Approche de réseaux de neurones récurrents pour le problème de l'ordonnancement cyclique et sa variante

THESE

Présentée et soutenue publiquement le 28 novembre 2008
en vue de l'obtention du

Doctorat de l'Université d'Artois

(Spécialité : Génie Informatique)

Par

Kok Seng LOW

Composition du jury

Président:	Joe CARTHY	Professeur University College Dublin
Rapporteurs :	Nahid EMAD	Professeur à l'Université de Versailles S^T Quentin-en-Yvelines
	Ouajdi KORBAA	Professeur à l'Université de Sousse
Examinateurs :	Tahar KECHADI	Professeur University College Dublin (Directeur de thèse)
	Gilles GONCALVES	Professeur à l'Université d'Artois (co directeur de thèse)
	Remy DUPAS	Professeur à l'Université de Bordeaux 1 (co-encadrant de thèse)
Invité :	Michela BERTOLOTTO	Professeur University College Dublin

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Scheduling deals with the allocation of tasks requiring processing to limited resources over time. The scheduling problems arise among others, in areas of product manufacturing, computer processing and transportation. In this thesis we review the properties of both the general scheduling and cyclic scheduling problems with a focus on the cyclic version of the problem.

As the cyclic scheduling problem is NP-Hard complexity, in the worst case scenario, the time required to solve the problem is exponential time. This difficulty has motivated this research work in developing an efficient neural network approach to solving the cyclic scheduling problem.

This thesis focuses specifically on the cyclic job shop and cyclic flexible manufacturing system problems. Hence, models that solve the minimum cycle time or work in progress of the problems are developed. These models are fundamental to which the neural network approach can be applied.

In existing literature, the absence of neural network research into solving the scheduling problem is due to its characteristics such as complex architecture, defining initial conditions, difficulty in tuning its parameters (i.e. learning rate, stoppage conditions, etc) and tendency for infeasible solutions. However, in this thesis, we develop and study three variations of the recurrent neural network approach. These are the Recurrent Neural Network (RNN) approach, the Lagrangian Relaxation Recurrent Neural Network (LRRNN) approach and the Advanced Hopfield network approach. Several algorithms are combined with these neural networks to ensure that feasible solutions are generated and to reduce the search effort for the optimum solutions.

A Competitive Dispatch Rule Phase (CDRP) is developed to generate initial feasible solutions before the three recurrent neural network approaches are initiated. This is important as the search space of the problem can be reduced through this approach. For the cyclic flexible manufacturing system problem, a Modified Competitive Dispatch Rule Phase (MCDRP) is developed. This ensures the best possible cyclic schedule with minimum work in progress, for the neural network approaches to work from. As the solutions may be trapped in local minimum energy state, a schedule perturbation phase is developed to "kick-start" the search effort. Finally, using the developed schedule Postprocessing phase that contains the Adhere Conjunctive and Adhere Disjunctive algorithms, the subsequent final solutions are always a feasible schedule.

We also extend the review into the cyclic job shop problem with linear prece-

dence constraints. A Delinearization algorithm is developed to solve this problem; an approach based on transforming the linear constraints of the problem into its uniform constraints as proven in existing cyclic scheduling literature.

We are able to demonstrate the suitability and applicability of the RNN, LR-RNN and Advanced Hopfield network approaches through computational and comparative testing. The experimental results indicate that the three approaches are attractive alternatives to traditional heuristics in solving cyclic scheduling problems, even though in some cases it is computationally expensive.

ABSTRACT

Un problème d'ordonnancement consiste à exécuter sur un horizon de temps donné un ensemble de taches au moyen de ressources en nombre limité. On rencontre ce problème dans divers domaines, comme l'industrie de production, dans les systèmes de transport ou encore dans les ordinateurs avec l'allocation des tâches. Dans cette thèse, nous nous concentrons sur le problème d'ordonnance cyclique. Les propriétés liées à l'ordonnancement en général, ainsi qu'à sa version cyclique seront étudiées.

Le problème d'ordonnancement des tâches est un problème NP-complet, le temps nécessaire pour le résoudre peut être exponentiel dans le pire des scenarios. Ceci a motivé notre travail de recherche et nous a mené à développer une approche efficace utilisant les réseaux de neurones pour le résoudre.

Cette thèse se concentre particulièrement sur le problème du Job Shop Cyclic et sur son utilisation dans le cadre des ateliers flexibles (FMS : Flexible Manufacturing System). Pour cela, nous avons développé deux modèles de résolutions basés sur les réseaux de neurones. Le premier a pour objectif de minimiser le temps de cycle et le second a pour objectif de minimiser les encours de production. On remarque une absence de travaux utilisant les réseaux de neurones sur ce type de problème dans la littérature scientifique. Ceci est dû à l'architecture complexe des réseaux de neurones, à la difficulté de définir les conditions initiales, au réglage de ses paramètres (taux d'apprentissage, condition d'arrêt, etc.) ainsi que sa tendance à générer des solutions impossibles. Néanmoins, dans cette thèse, nous proposons 3 variations autour des réseaux de neurones récurrents: un réseau de neurones récurrents (RNN), une relaxation Lagrangienne pour un réseau de neurones récurrents (LRRNN) et un réseau Hopfield avancé. Plusieurs algorithmes sont combinés avec ces réseaux de neurones pour assurer que les solutions générées sont toutes possibles et pour réduire l'effort de recherche des solutions optimales.

Une phase prétraitement CDRP (Competitive Dispatch Rule Phase) est imcorponée pour générer des solutions initiales correctes avant que les 3 réseaux de neurones récurrents soient initialisés. Celle-ci permet de réduire l'espace de recherche du problème. Pour le problème du FMS cyclique, une phase modifiée (MCDRP) est appliquée. Pour s'échapper des optimums locaux, une phase de perturbation est ajoutée pour relancer la recherche dans une autre région de l'espace de recherche. Enfin, une phase de post traitement assure que les solutions finales sont toujours dans l'espace des solutions possibles.

Dans une première partie, nous avons étendu nos travaux au problème du job shop cyclique avec des contraintes de précédence linéaire. Selon la littérature en ordonnancement cyclique, il est possible de transformer les contraintes linéaires en contraintes uniformes équivalentes, c'est pour cette raison qu'un algorithme de délinéarisation est mis emplace pour permettre de traiter ce problème avec notre approche lagrangienne précédente.

Nous sommes capable de démontrer la conformité et l'applicabilité des approches utilisant des réseaux RNN, LRRNN et Advanced Hopfield à travers une évaluation comparative. Les résultats expérimentaux indiquent que les 3 approches proposées sont des alternatives attrayantes par rapport à d'autres approches heuristiques traditionnelles même si parfois celles-ci restent coûteuses en terme de calcul.

To the memory of my parents.

ACKNOWLEDGEMENTS

This research would not have been possible without the enormous support and guidance of many people in my life.

I am especially grateful to my supervisor Dr. Tahar Kechadi for his excellent guidance, supervision, advice and attention to detail.

I would also like to to thank my co-supervisors Prof. Gilles Goncalves, Prof. Remy Dupas (currently in Université Bordeaux 1) and Prof. Tiente Hsu from Université d'Artois for the opportunity to collaborate and work with LGI2A in Bethune, France. All the interesting discussions and advices have helped me in pursuing new ideas and perspective in researching this subject.

I would also like to thank the following honourable members of the jury that have agreed and participated in reviewing my thesis: Prof. Ouajdi Korbaa (Université de Sousse), Prof. Nahid Emad (Université de Versailles ST Quentin-en-Yvelines), Prof. Michela Bertolotto (University College Dublin) and Prof. Joe Carthy (University College Dublin).

I would also like to thank all my friends in the Parallel Computational Research Group - Tony Fitzgerald, Liam McDermott, Jacques Bellec, Alex Cronin, Tariq Noor Ellahi, Ray Genoe, Benoit Hudzia, Tarik Rashid, Guillem Lefait for all their support and interesting encouragement during my years of pursuing my research. I am also grateful for all the helpful and supportive staff in School of Computer Science and Informatics as well as in University College Dublin for the experience of working with and researching in such a supportive institution.

Finally, I am extremely grateful to Claire for all her love, support, encouragement and faith in my abilities and for being there for me. Not forgetting Pat and Breda where their generosities must also be mentioned.

LIST OF PUBLICATIONS

- K.S. Low and M-T Kechadi, "Application Of Improved Hopfield Network To Cyclic Task Scheduling". In the 12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2006), Saint-Etienne, France, May 17-19, 2006.
- K.S. Low and M-T Kechadi, "Evaluation of a Neural Network Approach on FMS Cyclic Task Scheduling Problem". In the Artificial Intelligence & Applications the IASTED International Conference (AIA 2005), Innsbruck, Austria, February 14-16, 2005.
- K.S. Low, M-T. Kechadi, T. Hsu, R. Dupas and G. Goncalves, "A Recurrent Neural Network Based Approach for Flexible Manufacturing Scheduling Systems". In the International Conference on Industrial Engineering and Systems Management (IESM 05), Marrakech, Morocco, May 16-19, 2005.
- K.S. Low, M-T. Kechadi, T. Hsu, R. Dupas and G. Goncalves, "A Neural Network Model for Flexible Manufacturing Cyclic Scheduling Systems". In the 23rd Annual Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG 2004), Cork, Ireland, December 20-21, 2004.
- K.S. Low and M-T. Kechadi, "Cyclic Task Scheduling in Manufacturing Production Line Using Neural Networks". In the IEEE International Conference on Systems, Man and Cybernetics (SMC 02), Hammamet, Tunisia, October 6-9, 2002.

Part I

Background

CHAPTER

ONE

Introduction

1.1 Scheduling in general

In real-life situations, the usage of time and the activities associated with time have different meanings to each person or environment. Proper utilization of time is obviously more important for some people or environments compared to others. High utilization of time can result in productive activities leading to excellent returns. These returns may include higher income generated, more time available for other activities after certain activities are completed, or even longer rest-time, and less stress. On the other hand, low utilization of time may prevent us from obtaining the best results and returns possible in a given window of opportunity. Lost opportunities are often quoted when time is not utilized properly.

Before we can utilize a process to its fullest potential, some planning is usually required. Planning should involve stating most or all of the activities that are available to be completed, the actual time length to complete the activities and possibly preferences of order to undertake the activities. Sometimes other consideration may also be taken into account e.g. important requirements or skills or equipment needed and even information related to the activities.

However, when the environment involved is manufacturing, high utilization of time is vital. With the huge investment values, high cost of machinery and potentially high cost of labour, proper use of time is very important to guarantee the highest possible return on investment (ROI). The planning stage in this manufacturing environment is formally known as scheduling.

The activity of scheduling can be done manually in the case of small manufacturing size, or simple large jobs on a small number of resources. A scheduling specialist who has honed his or her scheduling skills from years of experience

could probably also do scheduling manually. However, in the case where a large number of jobs are involved, close due dates for every completed component is required or complicated rules are needed to handle changes in job processing, more sophisticated techniques are required. Currently, a variety of scheduling softwares are predominantly available. A majority of the scheduling software have large generic features that have to be customized according to the environment in which it will be utilized.

Research into scheduling has spanned from scheduling theory to applying these research theories into real-life manufacturing floor, service sector and various computing applications. The scheduling problem has been a researched area since 1950s [130]. As part of the combinatorial optimization problem, there is much interest in this area. The scheduling problem exists not only in particular types of manufacturing environment, but also in the service sector and areas of computing research.

Scheduling is defined as sequencing or ordering operations or tasks over a certain length of time [50]. It involves setting the start times and end times for the tasks. To achieve or obtain this particular sequence, a decision making process is required. Vital information regarding the system is required in the decision making process. This will include characteristics of the operations or tasks and the environment in which operations are to be undertaken. This scheduling is done on limited and available number of resources, based on certain restrictions or constraints. These constraints may involve the resources or even constraints on how the operations or tasks should be sequenced. Solving the scheduling problem has an objective or a number of objectives to fulfill.

The tasks are operations or activities that need to be executed. Examples of tasks are assembly jobs in a factory, processing clients' requests in a bank (in a service sector), allocation of departure times for a flight in an airport, or computing computer programs on a processor. Associated with each task, is its properties. These properties may include its priority level, the instance the task is available or should be completed.

The resource type differs according to which area the scheduling problem is focused on. It includes machines in manufacturing environments, vehicles in transportation systems, a particular labour in a service sector, or even processors for computing.

The constraints that could encompass either the tasks or resources are the order which the tasks are to be processed, restrictions on when and how the

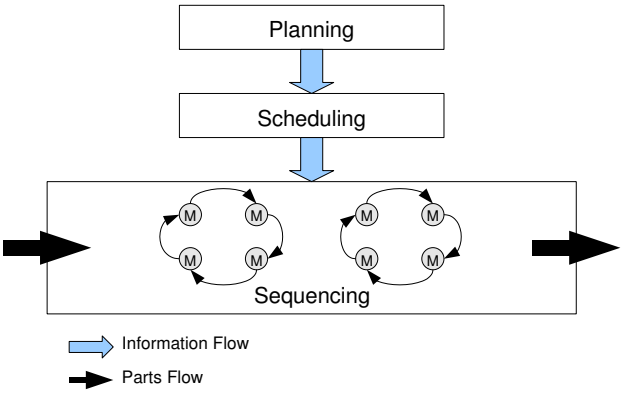


Figure 1.1: Planning, Scheduling and Sequencing on Production System

resources are available or even limitations of usage of the resources over a limited time period.

When comparing planning and scheduling, planning involves *what* must be done and the *restrictions* on methods to be used, whereas scheduling involves *how* the activities must be executed and *when* it must be done. Planning uses the estimates of the time and resources or skills required to complete the activities. Other information needed could include precedence relationships between jobs to be done. For scheduling, it is the temporal assignments of all the tasks and activities that must be executed for the plan which is important. Planning is measured through the feasibility of the plan and scheduling through its performance. An example of how planning, scheduling and sequencing interact can be seen in Figure 1.1.

