A MODIFIED MULTI-STEP CROSSOVER FUSION (MSXF) IN SOLVING SOME DETERMINISTIC JOB SHOP SCHEDULING PROBLEM (JSSP)

MAHANIM BINTI OMAR

UNIVERSITI SAINS MALAYSIA
2008

ACKNOWLEDGEMENTS

In the name of Allah, I would like to praise Allah S.W.T. for the strength and guidance He gives to me. The warmest and sincere thanks go to my supervisor Assoc. Prof. Adam Baharum, whose encouragement and willing support guided me through this end. Thanks also to my co-Supervisor, Dr. Yahaya Abu Hassan. I am also grateful to Professor M. Ataharul Islam for his guidance and valuable suggestion and comment throughout the work. My warmest thanks also go to School of Mathematical Sciences' Dean, Assoc. Prof. Dr. Ahmad Izani, Deputy Dean of Graduate Students, Assoc. Prof. Dr. Norhashidah Ali and all academic and non-academic staff at School of Mathematical Sciences for their generosity in providing the good study environment and facilities.

My appreciation also goes to all my friends and those who have in one way or another helped, encouraged and motivating me during the progression of this thesis. I would like to dedicate this work to my parents, Omar Othman and Mazenah Che Rus; and my family for their endless support and encouragement.

Thank you.

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF APPENDICES

A MODIFIED MULTI-STEP CROSSOVER FUSION (MSXF) IN SOLVING SOME DETERMINISTIC JOB SHOP SCHEDULING PROBLEM (JSSP)

Abstract

This thesis addresses the job shop scheduling problem (JSSP) with the objective of minimising the makespan value. In this study, stochastic sampling method mainly used in simulated annealing (SA) is implemented in population based approach framework called genetic algorithm (GA) to solve JSSP. A special crossover called multi-step crossover fusion (MSXF) with intuition of generating a child from the path re-linking technique is employed. MSXF is an extended version of local search. It utilizes a neighbourhood structure and a distance measure in its procedure. By using this type of crossover, a solution or child is generated between both parents using the search path joining parent solutions. In this study, two modifications have been made with intention to increase the effectiveness of the algorithm in producing good solutions.

The first modification is the changing in the structure of neighbourhood from active critical block neighbourhood (ACBN) proposed by Yamada and Nakano (1997) to the structure of neighbourhood proposed by Nowicki and Smutnicki which has been used by Gaspero (2003). Since there is a suggestion that the greedy initial solution often results in better quality solution, therefore, the implementation of pairwise interchange in GA framework is hoped to help with the early convergence. It is evident from the results that the implementing MSXF with pairwise interchange does increase the quality of solution. The experimental study has shown that, under our selected parameters based on the experimental study, the structure of neighbourhood proposed by Nowicki and Smutnicki with additional pairwise interchange performs better than ACBN.

SUATU PENGUBAHSUAIAN KOMBINASI PERSILANGAN MULTI-LANGKAH (MSXF) DALAM MENYELESAIKAN BEBERAPA MASALAH PENSKEDULAN KERJA KEDAIAN (JSSP) BERKETENTUAN

Abstrak

Tesis ini membincangkan masalah penskedulan kerja kedaian (JSSP) dengan objektif untuk meminimumkan masa operasi. Dalam kajian ini, prosedur pensampelan stokastik yang biasanya digunakan dalam simulasi penyejukkan (SA) diimplimentasikan dalam pendekatan rangka kerja berasaskan populasi dalam algoritma genetik (GA) bagi menyelesaikan JSSP. Suatu operator persilangan yang dikenali sebagai kombinasi persilangan multi-langkah (MSXF) bertujuan untuk menghasilkan anak atau individu baru daripada teknik paut-semula laluan digunakan. MSXF adalah teknik yang dikembangkan dari teknik pencarian setempat. Dengan menggunakan MSXF, suatu penyelesaian ataupun anak dihasilkan dengan menghubung ibu bapa menggunakan penyelesaian pencarian laluan hubungan ibu dan bapa. Dalam kajian ini, dua modifikasi telah dibuat dengan tujuan untuk meningkatkan kualiti penyelesaian.

Pengubahsuaian yang pertama adalah melibatkan struktur kejiranan yang digunakan di dalam kombinasi persilangan multi-langkah. Struktur yang dicadangkan oleh Nowicki dan Smutnicki serta di implementasi oleh Gaspero (2003) telah digunakan untuk menggantikan struktur yang dicadangkan oleh Yamada dan Nakano (1997). Oleh sebab terdapat cadangan yang menyatakan bahawa penyelesaian awal yang tamak biasanya akan menghasilkan penyelesaian yang lebih berkualiti, maka pertukaran pasangan demi pasangan diharapkan dapat membantu dalam penumpuan awal kepada penyelesaian terbaik. Keputusan membuktikan bahawa implementasi kombinasi persilangan multi-langkah dengan pertukaran pasangan dapat meningkatkan kualiti penyelesaian. Keputusan kajian juga telah menunjukkan bahawa dengan menggunakan parameter yang telah dipilih berdasarkan kajian experimentasi,

struktur yang dicadangkan oleh Nowicki dan Smutnicki dengan implementasi pertukaran pasangan mencapai keputusan yang lebih baik berbanding ACBN.

CHAPTER 1

INTRODUCTION

1.1 Background of the Study

In the modern competitive environment in manufacturing and service industries, the effective sequencing and scheduling has become an essential for survival in the marketplace. Companies have to produce their product untimely as opposed to due date. Otherwise, it will impinge upon reputation of a business. At the same time, the activities and operations need to be scheduled with the intention that the available resources will be used in an efficient manner. As a result, there is a great good scheduling algorithm and heuristics are invented.

Basically, scheduling is the allocation of shared resources over time to competing activities (Baker, 1974). The terminology of scheduling theory usually takes place in the processing, manufacturing industries, production, transportation, distribution, and in information processing and communication. Scheduling problem is solved by using the mathematical techniques or heuristic methods to allocate limited resources to the processing of tasks. This allocation of resources is important since a proper allocation enables the companies to optimise their objectives and achieve their goals. The familiar scheduling problems are the bus schedules, the university time-table and the construction work.

