be the same. The algorithm complexity remains unchanged as the original algorithm. Problems that are able to generate a shorter makespan and idle time are considered as "performing better" than the selected algorithm.

1.3 Research Objectives

With strong engineering background, flow shop scheduling problem (FSSP) is a commonly researched topic, where permutation flow shop problem (PFSP) is one of its subclasses. For FSSP with m machines and n jobs, there are total of $(n!)^m$ different ways to allocate the jobs on machines. However, in PFSP, the solutions is reduced to $n!$. The entire research deals with PFSP. The objectives of this research are:

i. To study the criteria of permutation flow shop problem.

ii. To propose and develop an enhanced heuristic based on NEH algorithm for makespan and idle time reduction.

iii. To validate and verify the effectiveness of the proposed heuristic through a case study.

1.4 Scopes of the Work

This research developed a new model for makespan and idle time reduction. By taking the surface mounting technology (SMT) as the background of the case study, the model is developed according to this machine environment, which is PFSP. Numerical experiments are carried out with different number of machines and jobs. For data verification, the results produced by the proposed algorithm are compared with the results generated by the existing algorithm. Meanwhile, for data validation, the schedule of the case study company is compared with the schedule generated by the proposed algorithm. The information extracted from the shop floor include sequence of job, processing time and amount of job run in each historical schedule. As shorter makespsan and idle time is computed, the algorithm performs better. Evaluation of the proposed algorithm is done based on different number of jobs and machines.

1.5 Structure of Thesis

This thesis is presented in 6 chapters, started with the introduction, followed by literature review, then the development of the proposed heuristic algorithm, validation and verification, results and discussion, and lastly the conclusions and future work.

Chapter 1: Introduction

- The background, objectives and scopes of the research are introduced.

Chapter 2: Literature Review

- Literature findings of permutation flow shop problem (PFSP) and proposed solutions.

Chapter 3: Methodology

- The methodology of numerical assessment is proposed, and the theory of NEH heuristic is stated in this chapter in the procedural steps.

Chapter 4: Model Development

- The heuristics for PFSP is developed to minimize makespan and idle time. A case study is carried out in the semi-conductors industry by adopting the enhanced heuristic.

Chapter 5: Verification and Validation of the NEH-M Heuristic

- Validation and verification is done by comparing the results of numerical tests, as well as comparing the results of case study schedule. Besides that, the effectiveness of the proposed algorithm is studied. The trends of the results are also discussed.

Chapter 6: Conclusions

- Conclusions and direction of future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter presented current studies and also the latest findings related to this research. The basic definitions and concepts applied in this research are introduced. This chapter started by giving an idea of general definitions and notations used throughout this research. It is then followed by the classification of approaches and development of each method. The next section focuses on the review of flow shop scheduling problems (FSSP). Then, the chapter provides an overview of previous researches on permutation flow shop problems (PFSP) for makespan reduction. Finally, a summary of this chapter is given in the final section.

2.2 General Definitions and Notations

This part presented the definitions of the terms and notations that are commonly used in scheduling problems. With the notation, one can easily differentiate the operational routine of the product with its process constraints and objective functions to achieve.

2.2.1 Overview

A process that makes the decision of determining the sequences of jobs relevant to a repeated basis in many manufacturing processes is defined as “scheduling” (Pinedo, 2012). It is a process that decide how to allocate the resources attached to the

activities with the objectives of optimizing the performance measures. "Resources" here may refer to jobs in a manufacturing line. Activities may include machines in the shop floor. The objective of scheduling may vary from one to another. A good scheduling algorithm can lower the production cost in the manufacturing process and enable the company to predict the completion time of a fixed amount of jobs.