Coming to the different terms associated with the scheduling problems, terms like sequencing, scheduling and scheduling policy are widely used. However from *Pinedo (2002)* [132], some distinctions related to the use of the terms sequence, scheduling and scheduling policy have been described. *Sequencing* involves the different permutations related to the order which the N number of tasks must be loaded onto the machine [36]. In contrast, *scheduling* involves much more complicated settings in addition to negotiating the sequencing process and the tasks that may be stopped during the processing period. This may be due to machine breakdown, or unavailability due to preventive maintenance tasks. The use of *scheduling policy*, is more evident in a stochastic environment when various parameters of the tasks or machines are of stochastic values. The policy dictates rules of action on how to sequence the tasks which includes *shortest processing time (SPT)*, *longest processing times (LPT)*, etc.

1.1.1 Why is Scheduling Difficult to Solve?

One main reason for this is the amount of data associated with the problem. Not only do we have to figure out what to schedule, when to schedule the tasks or activities but also on which resource to schedule the tasks on. Considering the fact that with the large number of resource available in actual real-life situations, this can be tedious. The simplest problem to schedule is obviously a single job on a single resource.

Also the problem gets more complicated when other factors like temporal relationships between tasks, time instance which the parts are released to be processed, due time of the whole product and also the availability of the resources over the time period being considered. So the larger the problem, the more data or information it involves, and the more difficult it is to solve the scheduling problem. From *Conway et. al. (1976)* [36], scheduling problems are proven to be NP-hard.

When we factor in the uncertainties that exist in machine processing environment, this leads us to the second main reason. Uncertainties or stochastic nature in the production system that greatly affect the scheduling include breakdown of machines, failures of tools, cut in supplies, abrupt change or cancellation of orders, changes in due dates or even variable processing time for incoming tasks. Some researchers have factored this in as stochastic models [160], [134], [91]. These unpredictabilities have an impact on the schedule thus requiring changes, making the whole scheduling process dynamic. Most researchers call this *dynamic scheduling*.

Changes in schedule may involve the substitution of resources, or delay in start times of tasks or even worse, the whole reformulation of the schedule. Obviously the most ideal case to approach these disturbances is a scheduling model that can adapt to any changes while maintaining the objectives of the process.

Infeasibilities within the schedule found can make the scheduling difficult. With large problems, depending on the model used, and assumptions, a feasible schedule may or may not exit. Constraints that restrict the scheduling problem sometimes also limits the possibility of feasible schedules. So, solving the scheduling problem depends greatly on the methods and algorithms used in the model, plus various other information that are associated with the machine processing environment.

To reflect on how difficult the scheduling problem is, assuming the existence

of N jobs to be scheduled on a single machine. The N jobs on a machine has $N!$ possible sequences. For example for a mere 20 jobs to be scheduled on a machine, this means that there are 20! distinct permutations to schedule the jobs (i.e. $2.43290201 \times 10^{18}$ distinct permutations).

In a manufacturing environment, where product deliverability is vital, late deliverability has an effect on the cost. The product manufactured must not only be of high quality, and low priced, its short requested lead time and on-time delivery and production rate is vital. With the manufacturing resource available to process the operations, scheduling will improve and maintain a high level of productivity and efficiency. Scheduling is also used primarily to cope with frequent change in product specifications. As such, the use of scheduling in an manufacturing environment will guarantee this. Obviously, this also depends on the capabilities of the type of resources available and the capacity of the manufacturing location itself.

The development of a schedule involves selecting a sequence that will guarantee that all required tasks are processed. It also involves designating the required resources needed and the appropriate times to start and complete the processing of each individual task.

1.1.2 Impact of good scheduling

A good, feasible schedule can very much impact on production costs such as variable production and overtime costs, inventory holding costs, penalty costs associated with missing deadlines, and possible expediting costs for implementing the schedule in a dynamic environment. An efficient scheduling approach may result in high resource utilization. This is especially true for many production systems having resources with limited capacity, and every hour's utilization contributes to the production system's efficiency. The quantity of work in progress requiring storage space of facilities could potentially be kept to a minimum or even totally eliminated with a good schedule. The chances of lateness can be reduced or eliminated too, as proper scheduling will accurately estimate completion time for all the jobs requiring processing. This will guarantee timely delivery of orders.

1.2 Research Objective

Our research has been motivated by the challenges in solving cyclic scheduling problems that, as will be discussed in subsequent chapters are complex to solve.

Our research in this thesis will highlight the work done in developing an optimal plan for cyclic scheduling of jobs in a manufacturing system. The manufacturing system here covers mainly the cyclic job shop and the cyclic flexible manufacturing systems environment. Certain assumptions and modelling of the real-life events are necessary to capture the real scheduling problems. We provide a mathematical model of the process plan or the scheduling problem. Using rules that are considered constraints, we include this in the formulated optimal plan. The plan will utilize the resources in the most optimal manner. Using this plan, the neural network is used to generate cyclic schedule solutions. An extension of the neural network is introduced, explained and proved. The cyclic schedule is feasible and optimal in relation to the objectives of the manufacturing system and the utilization of the resources.

In summary, the main proposals that this thesis will aim to present are:

- Exploring types of cyclic scheduling problem and associated elements in it;
- Proposing new models and formulations for cyclic scheduling problem, specifically cyclic job shop and flexible manufacturing system for mapping to a dynamic recurrent neural network architecture;
- Developing new dynamical Recurrent Neural Network that correlates to the new formulation of the cyclic scheduling problem;
- Extending the Recurrent Neural Network into the Lagrangian Relaxation Recurrent Neural Network to improve the approach in solving the cyclic scheduling problem;
- Extending our approach and modelling in solving the extended version of the cyclic job shop cyclic scheduling problem with linear precedence constraints;
- Developing an Advanced Hopfield Network to solve the cyclic flexible manufacturing systems scheduling problem;

- Developing hybrid solutions by integrating specific algorithms to these new approaches in order to escape from local minima and attaining most feasible and optimal solutions;
- Evaluating the computational complexity with qualitative and quantitative analysis of these approaches.

1.3 Organization of this thesis

This thesis aims to give an overview of the scheduling problem and further delve into the cyclic scheduling problem in the first part of this thesis. We have already looked into and discussed the general description associated with the scheduling and how scheduling can be applied in this ***Introduction Chapter***.

We will discuss how the scheduling problem has evolved, the inner properties of the scheduling and how it has been categorized from previous researchers in ***Chapter*** 2. This chapter will include all the major entities making up the scheduling problem and the cyclic scheduling problem, that reflects the real-life environment. We will discuss the different parameters in the cyclic scheduling problems compared to non-cyclic scheduling problems and various known classes. We then show how the various entities and properties can be modelled mathematically, the technical side of the scheduling problem and why the cyclic scheduling problem is difficult to solve in ***Chapter*** 3. The modelling of the cyclic scheduling problem is vitally required before any approach can be applied to these problems. The latest development in approaches used to solve the cyclic scheduling problem is also discussed. In this chapter, we also describe and review the cyclic scheduling problem of cyclic job shop scheduling problems (CJSSPs) where we propose a modelling approach with the main objective of minimizing the cycle time related to the problem. We then review the cyclic flexible manufacturing systems scheduling problems (CFMSSPs) and a proposed modelling approach. The differences in modelling this particular cyclic scheduling problem are explained while our approaches are aimed at achieving an accurate modelling of the constraints and objective functions.

We then introduce the neural networks (NN) as an approach to solving this scheduling problems. Different and popular types of artificial neural network are discussed and their potential in giving efficient results are discussed. We then present the evolution of research into neural network, used to solve the

scheduling problems.

Moving into the second part of this thesis in ***Chapter*** 4, we then present the dynamics of the Recurrent Neural Network and Lagrangian Relaxation Recurrent Neural Network that encompasses the preprocessing and postprocessing phases. We describe in detail the dynamics and architecture of the network applied to the cyclic job shop and flexible manufacturing systems. We also study the cyclic job shop problem with linear constraints and propose a modelling and delinearization approach in solving this special type of CJSSP. We further extend solving these cyclic scheduling problem with the developed Advanced Hopfield network technique.

Our proposed approaches are simulated on various benchmark problems to ascertain the accuracy and viabilities of the modelling techniques and neural network approaches in ***Chapter*** 5, and this thesis concludes with a discussion on our contributions and findings.

CHAPTER

TWO

State-Of-The-Art Review

As many research works have been directed into the scheduling arena, a classification of the scheduling problem is important. This chapter highlights the differences in each type of scheduling problem and where the importance is placed. This will also track the growth and the research into solving the scheduling problem in the past years, where their feasibilities and their viabilities can be placed in each problem.

From *Conway et. al. (1967)* [36], classification of the scheduling problem is primarily described, based on:

i. the jobs and operations to be processed,

ii. the number of machines and types of machines in the environment ,

iii. the restrictions or constraints that surround the manner in which the processing should be done,

iv. criteria or objective which the schedule can be evaluated.

Using the above classification, a scheme comprising of four fields has been used to describe the scheduling problems - $A|B|C|D$, where A denotes the number of jobs in the problem, B denotes the number of machines in the problem, C determines the machine environment set-up and D is the criterion to be minimized. Table 2.1 shows the different descriptions for each parameter.

An alternative to the above scheme, is the three parameter scheme from *Graham et. al. (1979)* [59] described as $\alpha|\beta|\gamma$. This less compact notation scheme describes α as the machine environment, β as the various job characteristics and γ as the criterion to be minimized. Notations in Table 2.2 play an important part in defining various basic and non-basic scheduling problems used by many researchers.

Number of jobs, A	
Z^+	Positive number of jobs
n	Variable number of jobs
Number of machines, B	
Z^+	Positive number of machines
n	Variable number of machines
Machine environment, C	
o	Single machine
J	Job shop
F	Flow shop
P	Permutation Flow Shop
D	Mixed shop
O	Open shop
Objective to minimize, D	
C_j	Completion time
C_{max}	Makespan
L_j	Lateness
T_j	Tardiness
E_j	Earliness
F_j	Flowtime
W_j	Waiting time

Table 2.1: Four parameters scheduling scheme

Cavalieri et. al. (2007) [26] formed a recent classification framework to benchmark the scheduling approach. This relates to the following models:

Production System Models : This is described as *production resources* in the system that include production components of types processors (machining stations), storage (buffers), transporters (transportation systems) and human operators. This also categorized the *process plan* of how products are processed in the plants with the models sequence of operations for a given product, composed of several operations or only one object (tasks). *Production Planning* can be long term (or Production Planning) of time length 6 months to one year, mid-term (or Work Order) which are monthly or the weekly short-term (or Job based).

Manufacturing Scenarios : This relates to the collection of events or dependent activities in the dynamic behaviour of the manufacturing domain.

Plant scenarios : This relates to the dynamic behaviour of production components that includes machine breakdown, stochastic variation in setup times and operations processing times, stochastic variations on transport

Machine Environment, α	
o	Single machine
P	Identical parallel machines
Q	Uniform parallel machines
R	Unrelated parallel machines
PMPM	Multi-purpose machines with identical speed
QMPM	Multi-purpose machines with uniform speed
G	general shop machines
X	Mixed shop of job shop and open shop
O	Open shop
J	Job shop
F	Flow shop
Job Characteristics, β	
pmtn	pre-emption allowed
prec	Precedence constraints exist between jobs
intree	Precedence constraints structured as rooted tree with outdegree for each vertex at most one
outtree	Precedence constraints structured as rooted tree with indegree for each vertex at most one
r_j	Release date specified
$\tau = 1$	each job has a unit of processing time
$a \leq \tau \leq b$	each job has processing time bounded by a and b
Objective to minimize, γ	
C_j	Completion time
C_{max}	Makespan
L_j	Lateness
T_j	Tardiness
E_j	Earliness
F_j	Flowtime
W_j	Waiting time

Table 2.2: Three Parameter Scheduling Scheme

service time and also material arrival time.

Operational scenarios : In this scenario, the condition on how product release is conceived, production order to be used, the expected release time, expected due dates and others (product cost, quality) operational issues e.g. product cost and quality.

2.1 Characteristics of Scheduling Problems

2.1.1 Characteristics of Tasks

In this thesis, the terms: ***task***, ***operation*** and ***job*** are commonly quoted. Here, a *job* J comprises of N *operations* $O_1, O_2, \cdots, O_N$. So to complete a job, all its operations must be processed. This paradigm is more obvious in product manufacturing which requires sub-components to be assembled together in a particular sequence. For example the job of assembling an automobile requires sub-component of parts such as windscreen, doors, seats, etc. The complexity of completing a job is influenced largely by the number of operations required and its associated individual requirements. However the term *task* is used generically to describe a process on its own, without any subtask required.

So a system that requires scheduling, may have N tasks or N numbers of jobs $(J_1, J_2, \cdots, J_N)$, each with a number n_i of operations $(O_{i;1}, O_{i;2}, \cdots, O_{i;n_i})$ where $i \in (1, 2, \cdots, N)$.

The next few subsections will identify various parameters that may be associated with the tasks or operations.

2.1.1.1 Processing Time of Tasks

The processing time is one of the main parameters of tasks that must be known before any scheduling can be done. Sometimes, this can also be known as the processing requirements. We denote p_i as the processing time of task i.

Most processing time are known in advance, deterministic and fixed. However some scheduling problem may involve processing times that are not known prior to start times, hence dynamic or stochastic processing time occurring is considered.

2.1.1.2 Release Time of Tasks

Another common parameter of tasks is the release time, r_i. This is the time at which a particular task is available to the system. The advanced knowledge of release times are vital for scheduling, as the earlier a tasks is available the earlier it can be loaded onto a resource to be processed. Consequently idle time will incur on the resource, awaiting for the task that has yet to be released.