The classified field of scheduling problem is fascinating area of study and research with continue interest even in recent times. Recently, researches have been focusing on investigating machines scheduling problems in manufacturing and service environment where jobs represent activities and machines represent resources. In this environment, each process can process one job at a time. If it is a low volume system,

it is known as job shop scheduling problem. In this type of environment, products are made to order. Usually, these orders differ from terms of processing requirements, material needs, processing time, processing sequence and setup times.

Most of the prevailing practical scheduling problems exist in stochastic and dynamic environment. Stochastic is a problem where some of the variables are uncertain while dynamic problem is when jobs arrive randomly. On the other hand, the problems with ready time is known and fixed are called problems static and for problem where all the parameter such as processing times are known and fixed is called deterministic problems (French, 1982). In spite of this, it is quite impossible to predict exactly when jobs will become available for processing. Additionally, the understanding of scheduling problems where there is no uncertainty involved will help us towards the solution of stochastic and dynamic problems. Therefore, in this work, we will study and limit our discussion in static and deterministic environment.

The main objective in solving the job shop scheduling problem is to find the sequence for each operation on each machine that optimises the objective function. The most common objective function that has been used in scheduling the job shop problem is minimisation of makespan value or the time to complete all jobs. It has been the principal criterion for academic research and is able to capture the fundamental computational difficulty which exists unconditionally in determining an optimal schedule (Jain and Meeran, 1999).

From the theory, we know that the cost of processing may depend crucially upon the choice of schedule. If we were not able to find the best schedule within reasonable time, we should use the knowledge to find the schedule which may at least be expected to perform better. During the 1960s, the way of solving scheduling problems has been shifted from the exact solution to an enumerative algorithm.

However, this technique is failed to find the feasible solution for many problems and very limited of used (Jain and Meeran, 1999). Only by the end of 1980s the researcher started to solve the problem by using approximation and heuristic methods. Since then, more innovative algorithms were formulated such as shifting bottleneck (Adam et. al, 1988), tabu search (Glover, 1994), simulated annealing (Lorenco, 1995) and genetic algorithms (Nakano and Yamada, 1991), (Yamada and Nakano, 1997) and (Mathur, 1999).

Adam *et al*. (Adam *et.* al, 1988) proposed shifting bottleneck to find the reasonably efficient schedule for job shop problems. This method iteratively identifies a bottleneck machines and optimise its job sequence. In 1990s, Fred Glover (Glover, 1994) proposed a deterministic local search but implementing the recording of previous solution to prevent cycling and to promote diversified coverage of the search space. On the other hand, simulated annealing solves the job shop problems based on the analogy with the physical process of annealing.

Different from other approximation procedures, genetic algorithms (GAs) can be uniquely characterized by their population based search strategy and their operators; selection, crossover and mutation (Yamada, 2003). Compared to other procedures, genetic algorithms search from a population point, not a single point. If we are searching from a single point, our system will possibly easy to get trapped at the local optima (Michalewicz, 1999). In addition, while working with genetic algorithm, we do not need too much information and it also can be easily adapted to our problem. Due to these factors, this study is dedicated to job shop scheduling problem using genetic algorithm framework.

Genetic algorithm maintains a population of individuals at each iteration. Each of this individual represents a potential solution to the problem. Each solution evaluated

to give some measure of its “fitness”. Then, by selecting the fittest individuals will form a new population for the next iteration. In the reproduction process, some selected members endure amendment by means of crossover and mutation, to form new solution. Crossover operator combines some parts from two individuals to form a new individual. On the other hand, mutation operator creates a new individual just by making some changes in a single individual.

In order to solve combinatorial problems such as job shop scheduling problems, we usually find it is difficult to construct an efficient crossover operator that recombines solution. The exchanging of genes between parents may violate the constraint of the problem, generating many infeasible solutions. In addition, genetic algorithm is not well suited for fine tuning structures which are very close to optimal solution. The general remedy is by incorporating local search methods such as neighbourhood search in the genetic algorithm framework (Yamada, 2003).

Yamada and Reeves (Yamada and Reeves, 1998) have been studying in merging the stochastic sampling method essentially used in simulated annealing and the best descend method elaborate in tabu search and implement them in genetic algorithm framework. This proposed method is called multi-step crossover fusion (MSXF). The multi-step crossover fusion (MSXF) usually used to solve a combinatorial problem. It utilizes a neighbourhood structure and distance in the search space. Traditionally, crossover combines some parts from two individuals to form new individual. Unlike other traditional crossover operators, MSXF is more based on search oriented. It generates descendents along the path connecting two parents. MSXF is especially useful while being implemented with local search since it exploits a good starting point for the subsequent local search. Apparently, the structure of neighbourhood is very important while working with MSXF since local search always

known consuming large amount of time. Stand for that reason; reducing the size of neighbourhood possibly could decrease the amount of computational time.

1.2 Problem Statement

The previous study of implementing multi-step crossover fusion in genetic algorithm framework did not focus on the size of problems and how it is related to the selected parameters. As we already known, application of GA may involve many parameters including the chosen of population size, the selection operators, the probability of implementing crossover and the important of mutation operator. Basically, the quality of solution using genetic algorithm relies on the choice of the best parameters in order to prevent premature convergence and to ensure the diversity in the search space (Essafi *et.* al, 2007). Based on this fact, in this study, we try to study the behaviour some of these parameters in varies size of problems.

Before we apply the local search to our problem, we have to define a neighbourhood structure on the set of feasible solution. Basically, the set of neighbours of a solution is defined as a set of solutions which differ only by small changes (Hurink, 1998). Defining the neighbourhood structure is very important since it will determine the way in which we navigate through the solution space and it also determines the computational time of finding the best solution. According to Hurink (Hurink, 1998), the computational time for finding the best or an improving neighbour is proportional to the size of the neighbourhood. In spite of this, large neighbourhood gives more possibilities to change the current solution and raises the possibility of reaching high quality solution. In practice, only computational tests have to show the alternative leads to better result. Therefore, we compared two different kind structure of neighbourhood namely the active critical block neighbourhood (ACBN) and the structure proposed by Nowicki and Smutnick to be implement in the MSXF.