In most scheduling problems, some assumptions were made, for example, the number of jobs and machines are expected to be limited (Blazewicz, 2007). Instead of repeating the terms, notations are usually used in scheduling problems. The numbers of jobs and machines are represented by *n* and *m* respectively. The letter *j* refers to a job while the letter *i* refers to a machine. The pair (*i, j*) indicates the process of job *j* on machine *i* when a job has to go through a sequence of manufacturing processes. Meanwhile, in the literature, $p_{i,j}$ represent the processing time of job *j* on machine *i*. The release date r_j of job *j* can be also defined as the ready date, or in other words, the time the job arrives at the system. The due date d_j of job *j* can be defined as the committed shipping or completion date. Completion of a job after its due date is acceptable, however a penalty will be charged. The weight w_j of job *j* refers to a prime concern factor, which indicates the importance of the job that is regarded as more important than other jobs in the system (Pinedo, 2012).

2.2.2 Scheduling Problems Notation

Most of the scheduling problems can be written in a notation form with three insertions. From the notation, it is able to tell how the product flows in the shop floor, under the types of constraints, and what should be achieved in that particular problem. Scheduling problem $\alpha|\beta|\gamma$ is a standard three-field notation presented by

(Graham et al., 1979). It represents each scheduling problem. The machine environment is written in the first field, α. The second field β presents the processing features and limitations, while the third field γ represents the objective function of a scheduling problem.

(a) Machine Environment, α

The notation **α** refers to column of the machine environment, where one will be able to know the flow pattern of the products in the shop floor by referring to this column. It can also be defined as a statement of the job routine pattern in the shop floor. Basically, machine environment can be divided into seven basic classes. They are single machine (1), parallel machines (P_m), unrelated machines (R_m), related machines (Q_m), flow shop (F_m), open shop (O_m) and job shop (J_m).

(b) Processing Characteristics and Constraints, β

The details of processing characteristics and constraints of a scheduling problem will be shown in the second column of the three-field notation. This column may contain no entry at all or multiple entries. Examples of processing characteristics and constraints are release date (r_j), preemptions ($prmp$), precedence constraints ($prec$), sequence-dependent setup times (s_{jk}), permutation ($prmu$), blocking ($block$) and recirculation ($rcrc$).

(c) Objective Functions, γ

The third column in the Graham notation reviews the objective function of a scheduling problem (Graham et al., 1979). It is an optimization process to find the parameter in its maximum or minimum values. The followings are some commonly seen objective functions to be solved in scheduling problems: makespan (C_{max}),

maximum lateness (L_{max}), total completion time ($\sum C_j$), total tardiness ($\sum T_j$), total number of tardy job ($\sum U_j$) and total weighted completion time ($\sum w_j C_j$).

2.3 Scheduling Approaches Classifications and Development

This section reviewed a range of approaches that have been proposed for solving scheduling problems. The approaches were built up based on different development theories. These approaches were basically divided into three main categories; optimization, heuristic and simulation approaches. Each category of approaches has to undergo a particular standard step while looking for the solution.

The classification of approaches was shown in Figure 2. 1. Optimization approaches can be further divided into enumeration method and mathematical equation method. Meanwhile, heuristic approaches can be categorized into constructive heuristics and improvement heuristics.