2.1.1.3 Due time of Tasks

Due time, d_j is the predefined time requirement set by a customers, by a job or task. This can be described as the expected time customers should receive the completed product or the shipping time of the product to the customers. Penalties are normally applied to any delay that violates the due time.

2.1.1.4 Preemption of Tasks

One of the other characteristics associated with a task being processed by a resource is whether preemption is allowed or not. If a task is non-preemptive, this means that the task, once started on a resource, cannot be stopped until it has been completed. In other terms, it means that no interruption is allowed once the processing of the task has started.

Task preemptiveness is common when preventive maintenance is scheduled on a resource, with three cases existing i.e. *resumable*, *semi-resumable* and *non-resumable*. In all cases, the task has to be stopped before it is completed due to the resource being taken offline, and is resumed again on the same resource at a later time. A preemptive task may also be allowed to be resumed on another available resource. This is known as *resumable*. In the *semi-resumable* case, the task is partially restarted from the beginning after the resource becomes available again. However in the *non-resumable* case, the tasks has to be completely restarted from scratch if it was stopped its completion. In all cases of preemption, the task may be stopped and started several times before the task is completely processed.

2.1.1.5 Others

The *non-delay* or *no wait* characteristics may exist in the processing of a task once the task has been started on a machine/resource. This is common in

manufacturing environment where the parts must be continuously processed until completion. This will include moving parts between various machines. An example of this case is the processing of copper wire that must be kept heated at a high temperature while is is rolled through different machines to its required diameter. Having any delay between operations will greatly affect the processing abilities and quality of the end product. This characteristic of no wait is commonly denoted by parameter *nwt*.

Recirculation is one of the minority of characteristics associated with tasks, when a task must be processed more than once in a particular machine. *Order cancellation* on jobs may also be considered where jobs are allowed to be taken off the production line before the jobs are fully completed. The common term of *weights*, associated with tasks is common where a priority status exists on the tasks to be processed. The weights correlate to the importance of the task, that is to be considered when selecting between tasks for processing on a particular machine.

2.1.2 Characteristics of Machines

The resources are the key elements used to process the tasks. In the manufacturing environment, machines are the resources. Some machines may only be able to undertake a particular operational ability e.g. etching, cutting, soldering or assembly. Whereas some machines may be fitted with tools from a particular set allowing several operational abilities on the same machine. *Multi-purpose machines (MPM)* have been used in flexible manufacturing systems (FMS) and can handle various but not unlimited operations, studied in [86], [81], [87].

In order to move the parts between machines, a part transport device or system is needed. This *material handling system (MHS)* can include human operator, conveyors, robots or automated vehicle. It is common that some sort of finite storage known as *buffers* is available too, to hold work in progress between machines and to prevent deadlock in the whole production system. *Smith et. al. (1999)* [143] included the modelling of a material handling system in the job shop scheduling problem. By including the material handling activities, the *complete job shop* problem is solved directly or indirectly by solving the job shop scheduling first followed by scheduling the material handling activities. Many scheduling problem solved simplifies and reduces the complexity of the scheduling problem by ignoring material handling activities thus known as

truncated job shop.

2.1.2.1 Setup Times

Setup in machines occurs when time is required to prepare the machine, process or product parts prior to starting the processing phase. This also includes obtaining the correct tools, returning unused tooling, machine cleanup, setting up the required fixtures, inspecting completed parts and positioning work in process material into the machine. Most researchers, in order to simplify the modelling of the scheduling problem, have for long considered setup time to be negligible and normally considered it as part of the processing time. *Allahverdi et. al. (1999)* [9] comprehensively reviewed the types of setup times. The reviewed setup times included sequence-independent and sequence-dependent setup, batch and non-batch setup times and how the setup times relates to the shop environments of a single machine, parallel machines, flow shops, etc.

2.1.2.2 Machine Breakdowns

Breakdowns occur on machines/resources due to faults that commonly involve breakage of internal working mechanisms. Quite rarely, breakdowns may also be due to human error. Factors like the length of time the machines has been in use, frequency of use and whether preventive maintenance has been regularly done on the machines, greatly affect occurrences of breakdowns. When the machines do breakdown, any work in the form of tasks or operations being processed must be unloaded and delayed. Thus the tasks must be preempted, forcing it to be relocated and resumed on another available machine. This unavailability of the machine may jeopardize the completion time of the overall job. According to *Albers et. al. (2001)* [7], there exists two types of machine breakdowns: *permanent breakdown* where machines do not recover at a later date and *transient breakdown* where machines are fixed and are operational again after breakdown. These machine breakdowns may be known in advance (which is extremely rare) called *offline settings* while in *online settings*, the machine availability is not known completely prior to starting the scheduling process. The term *lookahead* is commonly used to describe the scheduler's ability to anticipate with some accuracy, when the unavailabilities will happen.

2.1.3 Characteristics of Objectives

A measure commonly used to evaluate a schedule is known as *objective function*. These objective functions are always a function of the completion time of the jobs. With the scheduling problems, researchers have aimed to minimize or maximize a predefined objective function. Objective functions are categorized as Regular Objective Functions and Non-Regular Objective Functions.

Regular Objectives Functions: As a function of the completion time, C_i the function can only increase if at least one of the completion times increases. From [36], considering regular objective function as follows:

$$M = f(C_1, C_2, \ldots, C_n) \tag{2.1}$$

hence if $M' = f(C'_1, C'_2, \ldots, C'_n)$, then

$$M' > M \qquad \text{only if} \qquad C'_i > C_i \quad \text{for at least one } i, \quad 1 \leq i \leq n \tag{2.2}$$

This regular objective is used in schedules to define that the first schedule's completion time is no later than the second schedule's completion time. The regular objective characteristic will imply that the first schedule is at least as good as the second. This should include maximum or average completion times, flow times, lateness and tardiness. However, objective functions containing earliness, instead of completion time, is not considered regular from *Brucker (2001)* [22].

Non-Regular Objective Functions: The function increases with decreasing completion time parameters. The function only occurs when combining a regular objective and a non-regular objective e.g. total of tardiness and total of earliness.

2.1.4 Characteristics of Constraints

These are rules that affect the scheduling problem and determine how elements in the scheduling problem interact. Constraints in scheduling can be categorized into activities associated constraints, temporal constraints and machine related constraints.

Temporal constraints relate to the time frame rules for the delivery and processing of the operations. This normally can be in the form of earliest and

latest of the start and end times, and the fixed or variable processing times of the operations. Precedence constraints that dictate the order required for processing of the operations are part of this class. The extension of the precedence constraints are precedence constraints with minimal delays and precedence constraints with fixed delays.

Machine related constraints can be divided into capacity related and synchronization constraints. In the capacity constraints, the machine has predefined capacity that cannot be breached. This depends on various conditions e.g. periods for which a resource is unavailable or forbidden states due to closure. In the case of synchronization constraints, is when a certain task is executed on a particular machine, a corresponding machine must be activated to process an associated task.

2.1.5 Characteristics of Schedule

The schedules generated from the scheduling process may be improved based on whether the operations can be shifted to an earlier starting times or dates. This is required in order to optimize some objectives under certain constraints. There are two types of shifts possible, the *local shift* allows for a schedule shift while preserving the operation sequence while the *global shift* will change the operation sequence but do not delay any other operations. Types of schedules according to [132] are:

Inadmissible schedules - excessive idle time in the machine

Nondelay schedules - no idle time on the machines when there are operations waiting to be processed. This implies that the machines should be busy if there are any operations in waiting.

Active schedules - a feasible schedule cannot exist or be improved by moving the tasks forward or backward resulting in task finishing earlier and no tasks finishing later. This schedules cannot be improved without delaying other operations in the schedule.

Semi-active schedules - a feasible schedule in which no operation can be completed earlier without changing the order of the processing. This also mean that the schedule cannot be improved by shifting operations to the left of the schedule.

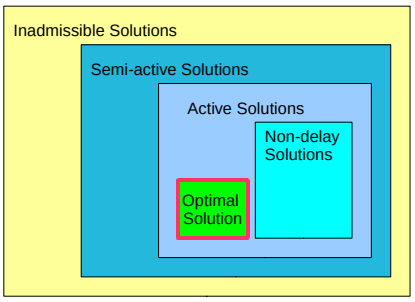


Figure 2.1: Types of Feasible Solutions

Operations with the associated processing times								
	Job 1			Job 2			Job 3	
O	M	P.T.	O	M	P.T.	O	M	P.T.
$O_{1,1}$	$M1$	6	$O_{2,1}$	$M1$	1	$O_{3,1}$	$M1$	5
$O_{1,2}$	$M4$	8	$O_{2,2}$	$M2$	3	$O_{3,2}$	$M3$	5
$O_{1,3}$	$M3$	9	$O_{2,3}$	$M3$	9	$O_{3,3}$	$M2$	3
$O_{1,4}$	$M2$	4	$O_{2,4}$	$M4$	6	$O_{3,4}$	$M3$	6

Table 2.3: Processing times for Gantt Chart Example

According to [132], optimal schedules tend to be semi-active schedules that are active but are not non-delay. This categorization of feasible solutions is depicted in Figure 2.1.

2.1.6 Representation Models

Representing the schedule that is feasible can be done in two main methods: *Gantt Chart* and *Disjunctive Graph*.

2.1.6.1 Gantt Chart

The Gantt chart [51] is the most common method to graphically represent a schedule. The chart consists of the unit time in the abscissa and machines on the ordinate axis. The chart shows the related jobs or operations to be performed on the different machines. Each block incorporates the starting time and completion time of that job or operation.

Figure 2.2 shows one of the solutions to job shop scheduling problem of the Table 2.3. The table shows the predetermined machines that the operations must be processed on, along with its required processing times.

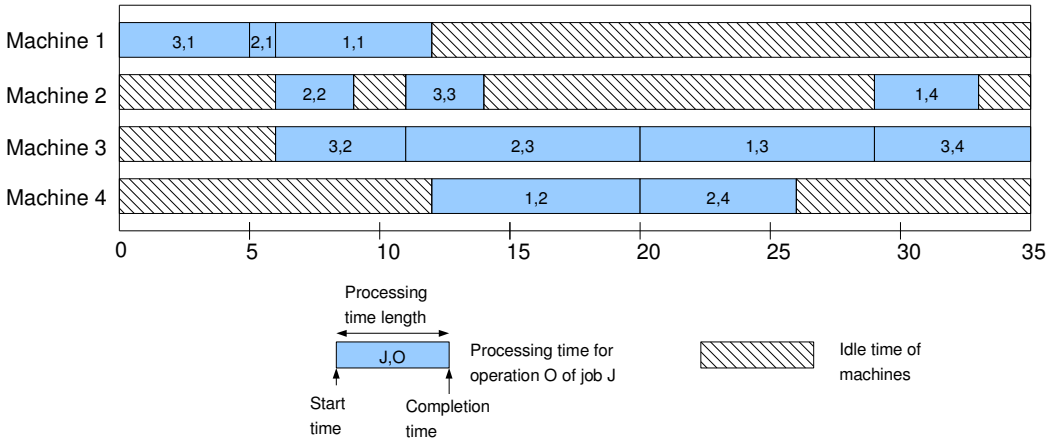


Figure 2.2: Example of Gantt Chart

2.1.6.2 Disjunctive Graph

The disjunctive graph is another method used to represent schedules [16]. This graph as shown in Figure 2.3 displays relationships between jobs and operations, as well as relationships between operations to be processed in a particular machine. The nodes represent the individual operations with the associated processing times beside it. The conjunctive relationships are shown between operations of the same job. However since each machines may consist of several operations that can be processed on it, the disjunctive clique is used. The clique contains the subset of operations to be processed on the same machine, such that each operation pair has a disjunctive relationship. The graph has two extra nodes representing the source and sink of the schedule that corresponds to the initial and final operation, but without any processing times attached. A deduction can be made from the graph for the actual schedule when the bi-directional disjunctive constraints are uni-directional and the directed disjunctive graph is non-cyclic. This deduction of a feasible schedule will determine the order the operations should be processed and also the sequencing of the operations on each machine.

2.2 Types of Scheduling Problems

The functional classification of scheduling problems as reflected in [85] comprises as follows:

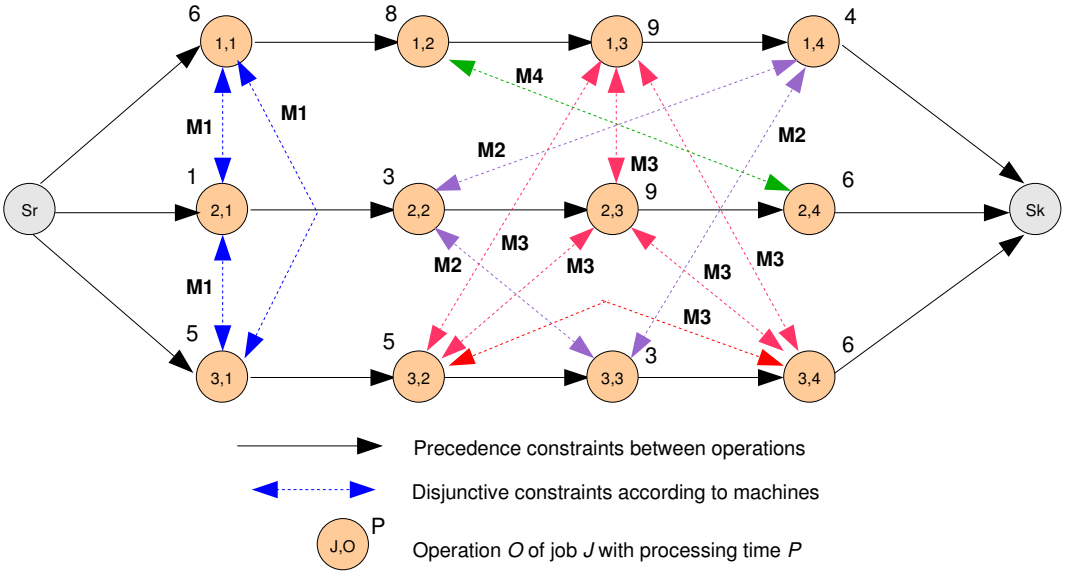


Figure 2.3: Example of Disjunctive Graph

1. Requirement Generation
2. Processing Complexity
3. Scheduling Criteria
4. Parameter Variability
5. Scheduling Environment

The requirement generation will determine if the manufacturing system is a closed shop or open shop. *Closed shop* is where the inventory parts are used from existing in-house stores. *Open shop* is where the plant will source parts from outside with no inventory kept in the plant. Hence there will be a smaller storage cost involved.