The multi-step crossover fusion (MSXF) in genetic algorithm framework is implemented to the different size of benchmark problems. The sizes of problems represent the small, medium and large size of problem. These problems are chosen because the best solution found for each problem is already known. This will help us to measure up the ability of this algorithm.

1.3 Objectives of the Study

The objectives of the study are;

- To study the performance of implementing modified MSXF in GA framework to job shop scheduling problem for small, intermediate and large size of problems.
- To study the influence of parameters and find the suitable set of parameters which is applicable in the MSXF-GA. The parameters discussed are;
    - Size of population.
    - Selection criteria for MSXF-GA. Three selection procedures will be compared. They are roulette wheel selection, tournament selection and uniform selection.
    - Mutation operator. We want to study the important of mutation operator in the algorithm.
    - *T* value in the Metropolis Acceptance Criterion.
- To compare;
    - The performance of MSXF-GA with conventional GA, basic dispatching rules and shifting bottleneck without backtracking procedure.
    - The performance of Multi-Step Crossover Fusion (MSXF) with Generalised Order Crossover (GOX) and Partially Preservative Crossover (PPX) in GA framework.

- The performance of different neighbourhood structures while implementing MSXF in GA framework to job shop scheduling problems.
- The effectiveness of implementing an additional pairwise interchange in the GA framework.

1.4 Significant of the Study

This thesis is devoted mainly to study the performance of multi-step crossover fusion in genetic algorithm framework in deterministic and static job shop problems. Different sizes of problem were used as the testing problems so that the conclusion of its performance can be derived for variety of problem sizes.

This study is hoped to provide the insight of implementing the multi step crossover fusion in genetic algorithm framework. Whilst, the simple study on structure of neighbourhood is hoped to act as starting point for more practically relevant and effective structure of neighbourhood.

1.5 Outline of the Thesis

This thesis focuses on the solving the deterministic and static job shop scheduling problems using the multi-step crossover fusion (MSXF) in genetic algorithm (GA) framework.

In Chapter 2, the basic concepts such as notations, formulation and representation of job shop scheduling problem are described. Followed this is the literature of the previous work devoted to the same problems. This chapter helps us to determine the focus problem and the algorithms and procedures ever built for the problem.

In Chapter 3, the methodology of implementing multi-step crossover fusion (MSXF) in genetic algorithm framework is explained. This chapter describes the procedures used and other partially procedures which need to perform along with the MSXF in the framework.

Chapter 4 consists experiments on some selected parameters. The relation of the selected parameter and the problems is shown regarded in finding the best to perform in the framework. Then, these selected parameters are applied to large size problem to measure the performance of the algorithm. In addition, comparison with other crossovers techniques were also been experimented.

In Chapter 5, we compare the multi-step crossover fusion (MSXF) with conventional genetic algorithm, shifting bottleneck procedure, some basic dispatching rules and simulated annealing.

Chapter 6 is a simple study on the comparison of the neighbourhood structure proposed by Nowicki and Smutnicki compared to active critical block neighbourhood (ACBN). Since the size of the neighbourhood sometimes affecting the computational time, therefore, this chapter is hoped to get an overview of the problem arises regarding the structure of neighbourhood. As an addition, we introduce the additional pairwise interchange to increase the quality of solution.

Finally, in Chapter 7, the study of this thesis is summarised and the future exploration are suggested.

CHAPTER 2

LITERATURE REVIEW

2.0 Introduction

In order to understand the problems as well as the methods available related to job shop scheduling problems, we have reviewed some of literature related to the problem in the past and recent publication. However, most of the literature in this area refers to the Job Shop Scheduling Problem (JSSP) and uses the terminology of manufacturing such as, job, machine, operation, routing and processing time. The developments of the idea in solving this problem by other researchers are briefly presented in this chapter. We reviewed the literature in three different sections. The first section we discussed some background of job shop scheduling problem. In the second section, the methods available for the problems are discussed. Finally, in the final section, we focussed on the work dedicated to genetic algorithms.

2.1 Background of Job Shop Scheduling Problem (JSSP)

In practice, scheduling problems are complex problems. According to Conway *et al.* (1967), scheduling a job shop process is the task of assigning each operation to a specific position on the time scale of the specified machine. Alternatively, scheduling can be regarded as the task of constructing an ordering of the operations associated with each machine. A job shop problem consists of n number of jobs J, $\{J_i\}_{i=1}^{n}$ which are needed to be processed on m number of machine M, $\{M_k\}_{k=1}^{m}$. Each job J_i consists in number of operations, O, that must be processed on each machine M_k in specific route or order. O_{iqk} is the q-th operation for job i, J_i, which should be processed on machine k, M_k, for an uninterrupted processing time period p_{iqk} and pre-emptive of operation are not allowed. Once the machine started to process an

operation, it must be complete before another operation started on that same machine. Each machine can process only one job at a time and each job can be processed by only one machine at a time. Each job must be processed until it complete. The set-up time for each machine and the time to shift the job between machines are neglected. In addition, the machines are assumed to be available throughout the scheduling period and no breakdown occurred.

Traditionally, scheduling problems have been viewed as problems in optimisation subject to constraints. The theory of scheduling also includes a variety of techniques that are useful in solving scheduling problems. The selection of an appropriate technique depends on the complexity of the problem, the nature of the model, and the choice of a criterion as well as other factors. According to Baker (1974), while classifying the major scheduling models, it is necessary to characterise the configuration of resources and the behaviour of the tasks. For example, if the set of tasks available for scheduling does not change over time, the system is called static. On the other hand, the system which new tasks arise over time is called dynamic. In addition, he also stated that traditionally, static models have been proven more tractable than dynamic models and have been subjected to more extensive study. Other classifications that have been made are deterministic problem and stochastic problem. Deterministic problem is a problem with all the parameters such as processing time, are known and fixed and the problems which the processing time is uncertain are called stochastic. All practical scheduling problems are dynamic and stochastic since it is impossible to predict exactly when the jobs are arrived or when the breakdown of machine are occurred. Even though most of the problems fall in dynamic and stochastic types, but still most of the research are done for static and deterministic. According to French (1982), there are problems in which any randomness is quite clearly insignificant. For example, the uncertainty in the various quantities is several orders of magnitude less than the quantities themselves. He also added that study of

dynamic and stochastic problems cannot be done until we have understood the static and deterministic problems.