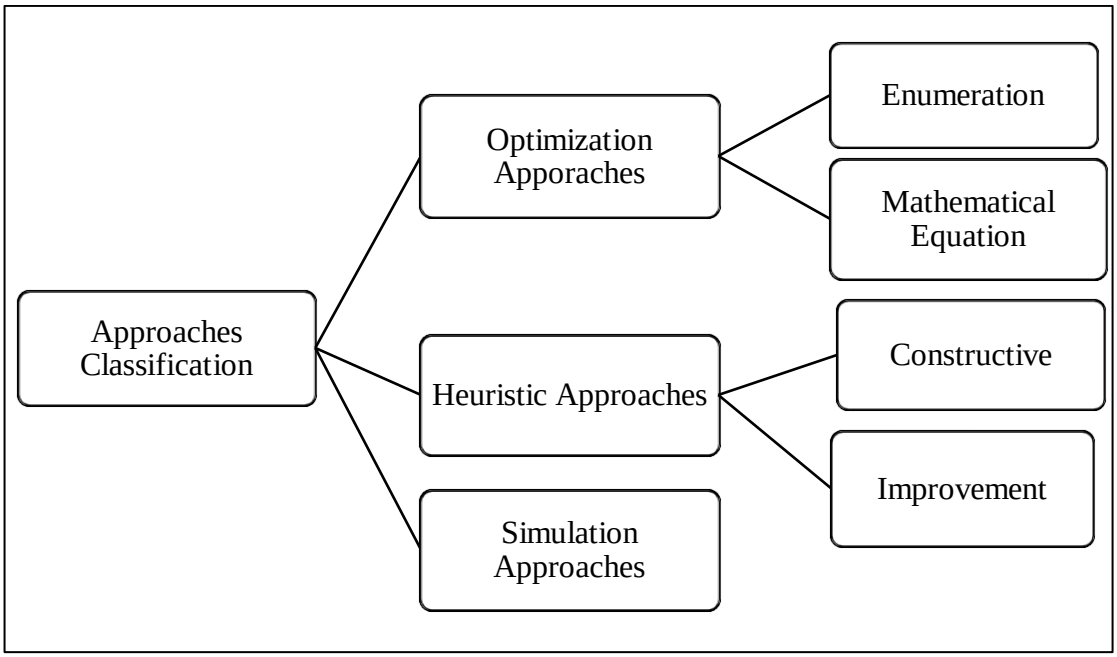

Figure 2. 1: Scheduling Approaches Classification

2.3.1 Optimization Approaches

Optimization methods are approaches proposed to obtain optimal solutions. The solution of the optimization problem is usually an optimum value of a certain objective function (Blazewicz, 2007). Optimizing problem can be either maximizing problem or minimizing problem. For example, minimization of idle time or maximization of throughput. This category can be further divided into two main groups, which are enumeration method and mathematical modeling. Enumeration method is a constructive approach which is usually developed by searching across a listing of all of the elements of a set. Meanwhile, mathematical modeling refers to a description of a system using mathematical equations and concepts to predict the future trend. This is extremely useful in scheduling problem that deals with time forecasting. One of the examples of optimization approaches is the Branch and Bound (B&B) method. However, due to the flow shop scheduling problem has been known to be NP-hard, the Branch and Bound method is not recommended to be adopted for large size problems (Lomnicki, 1965).

2.3.2 Heuristic Approaches

Heuristics refers to a solution that adopts the trial and error method or by rules that are only loosely defined. Heuristic method is built up by constructive heuristic and improvement method (Burke et al., 2007). The first category is named as constructive heuristic. This heuristic builds up a solution piece by piece. It determines the sequence of jobs by constructing a solution step by step. Although they may depend on the initial solution, search techniques may be deterministic, which means they always arrive at the same final solution through the same sequence

of solutions. Search techniques may be local. In other words, the solution is looking for the nearest optimum that may not be the real optimum (Ruiz and Maroto, 2005).

The second category is improvement heuristics that takes a solution and change the form of the structure to find a better solution. By adopting specific rules, improvement usually starts from an existing feasible solution that is usually found by one of the previous techniques and comes out with a better solution (Ancău, 2012). Search techniques may be stochastic where the solutions are considered and their order are different, depending on random variables. This search technique is able to find the true optimum even if it involves moving to the worst solutions during search, and they call this search technique as global.

2.3.3 Simulation Approaches

As stated by Nelson et al. (2001), simulation refers to the imitation of the operation of a real-world process or system over time. To run a simulation, first of all, a model has to be developed. The model here refers to the selected process. The model acts as the system, whereas the simulation acts as the operation of the system over time. Simulations are a very useful apparatus that allows experimentation without taking the risk. They are simplifications of the real world because they include only a few of the real-world factors, and are only as good as the situation meets their assumptions.

The approaches discussed above are proposed for solving all types of scheduling problems for different machine environments. Since flow shop problems have been proven to be NP-hard (Garey et al., 1976), heuristic method is the most suitable among other solutions, especially for large-size problems (Laha and Sapkal, 2014). This limitation has given the direction to researchers to develop the heuristic method.

2.4 Review of Flow Shop Scheduling Problems (FSSP)

Among the machine environment, flow shop scheduling problems (FSSP) are widely reviewed machine environment in scheduling problems (Haibo and Chunming, 2010). This sub-class environment has taken up about 25% of the manufacturing system, assembly lines, and information service facilities. In practical situations, flow shop scheduling problems are proven to be NP-hard (Garey et al., 1976). This section gives a general review on the criteria of different types of flow shop scheduling problem and its objective function to achieve.