The processing complexity of the scheduling problem is known from the number of processing steps and machines required. The following are divided into:

1. single stage, single processor
2. single stage, multiple processor
3. multiple stage, flow shop
4. multiple stage, job shop

The environment that dictates the layout of the machines in the shop floor depends on the type of products being manufactured or processed.

A good, well designed and compact machine layout can minimize cost, reduce completion time, have high throughput and allow for more flexibility.

2.2.1 Single Machine

The single machine environment is the simplest of all machine environment. Here only one machine exist to execute all the tasks. Some of the more complicated machine environments may be uniquely reduced to this state. Although there is only one machine to be utilized, characteristics like set-up time and capacity of the machine makes this problem interesting to solve.

2.2.2 Parallel machine

The parallel machines set-up allows for more processing strength and flexibility compared to the single machine layout. It is common in manufacturing floors to have several identical machines positioned as parallel layouts. A job to be processed may be loaded on to any one of the parallel machines.

Some parallel machine layouts may consist of machines with variable speeds. This is also referred to as *uniform machines*. If the speed of a machine m is v_m, the processing time of task i is p_i then the period that the task is loaded on the machine is p_i/v_m.

However cases also exist where parallel machines are not identical, this are known as *unrelated parallel machines*. These unrelated parallel machines would have different speeds depending on the task being processed.

2.2.3 Flow shop

Rather than having machines positioned in parallel, these machines are in series. All jobs must be processed on each of the machines, starting from the first machine and finishing at the last machine. Usually a *First In First Out* (FIFO) policy applies in flow shops as the machines are set in series. Tasks cannot be loaded on the next machine if the machine is busy so it must be held on the buffer or on the current machine. The processing

order on a machine will normally determine the order for the rest of the machines in the series, also known as permutation flow shop [100].

Another variation of the flow shop is the flexible flow shop also known as *hybrid flow shop* [113], where a combination of parallel machines exists in series. A set of parallel machines will form a stage, with tasks having to complete all stages. [113] reviewed various researches and approaches done into 2-stage, 3-stage, and k-stage hybrid flow shop. Several other research works into flow shops have concentrated on stochastic flow shop that incorporated unknown release time of tasks, breakdown in machines and stochastic processing times, as seen in [58]. *No-wait flow shop* [8] or continuous flow shop [44] is another special case where no operations must be left idle between consecutive machines, even if this delays the first loading of the operations on the first machine. The case where the machines are always kept busy once the production has started is the *no-idle flow shop* [88]. Flow shop with up to 2 machines are polynomially solvable but with more than 2 machines, the flow shop problem is *NP*-hard according to [53], [57].

2.2.4 Job shop

The job shop environment is a much more complicated scheduling case. Each N jobs has numerous operations to be processed on the limited number of machines, M. Each job may require different and specific routes through the machines. The operations may also require to be processed once on the machines. If the operations is routed and to be processed on the same machine more than once, that would be classified as a *re-entrant job shop*.

One of the earliest works into job shop scheduling stemmed from [125]. The common constraints associated with the job shop scheduling are:

- Each job must be processed by a particular machine exactly once in a pre-known order.
- All jobs are ready to be processed at time zero.
- Processing of a job on a particular machine is called an operation, hence a job can only be completed if all operations have been processed.

- The processing time of the operations are a deterministic, constant and are known in advance.
- A precedence constraint exists between operations from the same job, which means that the preceding operation must be finished before a succeeding operation can be started.
- Once the operation has been started by the machine, it can not be interrupted or preempted.
- Each machine can process only one job in any period of time.
- Assuming no machine breakdown, and negligible transportation time, and setup time.

2.2.5 Open shop

The open shop differs from the job shop such that no precedence relationship is considered between tasks. Similar to the job shop layout, for N jobs and each job j, there are a preset n_j number of operations to be completed or processed on M machines. Hence the scheduling problem aims to determine the job orders on the machines and machine orders [22].

2.2.6 Flexible Manufacturing Systems (FMS)

The Flexible Manufacturing System (FMS) normally comprises of numerically controlled machine tools, serviced by a material handling system. These machines are flexible and can handle numerous part types with minimum change over time required. FMS allows the choice of one or more stations for each operation and one or more processes to manufacture each part type. FMS has been sought after as one of the best compromises between flexibility and economical cost for high volume production [28]. The flexibilities associated with FMSs is in terms of operating sequence, transport and multiple resources working simultaneously. Scheduling in FMS differs from other conventional scheduling machine environment in its feasibility characteristics [83].The availability of alternative resources in the system to handle routing flexibility makes a big difference. This eliminates the possibility of major bottlenecks in the machines when routing is not feasible. FMS can be divided into two sub classes: *Real time* (or reactive) and *predictive* (or deterministic). Real time FMSs are usually subjected to unstable demand or frequent disturbance

to production. Whereas deterministic FMSs are more stable and allow for optimal states for optimizing criteria e.g. makespan, production flow, WIP, waiting times and buffers size.

2.3 The Cyclic Scheduling Problems

The cyclic scheduling, as a special case of the scheduling problem, can be defined as the infinite occurrence of generic tasks required to produce numerous assemblies. An occurrence is considered to be a single execution of the task. Even though the task occurrences are infinite, there is a particular cyclic pattern associated with these occurrences.

Compared to the non-cyclic scheduling problems, the *cyclic scheduling* problem differs in a few instances. In cyclic scheduling, the set of tasks will be executed repeatedly over a probable extended infinite time horizon. This may include either a number of cycles or an infinite number of cycles or loops. This can be achieved through generating a special framework (or schedule) comprising of a pattern of operating sequence that is to be executed repeatedly. This reduces the complexity by seeking out the optimization of a single schedule.In real-life, many systems, such as production line, the tasks are executed repeatedly.

Where else, the non-cyclic (*acyclic*) scheduling problems involve the scheduling of all finite tasks to accomplish the required demand. This is normally done without any definite pattern while trying to attain the desired objective e.g. makespan, tardiness, or earliness.

For the cyclic scheduling problem, a set of generic tasks, T are processed cyclically where there are N tasks in total. Let N be the number of generic tasks;

$$T = 1, 2, \ldots, N \tag{2.3}$$

Associated with each generic task, i is its processing time, p_i. The tasks are sequenced in a particular order in the form of a schedule. The operating sequence will generate a continuous flow of completed tasks in predefined intervals.

This will result in a single schedule that completes a task (or a set of tasks if the tasks are required to complete a certain job) or a job every τ unit of time. Here τ is defined as the *cycle time*. Hence, the schedule will repeat

itself every τ ($\tau > 0$) unit of time. The cycle time is a good measure of the throughput of the system [136].

In this research work, we denote start time of task i as S_i. Moreover if operation i is from a particular job, j, the start time will be $S_{i,j}$. In the case where, a consideration of which occurrence of the task, i is required, the superset parameter, k is attached to the start time variable as follows, $S^k_{i,j}$. Iteration $< i, k >$ denotes the k-th occurrence of task i.

The start time of occurrence k for a task i of a job j, with a fixed cycle time τ, is given by [64]:

$$S^k_{i,j} = S^0_{i,j} + \tau k \qquad \forall k \in Z \tag{2.4}$$

Also to guarantee that the *(k+1)*-th occurrence of an operation i can only proceed if the previous k-th occurrence of the same operation has been completed, the following statement holds true:

$$S^{k+1}_{i,j} \geq S^k_{i,j} + p_{i;j} \tag{2.5}$$

For the two operations i and i' that should be processed on the same machine M, we ensure guarantee that the two operations do not overlap, so the following statement holds true:

$$S^k_{i,j} \neq S^k_{i',j} \tag{2.6}$$

The average cycle time, ω can be expressed as follows [65]:

$$\omega = lim_{k\to\infty} \frac{max_{i\in T}(S^k_{i,j} + p_{i,j})}{k} \tag{2.7}$$

The average cycle time can be used to calculate the throughput of the system which is equal to $\frac{1}{\omega}$.

2.4 Parameter in Cyclic Scheduling Problems

The two most important parameters in the cyclic schedule are the *cycle time*, τ and *latency*.

2.4.1 Cycle Time

The cycle time would be the time taken to execute all the operations before the next schedule pattern is repeated. When processing tasks in a cyclic schedule, the difference between occurrences of the task is also measured as cycle time. Minimizing the cycle time is one of the major criteria in cyclic scheduling as it will increase the number of jobs completed and maximizing throughput of the machines.

2.4.2 Latency

Latency is the number of cycles required to complete a single job. This definition also normally denotes the number of jobs concurrently under production.

As an example, Figure 2.4 shows a cyclic schedule with cycle time of 15 units that is composed of 8 tasks. Figure 2.5 shows an improvement of the cycle time of only 9 units for the same number of tasks but with latency value of 4.

2.5 Characteristics of Cyclic Scheduling Problems

2.5.1 K-Periodicity

The cyclic scheduling problem can be divided into 1-periodic schedule (or periodic schedule) and K-periodic schedule for $K > 1$. 1-periodic schedule has only 1 occurrence of each operation per cycle period and K-periodic schedule has K occurrences of an operation per cycle. An example of the K-periodic schedule is shown in Figure 2.6.

2.5.2 Precedence Constraints

Precedence constraints determine the order of processing for the operations within each job or task. Any two tasks or operations with precedence constraints must have one task completed before another task can be started. Denoting start time and end time of task T_a by Start(T_a) and

Jobs with processing times		
Job	Machine	P.T.
O_1	$M1$	1
O_2	$M3$	2
O_3	$M1$	2
O_4	$M2$	3
O_5	$M1$	3
O_6	$M3$	1
O_7	$M2$	1
O_8	$M3$	2

Table 2.4: Cyclic Gantt Chart

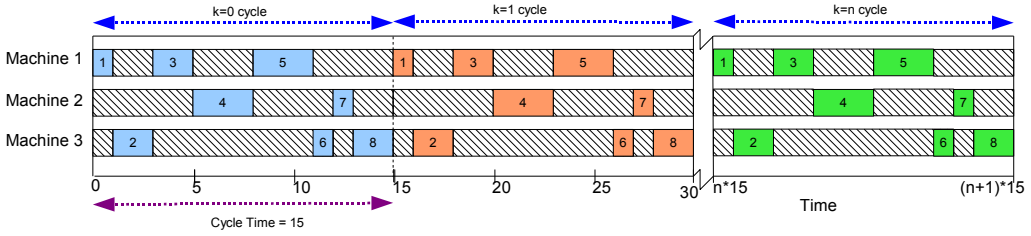


Figure 2.4: Cyclic Schedule of Latency equals 1.

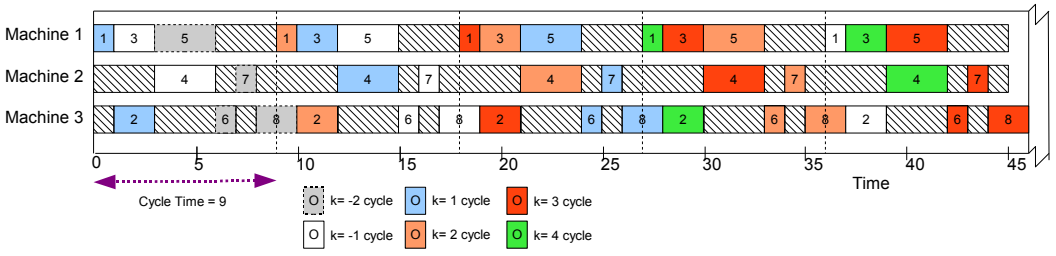


Figure 2.5: Cyclic Schedule of Latency equals 4.

Jobs with processing times		
Job	Machine	P.T.
O_1	$M3$	2
O_2	$M1$	1
O_3	$M2$	3
O_4	$M1$	2

Table 2.5: 2-periodic Cyclic Gantt Chart.

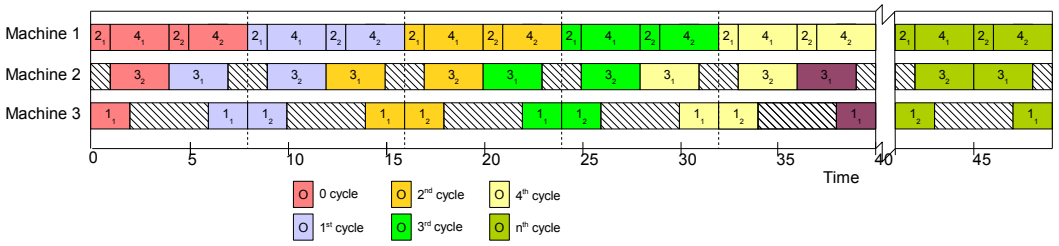


Figure 2.6: 2-periodic cyclic schedule

End(T_a) respectively, and start time and end time of task T_b by Start(T_b) and End(T_b) respectively. This is mathematically modelled as:

$$End(T_a) \leq Start(T_b) \quad or \quad End(T_b) \leq Start(T_a) \tag{2.8}$$

In the first case above, the task T_b is a successor of T_a, but in the second case, task T_b is the predecessor of task T_a. Previously *Lenstra et. al. (1978)* [108] studied the complexity of scheduling with precedence constraints while *Blazewicz et. al. (2002)* [17] reviewed the properties of task-on-arcs graph and task-on-node graphs representation of precedence constraints for the scheduling problems.