2.2 Problem Notations and Assumptions

Before we proceed to review previous works on this study, we shall indicate in general a few of criteria which we will use later. Table 2.1 shows the notations subject to the job shop scheduling problem;

Table 2.1: Notations

J_i	Job i. ($i = 1,2,....n$)
M_k	Machine k ($k = 1,2,...,m$)
O_{iqk}	q-th operation for job i, J_i, which should be processed on machine k, M_k.
s_{iqk}	Starting time for O_{iqk}.
r_{iqk}	Ready time for O_{iqk}.
p_{iqk}	Processing times for O_{iqk}.
C_{iqk}	The completion time (makespan) of O_{iqk}. $C_{iqk} = s_{iqk} + p_{iqk}$
F_i	Flow time of job. The time that J_i spend in the workshop. $F_i = C_i - r_i$. C_i is completion time of J_i.
$n \times m$	n number of jobs need to be processed on m number of machines where $n = 1,2,...$ and $m = 1,2,...$

For further analysis, we have to make some assumptions about the structure of our scheduling problem. Therefore, the assumptions we made are;

1. Each job is an entity. Although the job is combination of set of operations (tasks), however processing two operations of the same job simultaneously are not allowed.
2. Pre-emptive are not allowed. When the machine started to process an operation, it must be complete before another operation started on that same machine.
3. No cancellation. Each job must be processed until it complete.

4. Set-up times are not considered. The time to shift jobs between machines and the time setting for each machine after the job last processed are not considered.
5. There is only one machine for each type of process.
6. Machines may be idle.
7. Machine may process only one operation at a time.
8. Machines never breakdown and are available throughout the scheduling period.
9. The technological constraints are known in advance and are immutable.
10. There is no randomness. In particular;

    a. The number of jobs are known and fixed.
    b. The number of machines are known and fixed.
    c. The processing times (duration times) are known and fixed.

Assumption 8th and assumption 10th confine attention to non-random problems that is problem with all the numerical quantities are known and fixed in advance. There is no uncertainty.

French (1982) classified scheduling problem is the four field notation $A/B/C/D$. A can be defined as the number of jobs, B is the number of machines, C is the flow pattern within the shop and D is the performance measure for the problem. However, this descriptive technique is suitable only for the basic problem. Stating the performance measure for the problem is always conflicting. According to French (1982), there are at least 27 scheduling goals. However, Jain and Meeran (1999) suggested that minimisation of the makespan value is widely used in academic and industrial practice. Makespan is the time when the last operation leaves the

workspace and can be noted as C_{max}. Consequently, according to them, it has been the principal criterion for academic research and able to capture the fundamental computational difficulty. It is also suggested that a solution for C_{max} is likely to perform well on average with respect to the other criteria. Makespan can be formulated as equation 2.1.

$$min(C_{max}) = \min_{\text{feasible schedules}} \left(max\left(s_{iqk} + p_{iqk}\right): \forall\, i \in J, k \in M\right) \tag{2.1}$$

where s_{iqk} is starting time for operation O_{iqk} and p_{iqk} is processing time for operation O_{iqk}. Studies for static and deterministic job-shop scheduling problems with minimising the makespan value have been done by Adams *et al.* (1988), Nakano and Yamada (1991), Bierwirth (1995), Schmidt (2001) and Yamada (2003).

2.3 Problem Formulation and Representation of JSSP

Job shop problem can be represented in matrix form such as Nakano (1991). Consider a job shop problem with n jobs that have to be processed on m machines. Operation (j,k) refer to job j that must be performed on machine k. Each job has their processing times and denoted by p_{iqk}. Each job has to be processed by a given machines order which are called technological sequence. This technological sequence of machines can be different to each job and the order of jobs to be processed on a machine can also be different to each machine. The predefined technological sequence of each job can be given collectively by matrix $\{T_{jk}\}$ which $T_{jk} = r$ corresponds to the k-th operation, O_{jkr}, of job j on machine r. On the other hand, $\{p_{jr}\}$ corresponds to the processing time for operation O_{jkr}. An example of 3×3 job shop problem including the machine sequence for each job and processing time for

each operation are given in Table 2.1. Figure 2.1 represents the machines' sequence for each job in matrix form. The matrix's rows represent the jobs while the columns represent the machine's sequence for each job. Figure 2.2 represent processing time for each operation in matrix forms. The matrix's rows represent the jobs while the columns represent the machines.

Table 2.2: Machines Sequence and Processing Time for 3×3 Job Shop Problem

Job	Machines(processing time)		
1	1(60)	2(30)	3(02)
2	2(75)	3(03)	1(25)
3	3(05)	2(15)	1(10)

$$\{T_{jk}\} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 2 & 1 \end{bmatrix}$$

Figure 2.1: Machines Sequence

$$\{p_{jr}\} = \begin{bmatrix} 60 & 30 & 2 \\ 25 & 75 & 3 \\ 10 & 15 & 5 \end{bmatrix}$$

Figure 2.2: Processing Sequence

2.3.1 Disjunctive Graph

Another way to represents the job shop problem that also found to be useful is provided by Kan (1976) using the concept of disjunctive graph. Consider a set of n jobs to be processed on m machines. Let disjunctive graph, G with a set of nodes N and two sets of arcs A and B. The nodes N represent the set of operations' node as well as two dummy nodes at the beginning (source node, U) and the end (sink node, V) of schedule. The weight of each node is the processing time for each operation. The source node and the sink node have zero processing time.