The criteria of FSSP are listed in the paper by Solimanpur et al., 2004. At time zero, all jobs are ready for processing in flow shop scheduling problems. On each machine, each job can only undergo the process once, and the processing time is assumed to be zero if a job does not undergo a particular machine. Once an operation is started on a machine, it has to be not interrupted before the process is done (non-preemption). All machines are available at time zero. Only one job at a time is allowed in every machine. Each job can be processed on only one machine at a time. The setup times are sequence independent.

Meanwhile, another definition of flow shop is referred as a scheduling environment that contains *m* machines in series. Each job has to be processed on each one of the *m* machines. All jobs have a fixed route, i.e., they have to undergo the operation on machine 1 first, then on machine 2, etc.. After it is done on one machine, a job continues the process at the next machine (Pinedo, 2012).

2.4.1 Types of Flow Shop Scheduling Problems

In flow shop, jobs are assigned to undergo the processes in a series arrangement. Flow shop scheduling problems can be further categorized into general flow shop and flexible flow shop. Each category has its own constraints in its process characteristic. For general flow shop, it can be divided into permutation flow shop and non-permutation flow shop. Both permutation flow shop and non-permutation flow shop can be either finite buffer or infinite buffer. On the other hand, the flexible flow shop will only have infinite buffer. Figure 2. 2 shows the classification of flow shop scheduling.

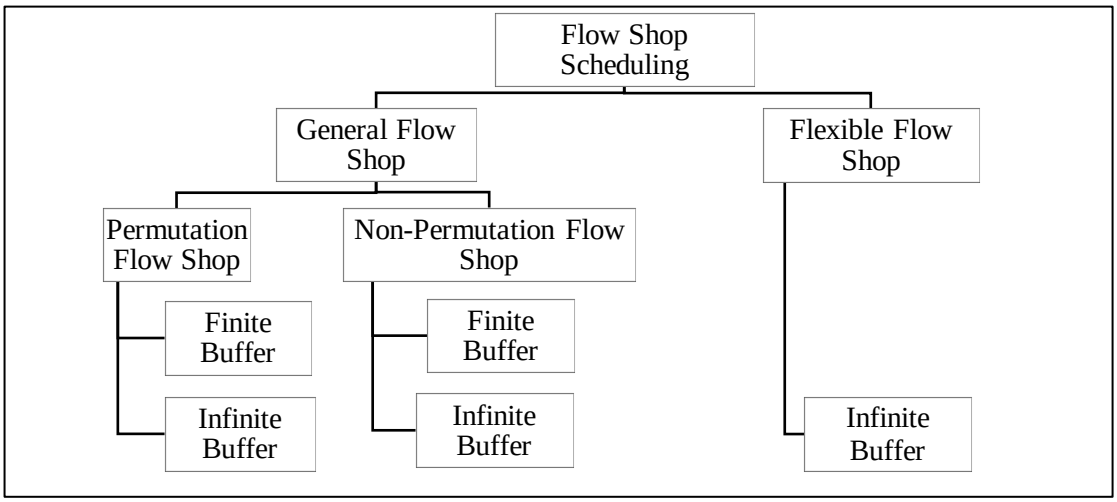

Figure 2. 2: Classification of Flow Shop Scheduling

(a) General Flow Shop

General flow shop can be divided into permutation flow shop and non-permutation flow shop. For non-permutation flow shop, it is usually represented by three-field notation written as $F_m \mid\mid C_{max}$. In this condition, the jobs may not flow

according to the First Come First Served principle. The order of jobs move to the machines may vary from one machine to another. This class of scheduling problem will give a total of $(n!)^m$ possible solutions (Vahedi-Nouri et al., 2014). Meanwhile, for permutation flow shop, it can be represented by $F_m \mid prmu \mid C_{max}$, in which this type of scheduling problem will generate a total of $n!$ solutions. In permutation flow shops, sequence between the machines is not allowed to change. In other words, the same sequence of jobs is maintained throughout the whole process (Juan et al., 2014a). Finding an optimal schedule for non-permutation flow shop is significantly harder than finding an optimal schedule for permutation flow shop. For both permutation flow shop and non-permutation flow shop, it can be further divided into infinite and finite buffer. Usually, the literature on flow shop scheduling is restricted to a specific case of flow shop, for example the permutation flow shop, in which each machine processes the jobs in the similar sequences. Therefore, in a permutation flow shop, the job sequence will be maintained on all remaining machines when the job sequence on the first machine is fixed (Blazewicz, 2007).