Here within the cyclic schedule, the restriction of when to start an operation only after the completion of another operation is vital for feasibility. We can distinguish two types of precedence constraints.

Uniform Precedence constraints: In basic cyclic scheduling problems (without any resource constraints) with uniform precedence constraints is when the execution indices between two cyclic tasks/operations are constants. One of the modelling methods of these uniform constraints is the *uniform graph* [65]. The uniform graph represents the precedence constraint by nodes (representing tasks), directed arrows (relationship of precedence), with value on relationship of length, L and height, H. Here L is the processing time of the task (or start/delay) and H as occurrence denominator (or height/distance). The generalized precedence constraint between task i and i' is as follow:

$$S_i^k + L_{i,i'} \leq S_{i'}^{(k+H_{i,i'})} \tag{2.9}$$

with L is a rational number and H an arbitrary integer.

The precedence constraint between the subsequent occurrences of a par-

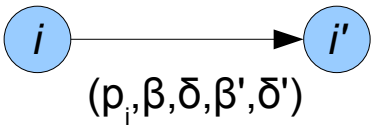


Figure 2.7: Linear precedence constraint

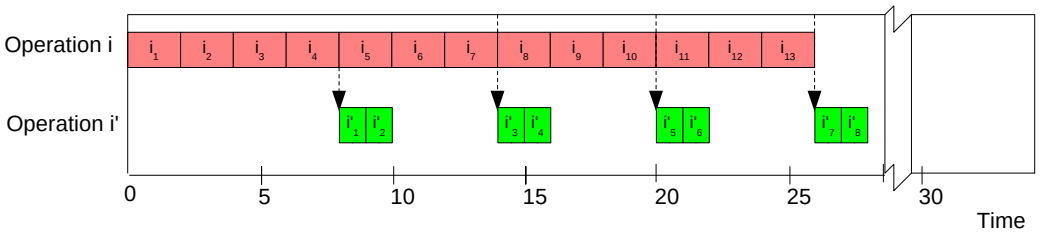


Figure 2.8: Gantt Chart for linear precedence constraints (2,3,1,2,0)

ticular task i is:

$$S_i^k + p_i \leq S_i^{(k+1)} \tag{2.10}$$

with p_i as processing time for task i.

Linear Precedence constraints: The linear precedence constraint that governs the generic tasks i and i' can be defined as follows [124]:

$$S_i^{(\beta_{i,i'}n+\delta_{i,i'})} + p_i \leq S_{i'}^{(\beta'_{i,i'}n+\delta'_{i,i'})} \quad \forall n > 0 \tag{2.11}$$

The $\beta_{i,i'}$ and $\beta'_{i,i'}$ are two positive natural numbers while $\delta_{i,i'}$ and $\delta'_{i,i'}$ are just two natural numbers. The linear precedence constraint between operations i and i' is denoted by $(p_i, \beta, \delta, \beta', \delta')$. This means that $\beta n + \delta$ occurrence of operation i must be completed before occurrence $\beta' n + \delta'$ of operation i'.

This linear relationship can be represented by a directed graph $G = (T, E)$ with nodes T and arcs E. The nodes represent the operations while the arcs represents the linear constraints. The example in Figure 2.8 shows the linear precedence constraints effect on the occurrences of both operations i and i'. Here the constraints are denoted by $(2, 3, 1, 2, 0)$ or $S_i^{(3n+1)} + 2 \leq S_{i'}^{(2n)} \quad \forall n > 0$.

According to [124], some linear precedence constraints can be reduced

into uniform constraints. From:

$$Minimize \quad \tau \quad s.t. \tag{2.12}$$

$$S_i^{(\beta_{i,i'}n+\delta_{i,i'})} + p_i \leq S_{i'}^{(\beta'_{i,i'}n+\delta'_{i,i'})} \quad \forall n > 0 \tag{2.13}$$

into

$$Minimize \quad \tau \quad s.t. \tag{2.14}$$

$$S_{i'} - S_i \geq L_{i,i'} - \tau H_{i,i'} \quad \forall (i,j) \in E \tag{2.15}$$

provided that the graph G is strongly connected. This means that for all cycles c of graph G, the combined weight of the arcs $\prod \frac{\beta'}{\beta} = 1$. Assuming that the weight for each arc (i, i') is $\frac{\beta'_{i,i'}}{\beta_{i,i'}}$.

From [67], it is proven that the periodic schedule holds true for linear precedence constraints for all arcs $(i, i') \in E$ if:

$$S_i^0 + (\beta_{i,i'}k + \delta_{i,i'})\tau_i + L_{i,i'} \leq S_{i'}^0 + (\beta'_{i,i'}k + \delta'_{i,i'})\tau'_i \tag{2.16}$$

Some key researchers that had investigated the linear precedence constraints in cyclic scheduling problem are *Munier (1996)* [124], *Hanen and Munier Kordon (2008)* [67] and *Cavory et. al. (2005)* [27].

2.5.3 Disjunctive Constraints

Another important constraints widely used in this scheduling domain is the ***disjunctive constraints***. These constraints are applicable in both non-cyclic and cyclic scheduling problems These constraints deal mainly with relations to the resource availabilities [13]. The purpose of these constraints is to ensure that the two or more tasks assigned to the same resource, are not carried out or processed simultaneously. This will eliminate the chances of overlapping in time when processing the tasks.

This can be illustrated by using 2 simple tasks: T_a and T_b that can only be processed on the same machine M. The disjunctive constraint on machine M at any instant of time is:

$$End(T_a) \leq Start(T_b) \quad or \quad End(T_b) \leq Start(T_a) \tag{2.17}$$

As such the example above denotes that variable constraints of start

times for task T_a (or start time task T_b) cannot start until end time of task T_b (or end time of task T_a) has been completed. Here the constrained propagation will try to reduce the values of the variables to adhere to the above equations. *Baptiste et. al. (1996)* [13] extended this constraint in a way as to include the *state resources* and the flexibility of time transition between operations being processed for the ILOG ^{TM}SCHEDULE software. The state resources where resources are operating only at a particular state by the tasks, are defined as:

$$\begin{aligned}[End(T_A) \leq Start(T_B)]or[End(T_B) \leq Start(T_A)] \\ or[duration(T_A) = 0]or[duration(T_B) = 0] \\ or[State(T_A) = State(T_B)]\end{aligned}$$

where $State(T)$ represent the state of task T and $duration(T)$ as duration for which the task requires the resource.

2.5.4 Cyclic Constraints

This constraint applies when defining the schedule, i.e. framework to be repeated when a deterministic and predefined cycle time is known and fixed. Consider the start time, $S_{i,j}$ for a particular operation $O_{i,j}$, the cyclic constraint is denoted as:

$$0 \leq S_{i,j} < \tau \tag{2.18}$$

where τ is the fixed cycle time.

2.5.5 Performance Measures

The most common objective functions or performance measures used in cyclic scheduling problems include:

1. Maximizing throughput of the cyclic production system that will also maximize the frequency of completing jobs. This is equivalent to minimizing the cycle time. In effect: Maximizing throughput will maximize work in progress (WIP) but will minimize cycle time.

2. Minimizing work-in-progress will thus save storage space and reduce parts waiting times.
3. Minimizing flowtime i.e. time needed to produce one occurrence of a product or complete one job comprising of a finite number of operations.
4. Minimizing cycle time i.e. minimizing time difference between two succeeding occurrences of operations.
5. Minimizing K-periodic cycles i.e. minimizing time difference between *1*-th occurrence and *1+K*-th occurrence of an operation where *K*-occurrence of an operation being processed in one period.

2.6 Classes of Cyclic Scheduling Problems

Static cyclic scheduling problems have fixed numbers of jobs, machines and processing times. However in the case of dynamic cyclic scheduling problem, stochastic factors are incorporated into the cyclic scheduling problems.

Cyclic scheduling problem can be categorized into two main categories. Firstly precedence relationship between tasks without resource constraints and the other with resource constraints.

2.6.1 Basic Cyclic Scheduling (BCS)

Basic Cyclic Scheduling (BCS) problem is one of the simplest form of cyclic scheduling problem without resource constraints. In this problem, the precedence constraints that dictate the order of processing the tasks are uniform where when executing the generic tasks, the constraints has an execution index that is a constant. Among the main objectives of BCS problem is to maximize the throughput. Since no resource constraints are considered, the number of machines are not limited. BCS with an exact solution is proven to be of complexity $O(n^3 logn)$ where n is the number of tasks in the system [65]. *Chretienne (1991)* [32] also looked into the use of expanded graph and latest schedule to solve the basic cyclic scheduling problem. The other cyclic scheduling problem without resource constraints is the ***Basic Cyclic Scheduling problem with Linear***

Precedence Constraints (BCSL). Some of the special cases of BCSL problem may be transformed into a BCS problem. According to *Hanen and Munier (1995)* [65], these can be achieved by expanding the graph modeling and the number of tasks of the cyclic problem.

2.6.2 Cyclic Scheduling Problem with Resource Constraints and Linear Precedence Constraints

An extension to the basic cyclic scheduling problem, *Brucker and Kampmeyer (2005)* [20] solved the cyclic scheduling problem with limited resources and with linear precedence constraints. Here the constraints of a machine only being able to process one operation per instant of time is added into the problem.

2.6.3 Robotic Flow Shop

Also another type of cyclic scheduling problem is the robotic cell problem or ***robotic flow shop*** (RFS). This particular type of problem involves a robotic arm that functions as a transportation mechanism for moving the work in progress parts between machines. The aim of the robotic arm is to complete $N - unit$ activity sequences where a sequence is a set of the robot moves, to load and unload each machine exactly N times [92], [38] to maximize the throughput of the tasks. The complexity of this cyclic scheduling problem when identical parts are involved has been proven to be $O(m^3)$ where m is the number of machines in the system and is *NP-Complete*.

2.6.4 Hoist Scheduling Problem

The ***Hoist Scheduling Problem (HSP)*** is concerned with the use of hoists to transport and submerge printed circuit boards into successive tanks to be electro-plated. Restrictions on the hoist includes using only a particular track and collisions must be avoided. Infinite numbers of boards are involved and usually there exists a lower and upper time limit in which the boards can be submerged. The main objective is to minimize the cycle time. It has been proven that it is NP-hard for a single hoist and unique job of identical parts.

2.6.5 Cyclic Scheduling Flexible Manufacturing Systems

This is normally used for medium to high volume production. Cyclic scheduling is possible as the high volume normally comprises of a combination of small number of identical quantities. So using the cyclic scheduling allows for scheduling of a small number of operations in one cycle. Followed by repeating the first cycle continuously to fill the required demand. The cyclic scheduling of the FMS is possible if the following holds:

- production of different types of parts are programmed by a cycle with respect to specified ratios resources are shared by many operations,
- each part is routed on repeated sequence of production,
- manufacture of parts that consist of assembly or disassembly of several components,
- a resource is associated with only one operation at any single period of time and no preemption is allowed.

Further descriptions of this class of cyclic scheduling problem can be found in [129], [96], [28], [149], [150], [104], [77], .

2.6.6 Cyclic Scheduling Flow Shop

Cyclic Scheduling Flow Shop is based on the properties that all the tasks having the same machine loading sequence and same processing sequence. *Minimal Part Set (MPS)* that represents the smallest set having the same proportions of the different item type as required to be produced, is commonly associated with flow shop and from the production requirements of r_l units of item type l, where $l = 1, 2, \cdots, L$. *MPS* can be expressed as [73]:

$$r^* = (\frac{r_1}{q}, \frac{r_2}{q}, \cdots \frac{r_L}{q}) \tag{2.19}$$

where q is the greatest common divider for parameters $r_1, r_2, \cdots, r_L$. *Karabati and Kouvelis (1998)* [90] studied the cyclic scheduling on the flowline that assumes finite or infinite buffers between machines and works to schedule jobs within the *MPS* before extending it to the α-*MPS*

to represent a cyclic schedule, where α is the integral constant of proportionality and a divisor of q.

2.6.7 Cyclic Open Shop

Cyclic Open Shop [99] represents the cyclic version of open shop. However the cyclic schedule with no idle and waiting time allowed for tasks and machines are known as compact cyclic open shop. This is more obvious for use in metallurgy industries where the same product is repeatedly produced, but the part being processed is sensitive to delay. Both cyclic open shop and compact cyclic open shop share similarities with normal open shop with no precedence constraints considered. The cyclic open shop is proven to be $\mathcal{NP}$-Hard for 3 or more machines while it is also $\mathcal{NP}$-Hard for compact cyclic open shop of 2 or more machines, [99] for the case of minimizing makespan.

2.6.8 Cyclic Job Shop

The ***cyclic job shop scheduling problem (CJSSP)*** is the cyclic version of the job shop scheduling problem. Here the operations must be cyclically processed in order to complete the infinite number of each jobs required. Also all the jobs may also be processed for a infinite number of times. All the properties of the cyclic job shop remain the same as the non-cyclic but with the addition of maximizing the throughput through a minimum cycle time. *Boussemart et. al. (2002)* [18] introduced the CJSSP with linear constraints, while *Seo et. al. (2002)* [141] introduced the term cyclic job shop with overtaking or *overtaking cyclic job shop* (OCJS) where the latency value for the cyclic schedule is more than one. However the basic cyclic job shop (BCJS) have no overtaking properties. Another form of cycle shop is the one introduced by *Middendorf and Timkovsky (2002)* [122]. This *cycle shop* is a special cyclic version of a job shop, however having flow shop properties. All jobs in a cycle shop have the same sequence of operations on the machines, but in contrast to a flow shop, some operations may be repeated on particular machines a number of times, as specified depending on the machine.