The conjunctive (solid) arcs A represent the routes of each job including arcs connecting the source node to the first operation node and the last operation to the sink node. These conjunctive arcs represent the technological precedence. For example,

arc $(j,k) \rightarrow (j,l)$ means that job *j* has to be processed on machine *k* before being processed on machine *l*. In other words, operation (j,k) precedes operation (j,l). In addition, there are *n* number of conjunctive arcs emanating from source node, *U*, to the first operations of each job and *n* number of conjunctive arcs coming from the final operations of each job to the sink node, *V*.

The disjunctive (broken) arcs *B* refer to operation that belong to the different job but have to be processed on the same machine. These arcs form *m* cliques of double arcs. Based on the graph theory, clique is a term that refers to a graph which any two nodes are connected to one another. In this case, all operations in the same clique have to be processed on the same machine. The arcs' length represent the processing time for each operation. We denote this graph with $G = (N, A, B)$. Figure 2.3 shows the disjunctive graph representation for problem in Table 2.1.

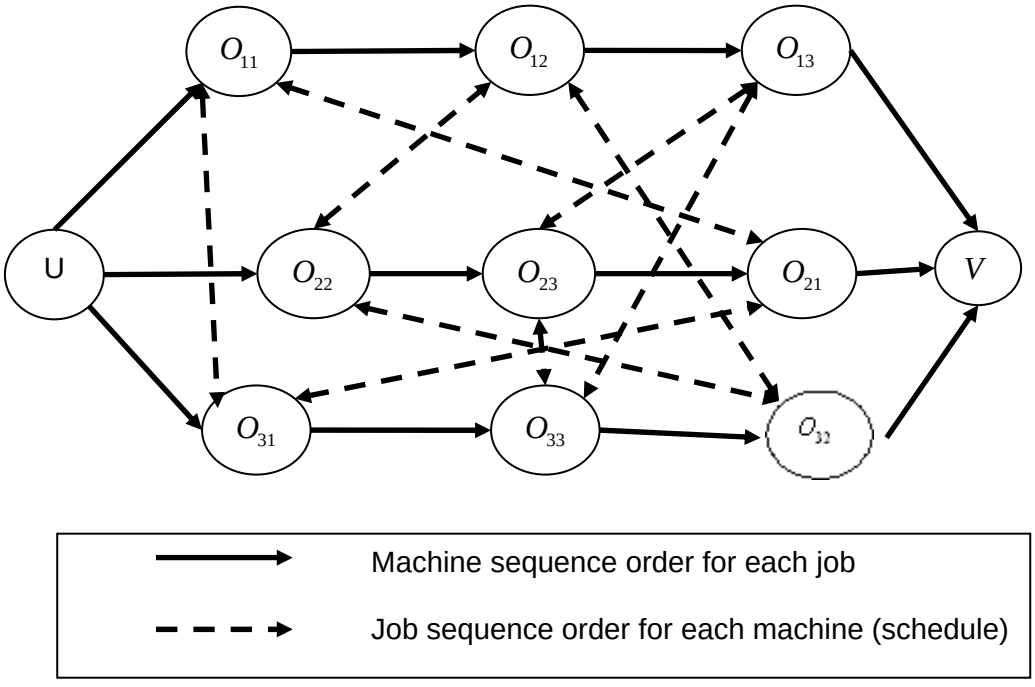

Figure 2.3: Disjunctive Graph for 3 Jobs and 3 Machines Problems

Another term that is important in scheduling a job shop problem is generating a feasible schedule. So far, we have introduced the technological constraint where a job should be processed on each machine in specific order. According to French (1982) schedules which compatible with this condition are called feasible. Otherwise the schedule is known to be infeasible. Obviously, a solution to scheduling problem must be feasible. The easiest way to explain the concept of feasible schedule is from the graph. Assume that we have select one disjunctive (broken) arc from each pair of clique that resulting directed graph is acyclic. If there is a cycle within a clique, it is impossible to find a feasible sequence of the operations on the corresponding machine. For example, let O_{jh} and O_{ji} denote the operations that belong to job j and let O_{ki} and O_{kh} denote the two consecutive operations that belong to job k. If, a given schedule is operation O_{ji} is processed before operation O_{ki} on machine i and operation O_{kh} precedes operation O_{jh} on machine h, then the graph will look like Figure 2.4. Such a schedule is physically impossible.

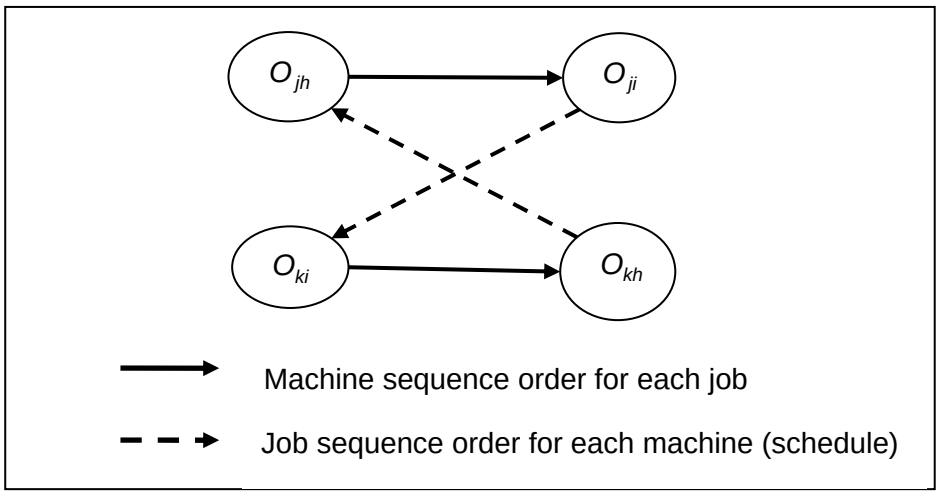

Figure 2.4: A Cycle within a Disjunctive Graph

A selection will determine the sequence of operations which perform on that particular machine. Now, denote that D is the subset of the selected disjunctive arcs

and graph $G(D)$ is correspond to a feasible schedule if and only if $G(D)$ contains no directed cycles. The longest path in $G(D)$ from node source U to the node sink V is called the makespan of a feasible schedule. It consist a set of operations which start at time 0 and finishes at the time of makespan. For example, assuming each operation has processing time equal to one and Figure 2.5 shows the feasible solution created. Two example of the longest path are $U \to O_{11} \to O_{12} \to O_{13} \to O_{33} \to O_{32} \to V$ and $U \to O_{22} \to O_{12} \to O_{13} \to O_{33} \to O_{32} \to V$.