(i) Infinite Buffer

In infinite buffer, the buffer capacities between successive machines may be unlimited. This condition is usually seen in the industry producing small size products such as the semiconductor industry, thus that large amount of products can be stored in between the machines. However, for a large-size product, the buffer capacity between two successive machines may be limited, and this is the reason for the occurrence of blocking. Blocking refers to the condition when the buffer is occupied and the machine at the upper stream is unable to place a job into the buffer

after the processing is done. In such situation, the job cannot be passed down to the lower stream machines.

(ii) Zero or Finite Buffer

Zero or finite buffer is the condition where *m* machines are in series with zero intermediate storage between successive machines. When a machine has completed processing a job, that job does not allow to be passed to the next machine if that machine is busy; the job must remain on the previous machine, therefore no subsequent jobs are allowed to take place and cause blocking. The problem of minimizing the makespan in a flow shop with zero buffer can be written as *Fm | block | Cmax*.

(b) Flexible Flow Shop

A machine environment that is divided into a number of stages that is in series with a number of machines in parallel at each stage. The job has to be on one of the machines at each stage (Nahavandi and Gangraj, 2014). Flexible flow shop sometimes is also referred to as a compound flow shop, multi-processor flow shop, or hybrid flow shop. There is an infinite buffer capacity between any two successive stages. Figure 2. 3 shows the flow in a flexible flow shop. Minimizing the makespan of flexible flow shop can be referred as $FF_c \mid\mid C_{max}$.

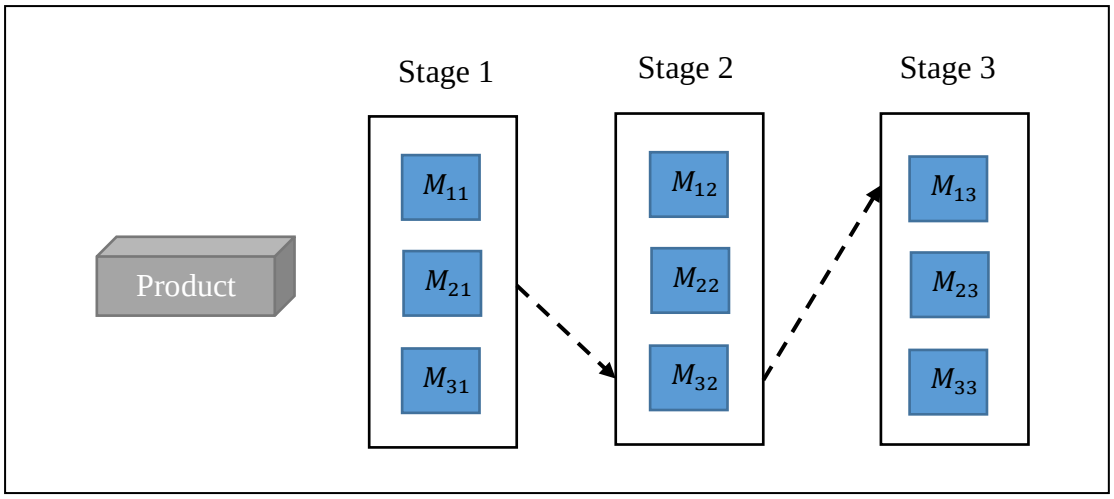

Figure 2. 3: Flexible Flow Shop

2.4.2 Objective Functions of Flow Shop Scheduling

Objective functions can be explained as the main objective to be achieved in a scheduling problem. Usually, in different machine environments, there will be a different concern of problems. In this section, the main goals seen in the literature under flow shop scheduling problems are reviewed. The main objective functions include makespan (C_{max}) minimization, flow time ($\sum C_j$) minimization, and tardiness ($\sum T_j$) minimization. Under the flow shop category, these are the most commonly seen objective functions solved by the papers proposed.