2.7 Periodic/Cyclic Schedule

The cyclic schedule is periodic if the following holds true for any start time of operation i of job j:

$$S_{i;j}^{k} = S_{i;j}^{0} + \tau(k) \qquad \forall k \in Z \tag{2.20}$$

where occurrence $\forall k \geq 1$, τ as average cycle time and $S \geq 0$.

The cyclic scheduling problem that we shall be focused on, is the periodic schedule. The periodic schedule here means that a particular schedule is repeated using a certain period value. This period that equals to the cycle time can be used to give the next iteration in the next cycle of a particular operation within the cyclic schedule.

In solving the cyclic scheduling problems and generating the cyclic schedules, the approaches normally include finding a schedule (framework) that will adhere to precedence and disjunctive constraints and repeating that schedule found at the earliest possible time. These efforts however may consider fixing a certain cycle time and generate the feasible schedules from that if a minimum latency is required. But in the case of minimizing the cycle time, fixing number of latency and generate feasible schedules from that may also work.

Alternatively, the other approach involving optimization tools will follow the sequence of finding a sequence for the operations, optimizing the sequence through some optimization technique, that will not violate any constraints set and using the objective of optimizing the set objective criteria.

For deterministic cyclic scheduling problems, some information such as number of jobs and their operations, the operations processing times, the number of machines capable of processing the operations, the set-up times and the restrictions involved in processing each operations, are vital for generating possible schedules. However whether the given schedule is feasible or not, depends on the restrictions involved and the accurate modelling of these restrictions. A feasible cyclic schedule must fulfill all constraints requirements, but may not be the best cyclic schedule for the problem. The best and optimum cyclic schedule will definitely maximize or minimize the criterion set, but in many cases due to the complexity of the problem, only come close in reaching the lower bound or upper

bound of the criterion available. The level of successfulness in attaining this depends greatly on the approach used to solve the cyclic problem.

A cyclic schedule T is considered optimal or efficient if no other cyclic schedule can match the cycle time and have a lower value of objective value than the cyclic schedule T.

2.7.1 Transient State Research

The study of transient state in cyclic scheduling for especially FMS can be found in [97]. The transient state occurs when the jobs are loaded onto the machines before reaching the steady state i.e. known as *pre-production* and also can be found when completing the production i.e. *post-production*. The goal then is to minimize this transient state and maximize the steady state of the productions.

The upper bound of pre-production transient state can be calculated (based on predetermined cycle time):

$$max_j \left\{ \tau * n_p(j, BPL) - BDOS(j, BPL) \right\} \tag{2.21}$$

where τ is the cycle time, $n_p(j, BPL)$ is the number of pallets used by the job j at beginning date of periodic loading (BPL). $BDOS(j, BPL)$ is the start date of the job j for the beginning date of periodic loading (BPL).

The lower bound of the pre-production transient state is given by:

$$max \left\{ OT\left(W_j\right)(BPL) \right\} \tag{2.22}$$

where $OT\left(W_j\right)(BPL)$ is the sum of all the operating times to be processed for job j in the pre-production for pallet W_j with fixed BPL.

The post-production duration bounds can be calculated from the upper bound:

$$max_j \left\{ (N-1) * \tau + EDOS(j, BPL) \right\} \tag{2.23}$$

with $EDOS(j, BPL)$ as end date of operating sequence of job j at BPL of the steady state and N as the total number of job j required to be produced.

The lower bound of post-production is as follows:

$$max_j \left\{ (N - n_p(j, BPL)) * \tau + BDOS(j, BPL) \right\} \tag{2.24}$$

[97] summarized that the transient state does not depend explicitly on the number of operations of jobs to be cyclically scheduled. However the importance of transient state diminishes compared to the importance of minimizing the makespan when the number of operations increases.

2.8 Summary

In this chapter we have described the classification of scheduling problems and the framework to compare scheduling techniques based on production system models, manufacturing scenarios, plant scenarios and operational scenarios. We have also summarized the various characteristics of the scheduling problem involving the tasks, machines, constraints and objective function of the problems. We also describes some of the types of scheduling problems in terms of differences and special features in relation to cyclic scheduling. In addition to fulfilling the objective function of cyclic scheduling problem, objectively cyclic scheduling also aims to:

1. generate the maximum number of jobs within a certain period of time,
2. leave least slack between operation of machines,
3. leave least time between cycle of schedules thus having least amount of jobs hanging(or in progress).

According to *Toguyéni et. al. (2005)* [148], cyclic scheduling is by definition deterministic and a viable option as breakdown of machines are supposedly rare, with sufficiently large interval during which estimated control can be fully exploited without interruption. Using both of these assumptions, these will allow for reaching the expected optimal performance and maintaining them over sufficiently long time. Deterministic characteristics associated with cyclic scheduling is vital as this allows for optimization of the flow of production while precise knowledge of the system state, allows for constant control of production.

Cyclic scheduling is advantageous in the sense that it is easier to implement as it is a matter of repeating a certain schedule infinitely or for a fixed number of iteration, compared to non-cyclic/non-repeating approaches to reaching the goal of completing a fixed amount of jobs in

minimum time. Referring to the complexity issue, cyclic schedule has reduced complexity, as it is independent of the number of operations. In terms of the production rate, the completed jobs and products period can be continuously known. Cyclic scheduling also encourages predictable shop behaviour as recurrent schedule is known to lead to more simplified flow control of products. This will then allow for smoother control of level associated with finished products and inventories of parts [90]. *Hall et. al. (1997)*[61] quoted that better utilization or simultaneous utilization of machines in processing the operations can be achieved with cyclic scheduling. [63] also noted that the cyclic scheduling approach allows for planning at a higher level being the ideal approach to planning of production. This can be done by aggregating processes according to similar routing and processing requirements.

However, the drawback of cyclic schedules is the existence of disruptions in the system. Disruptions in the form of machine failure or part delivery issues. This must be compensated for by dynamic real-time control to handle these uncertainties. [160] considered the stochastic programming of unpredictable machine failures and considered buffered stock and safety times to deal with these aspects.

This chapter has describe the cyclic scheduling problem and various parameters associated with this problem. This has also describe the different categories of the cyclic scheduling problem and various pieces of research done to solve this problem.

CHAPTER

THREE

Modelling Cyclic Scheduling Problems

3.1 Introduction

The aspect of problem modelling plays a vital part in the approach used in solving the scheduling problems. In order to capture all the equivalent elements of the real life conditions and how the elements interact and affect the behaviour of the problem, good modelling in mathematical terms, is necessary. The more complex a scheduling problem, the more sophisticated the mathematical modelling involved will be. The more parameters, variables and relationships between parameters in existence, the more helpful it is, in modelling the scheduling problems accurately and in real term. This also presents the problem of how many parameters are needed, and whether the parameters are necessary and accurate enough. Over the decades, researchers have tried to efficiently model the scheduling problem, in order to not only understand how the systems work, and interact but also in a way to validate their modelling characteristics.

After describing and discussing the individual characteristics of the scheduling elements in the previous chapter, we describe the relationship between the elements and how the system changes over time. The relationships between elements will help us understand how the scheduling system behave, and present a way of how to measure its performance and analyze its behaviour. This is important to assist in preparing the scheduling problem in a form that is suitable for applying a solving technique. An efficient modelling of the scheduling problem can help to reduce the effort in solving the problem and allow the solving approach to

fully utilize its potential in reaching a correct or optimal solution(s).

Two main representations of the scheduling problems exist:

1. Constraint Satisfaction Problems (CSPs) model,
2. Constraint Optimization Problems (COPs) model.

CSPs considers time limit as a constraint as regards to solving the scheduling problem while COPs consider the scheduling problem to be optimized using time limit as an objective function [18].

3.2 Modelling description

3.2.1 Problem formulation

A scheduling problem is formulated as one of the following objective functions.

The ***completion time***, C_i associated with a task i, is the end time of the task processed by the last machine. The completion time of the task, defined by the following equation, is generally used to define the other objective functions and affect the sequencing of other tasks on machines.

$$C_i \geq 0 \tag{3.1}$$

Keeping the completion time of each tasks to a minimum, this will not only make the completed job available sooner but it will also allow for the resource (i.e. machines) to be available sooner to process other jobs.

The ***makespan***, C_{max}: is the time when the last job is completed and exit the system. Mathematically, the makespan of a system is the maximum completion time of the set of jobs required to be processed by the machines, defined by

$$C_{max} = max(C_1, C_2, \cdots, C_n) \tag{3.2}$$

Another objective that define the scheduling problem is when the jobs have predefined due date, d_j with completion time C_j, is ***lateness*** from:

$$L_j = C_j - d_j \tag{3.3}$$

This measurement of lateness is positive if the job is late but negative if it is completed early.

Tardiness, T_j: is defined as the difference between the actual completion time and the desired completion time for a particular task (i.e. taking the positive value of lateness):

$$T_j = max(C_j - d_j, 0) = max(L_j, 0) \tag{3.4}$$

The direct opposite of lateness is ***earliness***, E_j when the job is completed early. This is defined as follows:

$$E_j = max(d_j - C_j, 0) \tag{3.5}$$

The total completion time of all jobs is the ***flow time***, F_j defined by the difference between completion time and the time at which the task was released to the production shop.

$$F_j = \sum C_j \tag{3.6}$$

The ***waiting time***, W_j is the sum of all the non-active time which the job is waiting between being processed, defined by:

$$W_j = C_j - r_j - \sum_{j=1}^{N} p_j \tag{3.7}$$

where r_j and p_j are release time and processing time respectively.

All the above objective functions are generally subjected to precedence and disjunctive constraints.

3.3 Scheduling problems - Complexity

To understand how difficult it is to solve the scheduling problem, consider a scheduling problem with N jobs to be processed on M machines. The M machines must process the N jobs. There are $N!$ permutations of sequences possible on each machine and $(N!)^M$ possible sequences for all the machines. However the $(N!)^M$ sequences may not contain all feasible solutions as some may violate the precedence or disjunctive constraints.

The hard problem is compounded by the fact that if N or M increases, the combinatorily explodes from a non-polynomial complexity!

As such the Complexity Theory is used to classify the difficulty of the scheduling problem. Basically, whether it is possible to solve the scheduling problem depends on the efficiency of the algorithms used. The efficiency of the algorithm is measured by its complexity or the maximum number of computational steps required in seeking out the solutions. Generally in the scheduling problem, the inputs to the problem are taken as the number of jobs (or number of operations). So using these inputs, the algorithm can be executed in *polynomial* time, $O(n^k)$ where n is the input size with k a constant value. The scheduling problem solved with this algorithm is then considered as polynomial solvable. Scheduling problems may also be *pseudopolynomial solvable* when the input is decoded as unary values [52].

The scheduling problems is classified as hard or easy, based on complexity $\mathcal{P}$, $\mathcal{NP}$-hard, $\mathcal{NP}$-complete [52], and can also be classified as either optimization problem or decision problems. Class $\mathcal{P}$ problems are decision problems solvable in polynomial time while $\mathcal{NP}$ (or non-deterministic polynomial time) problems can be solved in polynomial time where the correct answer can be determined from a proper clue or solvable by non-deterministic turing machine. $\mathcal{NP}$-Hard problems are when the $\mathcal{NP}$ optimization or decision problems can be reduce to $\mathcal{P}$ problems. Strongly $\mathcal{NP}$-Hard problems is when no polynomial time algorithm exists. The other classification is the $\mathcal{NP}$-Complete which also belong to set $\mathcal{NP}$ and is at least as difficult as any other problem in $\mathcal{NP}$.

Scheduling problems are part of the Constrained Optimization Problems (COPs) and generally these problems are $\mathcal{NP}$-Hard [22]. Surveys by *Brucker et. al. (2007)* [21] revealed that in the case of job-shop scheduling problems, where it has been proven to be $\mathcal{NP}$-Hard in case of $N \leq 2$, is solvable in polynomial time but there is no guarantee that an optimal solution can be found using the most efficient algorithm for $N \geq 3$.

Using the complexity theory of problem reduction, researchers have been able to classify the complexity of solving the common scheduling problems. *Brucker* [23] managed to track the latest findings to these complexities using a customized program called CLASS. Some major examples associated with complexities, can be shown in the Table 3.1. More detailed description of complexities associated with shop scheduling prob-

Problems	Specifications	Complexities
Single machine	$1\|prec; p_i = p; r_i\|\sum C_i$	polynomial solvable [101]
	$1\|prec\|\sum C_i$	NP-Hard [108]
Parallel machine	$P2\|\|C_max$	NP-hard [110]
Flow shop	$F\|prec; p_{i;j} = 1\|C - max$	NP Hard [111]
Job shop	$J2\|\|C_{max}$	NP-hard [109]
	$J3\|n = 3\|C_{max}$	NP-hard [144]

Table 3.1: Some Example of Complexity of Scheduling Problems

lems can be found in *Brucker et. al. (2007)* [21].

3.3.1 Cyclic Scheduling Complexity

For cyclic scheduling from the job shop to FMS, the complexity issue plays a large part in the ability to solve them especially with the additional cyclic attributes [142]. *Chretienne (1991)* [32] proved that basic cyclic scheduling problem can be solved in polynomial time. The complexity for basic cyclic scheduling problem can be solved with periodic schedule in $O(n^3 logn)$ for n tasks [65].