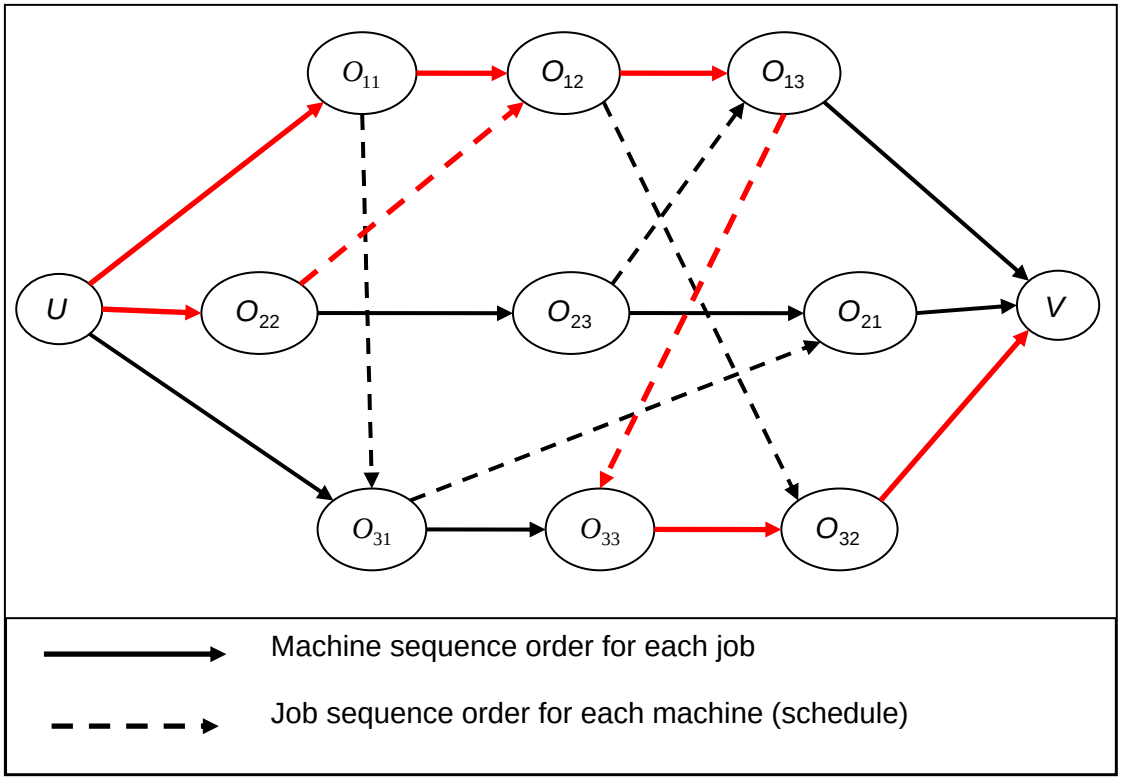

Figure 2.5: Disjunctive Graph Represents a Feasible Schedule

Finding an optimal feasible schedule is equivalent to the finding an orientation of disjunctive arcs that will minimise the longest path in the graph. To compute the starting time S_{jkm} and completion time C_{jkm} for each operation, we introduced C_{jkm} to represent time k-th operation of job j complete it process on machine m. We also introduced ready time r_{mq} of machine m where q is the job order on machine m.

Indeed, for each operation, we have,

$$C_{jkm} = S_{jkm} + p_{jkm} \tag{2.2}$$

where

$$S_{jmk} = \begin{cases} max\{C_{j(k-1)m}, r_{m(q-1)}\} & \text{if } k,q > 1 \\ max\{0, r_{m(q-1)}\} & \text{if } q > 1, k = 1 \\ max\{C_{j(k-1)m}, 0\} & \text{if } q = 1, k > 1 \\ 0 & \text{if } q = 1, k = 1 \end{cases} \tag{2.3}$$

Equation 2.2 computes the completing time for O_{jkm} by summing the starting time, S_{jkm}, and the processing time, p_{jkm}, of O_{jkm}. In addition, equation 2.3 computes the starting time for job j on machine m. As we can see, the starting time is depending on the maximum value between the completion time of the previous operation for job j ,$C_{j(k-1)m}$, and the completing time of previous operation on machine m , $r_{m(q-1)}$.

Creating an initial feasible solution can be obtained by the following steps. First, consider each job in arbitrary order. Then put its operation on the appropriate machine following the technological sequence. This will give a feasible solution which its makespan is not really good. Fortemps and Hapke (1997) suggested that the current scheduled can be modified in order to improve the makespan value. Note that only disjunctive arcs can be reverted since the conjunctive arcs represent technological constraints. In order to get other feasible schedule with a better makespan value, the orientation of some disjunctive arcs are need to be reversed. For example, referring to Figure 2.5, if the operation O_{13} and O_{33} are reverted, this will lead from the sub-sequence $O_{23} \rightarrow O_{13} \rightarrow O_{33}$ to $O_{23} \rightarrow O_{33} \rightarrow O_{13}$. However, according to Fortemps

and Hapke, (1997) not all disjunctive arcs can be reverted without introducing cycle if there already exist in another path. Suppose that if O_{32} is reverted to O_{22}, from original $U \to O_{22} \to O_{23} \to O_{13} \to O_{33} \to O_{32} \to V$. After inversion, there will exist a cycle $U \to O_{22} \to O_{23} \to O_{13} \to O_{33} \to O_{32} \to O_{22}$. By considering the example before, only arc from O_{32} to O_{22} can be reverted. This type of arcs is called 'only arc'.