(a) Makespan

Makespan refers to the completion time of all jobs in a batch. Flow shop scheduling problems with the objective of makespan minimization ($\alpha \mid \beta \mid C_{max}$) have been an eminent research topic. This topic gains a lot of attention due to the reduction of makespan, in which the utilization and throughput will also be improved (Pinedo, 2012, Mirabi, 2014). To solve the flow shop scheduling problem with the

objective of makespan minimization, the proposed solution can be divided into exact approaches and heuristic approaches. For exact approach, a combinatorial algorithm was proposed (Smith and Dudek, 1967). Meanwhile, a branch and bound technique was proposed to solve flow shop problems (Ignall and Schrage, 1965). Among all the papers that have proposed for solving makespan minimization in flow shop problem, a test was carried out to determine the superior solutions (Framinan et al., 2004). Among the 177 approaches, NEH heuristic proposed by (Nawaz et al., 1983) performed best for makespan minimization. This approach contains two main steps; first, the job is arranged in a decreasing total processing time order. Then, the job is slotted in according to the sequence to look for the minimum makespan.

(b) Flow time

Flow time refers to the total time a job spent in the shop. Besides processing time, the time spent in inventory and pre-production time are also taken into consideration. Based on several studies, the reduction of flow time is able to reduce the average number of work in progress (WIP) (Pasupathy et al., 2006, Yenisey and Yagmahan, 2014). Since the time spent in the system is shortened, then the inventory in the process is also reduced. Due to this reason, minimizing total flow time in flow shop scheduling problems ($\alpha \mid \beta \mid \sum C_j$) is gaining more attention. Based on the literature, not much work has been carried out on the exact techniques proposed for minimization of flow time under flow shop scheduling problems. In the paper of (Framinan et al., 2003), a study was carried out to review a range of heuristics for flow time minimization based on permutation flow shop problems. For the heuristics that were proposed before the year 2000, the work of WY heuristic (Woo and Yim, 1998) is an example of good current work. In 2003, an FL heuristic was developed by Framinan and Leisten by using NEH heuristics as the framework. The results

were then compared to the WY heuristic. Computational experiments carried out showed that FL heuristic outperformed WY heuristic.

(c) Tardiness

Total tardiness means the sum of the difference between the completion time and due date of the jobs in the system. A flow shop problem with the objective of total tardiness minimization can be written as $(\alpha \mid \beta \mid \sum T_j)$. This function mainly focuses on satisfying customer's demand (Amin-Tahmasbi and Tavakkoli-Moghaddam, 2011). In the work of (Pan et al., 2002), a Branch and Bound algorithm was presented to solve the two-machine flow-shop scheduling problems with the objective of minimizing total tardiness. Based on the computational experiment, the proposed algorithm solved almost all of the test problems. A flow shop scheduling problem with blocking in-process was discussed in the paper of (Ronconi and Henriques, 2009). In their work, no buffers were available between the successive machines. Heuristic approaches were proposed to minimize the total tardiness objective. The main idea proposed a constructive heuristic that explores specific characteristics of the problem.

Among the three objective functions discussed above, only few literature made a statement that minimizing makespan is the top concern for the heuristics proposed in solving flow shop scheduling problems (Vasiljevic and Danilovic, 2015, Framinan et al., 2004, Framinan et al., 2003, Ruiz and Maroto, 2005).