In the job shop cyclic scheduling, it has been proven to be solvable in polynomial time for 1 machine with a maximum number of operations less or equal to 2, [82]. However, according to *Hall et. al. (2002)* [62] the 2-machine job shop is also polynomially solvable in time. This is only true for the maximum number of operations per job equal to 2. Further research have also proven that when the number of operation is a maximum of 3, this is binary NP-hard. Where as job shop with cyclic scheduling with 3 machines and maximum operations per job of 3 are unary NP-Hard [110].

The combination of complex resource flexibility shown by FMS and cyclic attributes push these cyclic problems to remain NP-Hard according to *Serafini and Ukovich (1989)* [142], [78].

So far the complexity issues on flow shop and open shop cyclic scheduling are not well known. *Lee and Posner (1997)* [106] proved that there exists a stable schedule that minimises the cycle time in flow shop, that is independent of the number of job sets and can be computed in $O(\sum_{j \in N} n_j)$ time.

As a summary, the complexity of the cyclic scheduling problem for two or more machines are mostly proven to be NP-Hard, hence meta-

Problem	n	m	Optimal makespan	First researchers to obtain optimal solutions
$BCL\|n\|\tau$	$O(n^3 logn)$	6	55	*Hanen,Munier(1995)* [66]
$J\|n_j \leq 2\|\tau$	Polynomial	6	55	*Balas (1969)* [12]
$J2\|n_j \leq 3\|\tau$	Binary $\mathcal{NP}$-Hard	10	930	Lageweg (1984)
$J2\|n_j \leq k\|\tau$	NP-complete	$k > 3$	1165	*McMahon and Florian (1975)*[119]
$JM\|n_j \leq k\|\tau$	Strongly NP-complete	$k > 3$	$M \geq 2$	*Hall et. al.(2002)* [62]

Table 3.2: Complexities for Cyclic Job Shop Problems

heuristic approaches are the best computationally efficient method available. Table 3.2 shows a summary of the complexity associated with each job shop cyclic scheduling problem.

3.4 Solving Scheduling Problems

The past research into solving the scheduling problem are generally split into two main categories - *approximation* or *optimization* technique and further into *constructive* or *iterative*. As described in Section 3.3, due to the complexity of the scheduling problems, the approximation approaches have the ability to deliver good results (but not global optimum) in an acceptable time frame. Ignoring the computational time in some way, the optimization approaches are more relentless in seeking the global optimum solution, although this tend to require massive computational effort. Digging deeper into these two types of techniques, the constructive approach builds the schedule from scratch using the information given whereas the iterative techniques will continuously reorder the initial feasible schedule to create better schedules.

The effectiveness of the two mentioned approaches depends greatly on the complexity of the particular scheduling problem. This effectiveness may be in terms of accuracy of solutions or speed in obtaining the optimal solution. Generally, with increasing number of operations and machines, the computational time required to solve it will increase tremendously except for special cases of scheduling problem that can be solved in polynomial time.

As well, the approaches to solve the scheduling problem may depend on the readiness knowledge of the pertaining scheduling information. Where categorically, *off-line scheduling* or static scheduling which generate a solution according to the known information provided on the

jobs and available resources over a planned period. In the case of *online scheduling* or dynamic scheduling, the approach is intrinsically able to alter an existing schedule to cater for new changes in demand or even completely regenerate a new schedule [132], [153].

3.4.1 Survey on scheduling techniques

There have been quite a few surveys done into the techniques used to solve the scheduling problems. Among them are *Yahyaoui (2006)* [156] who surveyed research papers that covered approaches in solving the job shop scheduling problem. These papers covered are based on the three main categories of approaches covering Heuristic Rules, Classical optimization techniques and Neural network optimization systems.

Recently *Proth (2007)* [133] extended the survey by *Panwalkar et. al. (1978)* [130] on Scheduling Rules. As the initial survey concentrated on 113 rules used in Priority Rules, heuristics and scheduling rules, Proth listed the following as the most popular in the past decade:

- *Shortest Processing Times (SPT)* where tasks with the smallest processing times have the highest priority to be scheduled, followed by tasks with next smallest processing time.
- *Largest Processing Times (LPT)* is the reverse version of SPT where tasks with the largest processing times must be scheduled with highest preference.
- *First In First Out (FIFO)* rule dictates that the earlier the task arrive, the earlier it must be processed or scheduled.
- *Last In First Out (LIFO)* is the direct reverse of FIFO where the last tasks arriving will have highest priority.
- *Shortest Setup Time (SST)* is relevant when setup times are fixed and known, the scheduling priority is based on the tasks with the least amount of setup time required.
- *Largest Setup Time (LST)* rule is similar to SST in characteristics, but the tasks with largest setup time will be given preference over the other tasks.
- *Shortest Processing and Setup Time (SPST)* is the combined version of SPT and SST.

- *Largest Processing and Setup Time (LPST)* is the combined version of LPT and LST.
- *Earliest Due Date (EDD)* rules that tasks with nearest due dates must have highest priority.
- *Latest Due Date (LDD)* is the reverse version of EDD, hence highest priority is for tasks with latest due dates.

Proth (2007) [133] also described the concept of dynamic scheduling approach and how the real time assignments used in solving scheduling problems, takes a different approach in the form of meeting new scheduling demands, through rescheduling of the whole schedule when required. Another effective approach used is the *Branch and Bound* approach that utilizes the dynamic tree representation of the solution space representing the scheduling problem. This enumerative approach will search the space of all feasible schedules. This formulation procedures and rules will remove large portions of the tree until a lower bound associated with the optimal solution is found. This approach has been successful in solving single machine by *Chang (1999)* [29] and general shop scheduling problems by *Kindt et. al. (2004)* [147].

Also the *Shifting Bottleneck* procedure from *Adams et. al. (1988)* [2] is one of the most popular heuristics proposed for solving the job shop scheduling problems. The actual strategy involves considering the M machine problems and solving each subproblem. The solutions obtained on each machine are ranked, with the largest lower bound is identified as the bottleneck machine. The scheduling of the jobs are done based on this bottleneck machine.

Tabu search (TS) [54] algorithms is also one of the most effective approaches for solving scheduling problems. As TS utilizes the principle of neighborhood structures and move evaluation strategies, these principles have been proven successful in obtaining optimum solution within a fast period of time. Among the researchers that have developed Tabu Search approaches for solving the job shop scheduling problem are *Nowicki and Smutnicki (2005)* [128] and *Zhang et. al. (2007)* [159].

One of the latest technique used is the *fuzzy logic*, that has the inclusion of uncertainty characteristics infused in the approach. The modelling is based on fuzzy logic consisting of "IF-THEN" rules to represent the relations between input and output variables of the scheduling problem.

One example was developed by *Bilkay et. al. (2004)* [15] who developed a fuzzy logic-based decision-making algorithm to determine the scheduling priorities for part types in the flexible manufacturing system problem.

Another variation to the local search methods is the *Genetic Algorithm (GA)* inspired by the process of Darwinian evolution. GA utilizes a population of solutions in its search based on survival-of-the-fittest step of mutation and crossover in generating new solutions until the best solution is found. Researchers such as *Gonçalves et. al. (2005)* [56], *Park et. al. (2003)* [131] and *Mattfeld et. al. (2004)* [118] were successful in applying the GA in solving the job shop scheduling problem.

3.5 Modelling The Cyclic Job Shop Scheduling Problem

As already briefly described in previous chapter, the Cyclic Job Shop Scheduling Problem (CJSSP) is an extension of the job shop scheduling problem (JSSP) that is a combinatorial optimization problem with a complexity of NP-Hard. The job shop scheduling problem involves the scheduling of jobs that comprise of a definite number of operations. The objective associated with this problem is normally minimizing the schedule length to complete all the jobs. The case of cyclic version of job shop scheduling problem involves iterative repetition of jobs to be solved. The objective when solving the scheduling problem involved is slightly different where generally it is the cycle time that is to be minimized. This is based on k occurrences of the jobs. Among the first research into CJSSP are work by *Roundy (1992)* [136] on finite machines processing multiples of the same job while minimizing the cycle time whilst *Draper et. al. (1999)* [39] utilized constraints satisfaction to find minimum cycle time and work in progress in the CJSSP. Also [64] developed a branch and bound approach in solving the generalized version of the cyclic job shop, by incorporating the precedence constraints, but in the context of computer pipeline.

Modelling-wise, the cyclic job shop scheduling problem is extendable from the non-cyclic job shop scheduling problem based on work done by *Hanen (1994)* [64] and *Brucker and Kampmeyer (2005)* [19]. A cyclic

system can be implied to the disjunctive graph (from Section 2.1.6.2) by inserting the source and sink operations and creating an arc from sink node to source node. Also the arc is defined with delay $L_{sink;source} = 0$ and height $H_{sink;source} = 1$. This arc defines that no processing time is associated between the sink and source. It is proven that $H_{sink;source} = 1$ for the cyclic job shop scheduling problem will mean that the optimal cyclic time is equal to the optimal makespan of the equivalent non-cyclic job shop scheduling problem [64].

An extension to the general cyclic job shop, was researched by *Kampmeyer (2006)* [89] includes the *cyclic job shop problems with job repetition* where $H_{sink;source} \geq 1$. This will mean that there will be more than one occurrence of the operation per cycle time cyclic job shop problem with machine repetition utilizing the minimal part set (MPS) from *Hitz (1979)* [73]. This minimal part set (MPS) defines the minimal set of operations necessary to include in a cycle and which must be repeated a certain number of times to complete the whole production order. This scenario still aims to find the minimal cycle time with all MPS having the same machine processing order. Another extension to the cyclic job shop problem is the *cyclic job shop problem with blocking* from *Brucker and Kampmeyer (2005)* [19]. In this case the operations O_i and $O_{i'}$ are blocking. In the case of machine repetition with blocking, the two blocking operations O_i and $O_{i'}$ which are both processed on machine M_{O_i} and $M_{O_{i'}}$, cannot be released from their machines unless the succeeding corresponding operations O_{i+1} and $O_{i'+1}$ can be loaded on their machines $M_{O_{i+1}}$ and $M_{O_{i'+1}}$.

In this section we will describe the parameters in the cyclic job shop problem and propose an approach to model the cyclic job shop scheduling problem that is non-blocking. The approach will then model the problem as a Linear Programming (LP) Problem.

The cyclic job shop is generally characterized by the following: More precisely the systems is defined as follows:

- A set of $M = m_1, m_2, \ldots, m_M$ machines.
- A set of $N = j_1, j_2, \ldots, j_N$ Jobs,
- Each job has N_j cyclic operation, $O_{i;j} = O_{1;j}, O_{2;j}, \ldots, O_{N_j;j}$.
- A machine can execute only one operation at a time, and in any given time frame.

- The operations are linked by the Precedence constraints.
- The operations are to be processed and assigned to a particular dedicated machine from a Set M.
- Each machine may have a set of operations that has to be processed.
- Each job has its own unique route through the machines, independent of any other jobs.
- Each task is assigned to one and only one machine. (This will be extended to a group of machines).
- Each task must be processed on the assigned machine with uninterrupted processing time without any preemption.

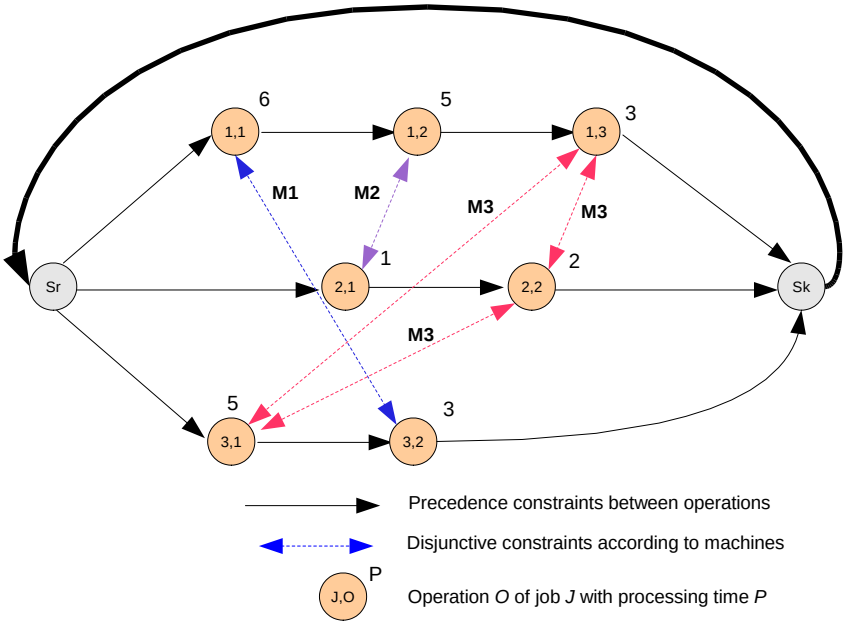


Figure 3.1: Example of a Cyclic Job Shop

An example of a cyclic job shop can be seen in Figure 3.1 that consists of 3 jobs. Job 1 has 3 operations (i.e. (1,1), (1,2) and (1,3)), Job 2 has 2 operations(i.e. (2,1) and (2,2)) and Job 3 also has 2 operations (i.e. (3,1) and (3,2)). Processing times for each operation are deterministic and known prior to processing, where for (1,1), (1,2), (1,3), (2,1), (2,2), (3,1) and (3,2), the processing times are defined by 6, 5, 3, 1, 2, 5, 3 respectively. In our example here, there are only 3 machines available in the job shop, M1, M2, M3. Here operations (1,1) and (3,1) can be processed on machine M1, while operations (2,1) and (1,2) can be processed on machine M2. Lastly operations (3,1), (1,3) and (2,2) can be processed on machine M3.

The required processing precedence relations are (1,1) precedes (1,2), operation (1,2) precedes (1,3), operation (2,1) precedes (2,2) and operation (3,1) precedes (3,2). However due to the cyclic nature of this particular job shop these relationship are cyclic and can be repeated in the schedule.