Testing weather an arc is 'only arc' or not, is time consuming because the search needs to cover the whole graph. To make the search more effective, the search is limited to the critical arcs. Critical arcs are defined as arcs on the longest path and related to machine constraints. Obviously, if an arc $a \to b$ on the longest path, no other path may exist from a to b, otherwise the longest path will go through other path so the weight of additional node will take into account. In addition, if other 'only arcs' which are not critical arcs is reverted, it would not reduce the makespan because the previous longest path will remain in the new solution and the new makespan will be greater or equal to the previous one. In addition, any exchange of two adjacent operations on a critical path will never lead to an infeasible schedule. This fact is shown by Yamada (2003) and also known as feasibility property.

Theorem 2.1: (feasibility for adjacent exchange)

Let S be a consistent complete selection and $P(S)$ be a critical path in S. Consider a pair of adjacent critical operation (u,v) on a same machine on $P(S)$. For example, there is an arc selected from u to v. Then a complete selection S^{uv} obtained from S by reversing the direction of the arc between u and v is always acyclic (thus the corresponding schedule is always feasible).

Proof: Assume the contrary, and then the exchange introduces a cycle in S^{uv}. This means that there is a path $P^{u,v}$ from u to v in S^{uv}, and this $P^{u,v}$ also exist in S. $P(S)$ can be represented as $P(S) = (0,...,u,v,w,...,*)$ and $(0,...,P^{u,v},w,...,*)$ is also a path from source to sink in S but clearly longer than $P(S)$. This contradicts the assumption of the theorem that $P(S)$ is a critical path of S.

Finally, from Figure 2.5, we can see that the makespan is 5 and the longest path is $U \to O_{11} \to O_{12} \to O_{13} \to O_{33} \to O_{32} \to V$. If we reverse arc $O_{13} \to O_{33}$ which is also the critical arcs to $O_{33} \to O_{13}$, we will get the new feasible schedule $U \to O_{11} \to O_{31} \to O_{33} \to O_{13} \to V$ with makespan 4 which is also the optimal solution to the problem. Optimal solution is shown in Figure 2.6.

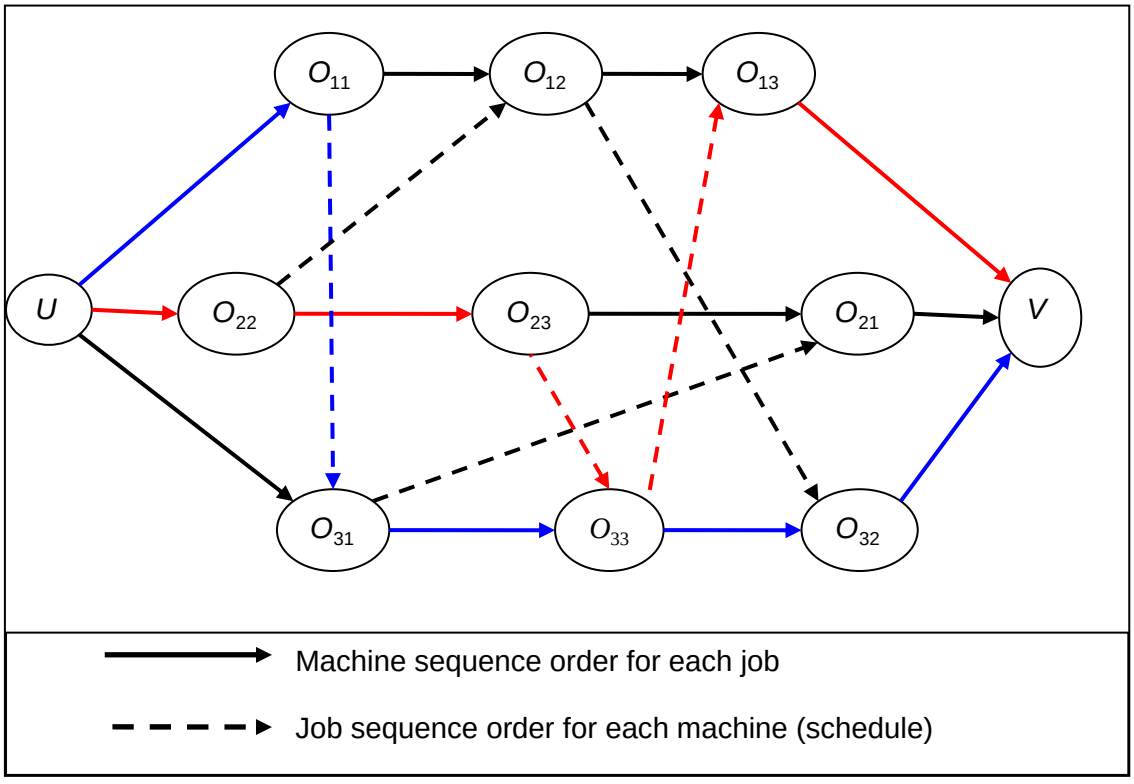

Figure 2.6: Disjunctive Graph Represents a Feasible Schedule (After Reversing an Arc)

2.3.2 Gantt Chart

Apart from the disjunctive graph, another representation for solution to job shop scheduling problem by rearranging blocks representing the given set of operations. The chart is named Gantt chart. Developed by Henry L. Gantt in 1910 to represent the project schedule, nowadays Gantt chart is widely used to represent the solution of job shop scheduling problem. According to Conway et al. (1967), Gantt chart is constructed with a horizontal axis and horizontal bars. Horizontal axis represents the total time span for each operation while the horizontal bars or blocks represent the operations. The length of each block is proportional to the processing time required to perform the operation. The operation blocks are arranged in rows in accordance with the machine it should be processed on and the sequence ordered constructed by the solution methods. Figure 2.7 showed the example of Gantt chart.