2.5 Overview of Permutation Flow Shop Problems (PFSP)

This section gives particular attention on PFSP as this is one of the important subclasses of scheduling problem that gains a lot of attention among the researchers (Zhao et al., 2014). PFSP fall under the general flow shop problem as discussed

earlier. Most researches of flow shop scheduling problems have focused on permutation schedules due to the relative combination simplicity of scheduling that can be identified clearly, simply by giving a permutation of the jobs (Potts et al., 1991). The permutation flow shop problems focus on deciding the sequences of *n* jobs to be place in the process on *m* sequential machines. The jobs undergo the operation according to the sequences of machine 1, machine 2, . . . machine *m*. Each machine can only process one job at a time and each job can be processed by only one machine at a time without preemption. No job is allowed to leap over any other job, in other words, jobs are processed in machine 1 will be leading throughout the system. In addition, when a job is ready for processing, no machine will remain idle. All jobs and machines are available at time zero (Singhal and Hemrajani, 2013). In PFSP, the order in each machine processes the jobs remains unchanged for all machines. There is numerous literature on heuristics and metaheuristics methods for the PFSP problem and makespan criterion. Makespan is emphasized here as it is the most common optimization criterion in the proposed papers. Since the publication of the paper by Johnson (Johnson, 1954), the PFSP have become one of the most broadly studied topics in scheduling. For m$\geqslant$3, the problem is proven to be strongly NP-complete (Garey and Johnson, 1979).

The characteristics of PFSP have the criteria as follows (Wu and Gu, 2004). At the beginning of the planning period, all jobs are available. The set-up time is included in the processing times. On each machine, only one job can be processed at a time. No preemption is allowed. If the previous job is still being processed, a job is not allowed to begin its process. There is infinite buffer storage capacity between all machines. If job *j* is completed on machine *i*, then it moves out from machine *i* and

goes to machine *i+1* if it is ready; otherwise, the job waits in the intermediate storage. Each job will only be processed on every machine once.

2.5.1 Notable Proposed Approaches

In this section, several well-known proposed algorithms are discussed. All these algorithms are proposed for solving flow shop problems that fall under heuristics category. The algorithms can be divided into three main development phases. These three phases are:

- Phase 1: Index development
- Phase 2: Solution construction
- Phase 3: Solution improvement (Framinan et al., 2004)

These three phases are discussed one by one in this section.

- **Phase 1: Index Development**

An algorithm that is built up by index development will arrange the jobs according to a certain property, and one of the common indexes is the processing time of jobs on each machine. When adopting this method, no assumption is needed while the arrangement is made. The output of this phase is an increasing or a decreasing arrangement of the jobs that can be used as an input in the next phase.

The first heuristic approach regarding the idea of ranking the jobs according to the processing time was proposed (Palmer, 1965). The author proposed a "slope index" to arrange the jobs on the machines according to the processing time. The idea stated that jobs located at the beginning of the sequence should have increased processing times from machine to machine, while jobs located at the end of the sequence should have decreased processing time requirements.

The CDS (Campbell, Dudek and Smith) heuristic has been proposed by (Campbell et al., 1970). It has been developed based on the idea of Johnson's algorithm. It is categorized under index development because it assigns jobs based on the processing time of the two virtual machines. By grouping the m original machines into two virtual machines, CDS comes out with $m - 1$ schedules and continues solving the two machines problem by adopting Johnson's rule. Gupta modified the idea of Palmer's slope index and proposed a new solution which takes the benefit from the similarities between scheduling and sorting problems (Gupta, 1971).

Dannenbring proposed a rapid access (RA) heuristic (Dannenbring, 1977) that combined the previous idea of Johnson's algorithm (Johnson, 1954) and Palmer's slope index. The concept of virtual two machines problem as defined in the CDS heuristic (Campbell et al., 1970) was used; however two weighting schemes were calculated instead of directly applying Johnson's algorithm over the processing times. RA developed a good solution in a short period as the name implied.

- **Phase 2: Solution Construction**

An algorithm is developed in a repetition method with the purpose to insert one or more unscheduled jobs in one or more positions of a partial schedule until all the jobs have been inserted in the schedule. In this phase, jobs are divided into two main groups, which are scheduled jobs and unscheduled jobs. A job is selected from an unscheduled group and placed into the scheduled group. The property that is concerned during the job selection in the partial sequence might be the objective function of the scheduling problem. For example, the makespan.

In 1983, an NEH (Nawaz, Enscore and Ham) heuristic was proposed for solving permutation flow shop problem (Nawaz et al., 1983). The authors proposed the idea