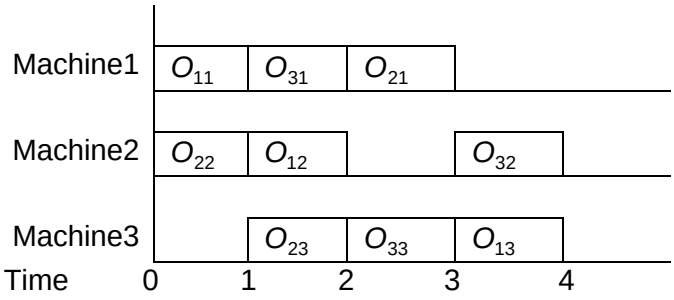

Figure 2.7: Example of Gantt Chart for 3×3 size of Problem

2.4 Solving Methods for JSSP

In 1976, Kan (1976) showed that complete enumeration is not suitable even for a small job shop problems. Each permutation gives the processing sequence of jobs on a particular machine. In addition, French (1982) stated that since each of the permutations may be different from the rest, it follows that the total number of schedules is $(n!)^m$ where n is number of jobs and m is number of machines. However, because of the technological constraints, some of these schedules may be infeasible.

Thus, the number of schedules to be consider is less than $(n!)^m$. French (1982) also noted that job shop scheduling problems are NP-hard and could not be solved in polynomial time.

2.4.1 Integer Programming

The general job shop problem can be modelled as an integer programming problem. In 1959, Wagner modelled general job shop scheduling problems in integer programming formulation. His approach suitable for $n/m/A/B$ problems and represent some variables in $0-1$ form in his constraints. In his model, Wagner represented the variable constraint $X_{ik}=1$ if J_i is scheduled in the k-th position. Otherwise, $X_{ik}=0$. Meanwhile, in 1960, Manne introduced Manne's model. In his model, Manne introduced real variable S_r to represent the starting time of operation r. He also used $0-1$ representation in his constraints. Manne's model of formulation is closely reflects the disjunctive graph structure. Both approaches are viewed by Kan (1976).

However, Conway *et al.* (1967) have shown that computational experience with the models using general $0-1$ methods is very time consuming. According to Pinedo and Chao (1999), the most common formulation used for job shop scheduling problems was proposed by Roy and Sussmann in 1964, known as disjunctive programming formulation. The following formulation described disjunctive programming in term of mathematical formulation due to Roy and Sussmann in 1964. Denote that y_{ij} is the starting time of operation (i,j).

Minimize C_{max}

Subject to

$$y_{jl} - y_{jk} \geq p_{jk} \quad \text{for all } (j,k) \rightarrow (j,l) \in A \tag{2.4}$$

$$C_{max} - y_{jk} \geq p_{jk} \quad \text{for all } (j,k) \in N \tag{2.5}$$

$$y_{jk} - y_{hk} \geq p_{hk} \text{ or } y_{hk} - y_{jk} \geq p_{jk} \quad \text{for all } (h,k) \text{ and } (j,k),\ j = 1,2,...,m \tag{2.6}$$

$$y_{jk} \geq 0 \quad \text{for all } (j,k) \in N \tag{2.7}$$

The first constraint ensure that operation (j,l) cannot start before operation (j,k) is completed. Subsequently, the second constraint defined the value of makespan should be large or equal to the completion time of each operation. The third set of constraints is the disjunctive constraints. They ensure that some ordering exists among operations of different jobs that have to be processed on the same machine.

2.4.2 Enumeration Methods

Though, a scheduling problem can be formulated as a disjunctive programming; it does not imply the availability of a standard solution procedure will work satisfactorily. The number of constraints will increase as the size of the problem increased. Minimising the makespan in a job shop is a hard problem since according to Yamada and Nakano (1997), Problem MT10 sized 10×10 which formulated by Muth and Thompson remained unsolved for over 20 years. Therefore, the solution procedures are either based on enumeration or heuristics techniques.

Enumeration methods consider all the possible schedules and then eliminate the non-optimal schedules from the list, leaving those optimal. According to Pinedo and Chao (1999), the most widely known use in scheduling is branch and bound method.

To apply branch and bound, we must have a partitioning of feasible region for a problem to create smaller sub-problem (branching) and a lower bound on an instance of optimisation problem (bounding). One of the branching procedures that always been used is by limiting the branch to a specific class of schedules.

Bierwirth and Mattfeld (1999) in their paper did some study on classes of schedules. The classes are semi-active schedules, active schedules and non-delay schedules. Semi-active schedules will ensure that each operation will be start as soon as possible while obeying the technological sequence and the processing sequence. On the other hand, active schedule is a schedule with no operation can be started earlier without delaying some other operation or violating the technological constraint. The active schedules form a subclass of the semi-active. In non-delay schedules, a machine do not allowed to be idle if it could start processing some operation. According to them, concerning the minimisation of C_{max} and $\overline{F}$, it is well known that at least one of the optimal schedules is an active one.

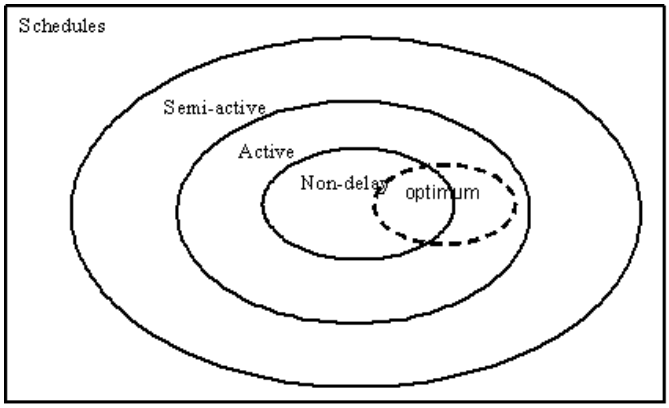

Figure 2.8: Venn Diagram for the Semi-Active, Active and Non-Delay Classes of Schedules

According to Yamada (2003), based on study by B. Giffler and G. L. Thompson in 1960, they suggested that it is not necessary to search for an optimal schedule over all possible schedules, but it is enough to search over a subset of feasible schedules.