3.5.1 The Proposed Approach

The system consists of a set of machines and a set of operations or tasks assigned to each machine. The whole system is governed by two types of constraints; conjunctive and disjunctive constraints. The conjunctive constraints deal with the precedence relations between different operations and the disjunctive constraints deal with the operations that are assigned to the same machine.

3.5.1.1 Model Assumptions

Our approach is based on the following assumptions or conditions for application:

Assumption 3.5.1 *The information relating to the jobs and machines is readily available.*

Assumption 3.5.2 *The processing time required by the operations are deterministic and known.*

Assumption 3.5.3 *All jobs and operations are available at time zero.*

Assumption 3.5.4 *Unlimited storage space is available, so no penalties associated with jobs waiting.*

Assumption 3.5.5 *The jobs with fixed number of operations to be completed in a predetermined given order.*

Assumption 3.5.6 *All set-up times on the machines are included in the processing time.*

Assumption 3.5.1 is vital to ensure that knowledge on the number of machines, the jobs and corresponding characteristics of all the operations

are known prior to modelling the scheduling problem. From *Assumption 3.5.2* and *Assumption 3.5.6*, the deterministic nature of the processing time of the operations eliminate a stochastic effect to the modelling, and allow for inclusion of setup time (if any exists) into the processing time. Whereas *Assumption 3.5.3*, by allowing for the availability of all the jobs and operations at time zero, we allow for the operations or jobs to be selected to start immediately when the machines are available, thus minimizing slack time or unnecessary waiting time on the machines. This condition of minimum waiting time on machines from immediate processing of the waiting operation can be assumed from *Assumption 3.5.4*. *Assumption 3.5.5* defines the cyclic job shop with predetermined order which the operations must be processed in order to complete each job. All the above assumptions are vital to allow for an accurate modelling and representation of the cyclic job shop problem.

3.5.1.2 The Proposed Model

We define S_i^k as the starting time of the occurrence k of the Operation O_i, and let p_i be the its processing time. Based on the availability of machines in the cyclic job shop to process the required operation, disjunctive constraints for every machine are defined between operations O_i and O_j where $i \neq j$ using the following expression:

$$(S_i^k + p_i) \leq S_j^l \vee (S_j^l + p_j) \leq S_i^k \qquad S_i, S_j \in T; k, l \in Z \tag{3.8}$$

So from above it is deduced that $S_i^k + p_i \leq S_j^l$ is true if the Operation O_i is processed before O_j while $S_j^l + p_j \leq S_i^k$ becomes true in the reverse case. In cyclic job shop cases, jobs are composed of operations. Using unit time as a standard to prevent overlapping of operations being processed on each machine, we assume that each machine can only process one operation per unit time. From this, we can assume that only one part of Equation 3.8 can hold true. We assume this knowledge of the list of operations to be processed on particular machine m is known prior to starting the scheduling process. So any operations not being processed on machine m at a particular time must be in a waiting state.

This can be rewritten in an alternative expression, by introducing the

parameter δ_{ikjl} as follows:

$$\delta_{ikjl}(S_i^k - S_j^l + p_i) \leq 0 \qquad S_i, S_j \in T; k, l \in Z \tag{3.9}$$

$$(1 - \delta_{ikjl})(S_j^l - S_i^k + p_j) \leq 0 \qquad S_i, S_j \in T; k, l \in Z \tag{3.10}$$

where δ_{ikjl} as the Kronecker symbol that will ensure only one condition is true, given by

$$\delta_{ikjl} = \begin{cases} 1 & \text{if } S_i^k - S_j^l \leq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

As we are considering the cyclic version of the job shop scheduling problem, we also consider the disjunctive constraints in relation to operation from different iterations. This is obvious from denotation of k and l depicting two different iterations, associated with the operation i and j. In the case when considering operations from the same iteration, then iteration $k = l$.

The above expressions are based on several assumptions made on the machines. These include:

1. Each machine is uniform and have same processing speed.
2. Each machine has predefined loading abilities, as such it can only accommodate one operation per unit of time or be in a waiting state if operation are unavailable to be loaded.
3. Each machine is not subjected to any interruptions that may render the machine breaking down, during the entire time the operation is being processed on the machine.
4. No corresponding set-up time relating to the operations occur on each machine and if there exist any set-up time required on the machine, this would have included in the processing time of the operation.
5. Once the operation has been loaded on the machines, the operations may not be stopped and re-continue i.e. no preemption allowed for each operation.

To illustrate the above disjunctive constraints, we consider the example given in Figure 3.1, in the case for Machine $M3$, where 3 operations (3,1),

(1,3), (2,2) can be processed on that particular machine. As such the constraints involved are defined as follows:

$$\begin{aligned}
\delta_{(3,1)(1,3)}(S^k_{(3,1)} - S^l_{(1,3)} + p_{(3,1)}) &\leq 0 \\
(1 - \delta_{(3,1)(1,3)})(S^l_{(1,3)} - S^k_{(3,1)} + p_{(1,3)}) &\leq 0 \\
\delta_{(3,1)(2,2)}(S^k_{(3,1)} - S^l_{(2,2)} + p_{(3,1)}) &\leq 0 \\
(1 - \delta_{(3,1)(2,2)})(S^l_{(2,2)} - S^k_{(3,1)} + p_{(2,2)}) &\leq 0 \\
\delta_{(1,3)(2,2)}(S^k_{(1,3)} - S^l_{(2,2)} + p_{(1,3)}) &\leq 0 \\
(1 - \delta_{(1,3)(2,2)})(S^l_{(2,2)} - S^k_{(1,3)} + p_{(2,2)}) &\leq 0
\end{aligned}$$

As such for a particular cycle, there would exist $\frac{N_m!}{(2)!(N_m-2)!}$ disjunctive constraints per machine with N_m operations allowed to be processed on it.

Based on the above notations, the precedence constraints (or conjunctive constraints) for operations from same job between tasks, can be expressed as:

$$S_i - S_j + p_i \leq 0 \tag{3.12}$$

Therefore, we can say that the operation i precedes the operation j. Note that any occurrences of operations i and j can also be constrained such that the model can encompass any combination of the occurrences in order to respond to the system output. So from above constraint 3.12, the number of precedence constraints depends on the number of operations required to complete that particular job. For example, in the case of a job with 3 operations (i.e (1,1), (1,2), (1,3) and (1,4)), the precedence constraints for a particular cycle can be defined as:

$$\begin{aligned}
S_{(1,1)} - S_{(1,2)} + p_{(1,1)} &\leq 0 \\
S_{(1,2)} - S_{(1,3)} + p_{(1,2)} &\leq 0 \\
S_{(1,3)} - S_{(1,4)} + p_{(1,3)} &\leq 0
\end{aligned}$$

Generally, in the case of a particular job requiring N_j operations to be processed in order to complete that job, there would exist $(N_j - 1)$ number of precedence constraints.

A model using linear constraints between occurrences is proposed in *Munier (1996)* [124] where an occurrence i for task k at cycle n is rep-

resented as $i = \alpha_k n + \beta_k$, where α and β are natural numbers. This will be further discussed in ***Section*** 4.3 where a approach is proposed to solve the CJSSP with linear constraints.

The end of a particular schedule before being utilized as a cyclic schedule is taken as the point of the completion time of the last operation. This is taken as the maximum completion of the jobs. The completion time of a particular job j can be calculated from

$$C_j = S_{N_j;j} + p_{N_j;j} \tag{3.13}$$

Taking the maximum completion time from all the jobs, from expression:

$$C_j^{max} = \max_{j=1}^{N} \{C_j\} \tag{3.14}$$

This can also be calculated from maximum completion from the full list of operations in the CJSSP.

$$C_{i;j}^{max} = \max_{j}^{N} \left\{ S_{N_j;j} + p_{N_j;j} \right\} \tag{3.15}$$

The cycle time can then be calculated from the difference of maximum completion time of operations to the earliest start time.

$$\tau = C_{i;j}^{max} - S_{i;j}^{min} \tag{3.16}$$

The above Equation 3.16 yields a feasible solution, however it may not yield the most optimal cycle schedule. This can be further achieved if the next iterated schedule cycled can be started at an earlier time, pending on no violations in constraints. This global shift of the defined schedule will reduce the cycle time.

Hence using expression $S_{i;j}^{k+1} - S_{i;j}^{k}$ as the objective function, will give us the optimal and feasible cyclic schedule.

This linear formulation represents the shift between occurrences of tasks which are linked by a precedence constraint. The general problem we want to optimize can be formulated as follows:

$$\text{Minimize} \quad f(S) = \sum_{j=1}^{N} \sum_{i=1}^{N_j} (S_{i;j}^{k+1} - S_{i;j}^{k}) = C^T \vec{S_n} \tag{3.17}$$

Subject to

$$S_i - S_j + p_i \leq 0 \quad (i,j) \in \mathcal{C} \tag{3.18}$$

$$S_n^k \geq 0 \quad n \geq 0 \quad k > 0 \tag{3.19}$$

$$\delta_{ikjl}(S_i^k - S_j^l + p_i) \leq 0 \quad (i,j) \in \mathcal{D} \tag{3.20}$$

$$(1 - \delta_{ikjl})(S_j^l - S_i^k + p_j) \leq 0 \quad (i,j) \in \mathcal{D} \tag{3.21}$$

Where $\mathcal{C}$ is the set of operations, which are conjunctive constrained and $\mathcal{D}$ a set of couples of operations, which are assigned to the same machines. Also $k = \alpha n + \beta$ for occurrence of task i in cycle n.

The problem (3.17) is called a Linear Programming (LP) Problem with inequality constraints. The constraint (3.18) above represents the precedence rule. Here task i precedes task j. The constraint (3.19) ensures that start times for all tasks will start at or after time zero. Constraints (3.20) and (3.21) represent the disjunctive rule for tasks j and i which require the same machine. Depending on whether δ_{ikjl} is conditioned as in (3.11), only one of the task is processed on the machine at any instant of time.

In order to be able to use Neural Network technique, the problem (3.40) should be transformed to its equivalent unconstrained optimization problem. This transformation is accomplished through modification of the objective function so that it includes terms that penalize every violation of the constraints.

The inequality constraints described in the problem (3.17) can be rewritten as

$$r_k(S_n) = S_{m_n,i} - S_{m_n,j} + p_i \quad \forall n \geq 0 \quad (i,j) \in \mathcal{C} \tag{3.22}$$

$$r_{\Gamma+k}(S_n) = S_{m_{n_1},i} + S_{m_{n_2},j} + p_i \quad (i,j) \in \mathcal{D} \tag{3.23}$$

where $\Gamma = \mid \mathcal{C} \mid$ = Cardinality of $\mathcal{C}$, and $\varphi = \mid \mathcal{D} \mid$. We denote $\xi = \Gamma + \varphi$.
From equation (3.22), $S_{m_n,i}$ depicting start times of task i occurring at $\alpha_i n + \beta_i$ and $S_{m_n,j}$ for task j occurring at $\alpha_j n + \beta_j$ at cycle n.Also from equation (3.23), $S_{m_{n_1},i}$ for task i occurring at $\alpha_i n_1 + \beta_i$ and $S_{m_{n_2},j}$ for task j occurring at $\alpha_j n_2 + \beta_j$.
However for simplicity in formulating the initial energy function, we assumed that α_i, α_j equals 1 and β_i, β_j as 0 in both equations (3.22)

and (3.23). Thus the resulting equations become as follows:

$$r_k(S_n) = S_{n,i} - S_{n,j} + p_i \quad \forall n \geq 0 \quad (i,j) \in \mathcal{C} \tag{3.24}$$

$$r_{\Gamma+k}(S_n) = S_{n_1,i} + S_{n_2,j} + p_i \quad (i,j) \in \mathcal{D} \tag{3.25}$$

In the following, we formulate a suitable energy function that can be easily solved by a Neural Network. The energy function needs to be constructed so that it penalizes every violation of the inequality constraints.

Consider an energy function defined as

$$E(S_{n,k}) = F(S_n) + KP(S_n) \tag{3.26}$$

where $F(S_n)$ is the cost function expressed as $C^T S_n$, $P(S_n)$ is the penalty function given by $\sum_{i=1}^{\xi} \varphi[r_i(S_n)]$ and K is the penalty factor. Giving the following:

$$E(S_{n,k}) = C^T S_n + K \sum_{i=1}^{\xi} \varphi[r_i(S_n)] \tag{3.27}$$

β_i is considered to be $= 0; i = 1, \cdots, N$

$$\varphi[r_i(S_n)] \begin{cases} = 0 & \text{if} \quad r_i(S_n) \leq 0 \\ > 0 & \text{if} \quad r_i(S_n) > 0 \end{cases} \tag{3.28}$$

$$and \quad S_{n,k} \geq 0; \forall n \geq 0; k = 1, \cdots, N \tag{3.29}$$

The function $\varphi(\vartheta)$ has been chosen so that the property (3.28) is fulfilled. K is a positive parameter that controls the scaling factor between the cost term and the penalty term of the unconstrained optimization problem in (3.26). The scaling factor should be chosen such that the optimal solution can be reached and the constraint violation is also penalized. It can be easily shown that the two problems become equivalent as $(K \to +\infty)$. For that reason, K is commonly selected as a sufficiently large positive number.

3.6 Modelling The Cyclic Flexible Manufacturing Systems Scheduling Problem

Flexible manufacturing systems (FMSs) are widely applied in manufacturing floor with high variety of part types to be manufactured. There is also the application of a transportation system that connects all the machines. Compared to the job shop, FMS has the flexibility of job shops while approaching the efficiency of transfer lines. Scheduling in FMSs differs from that in a conventional job shop because of the availability of alternative manufacturing resources resulting in routing flexibility. Also FMS machines increase output by eliminating the bottlenecks. Bottleneck machines often occur when alternate routes are not feasible.

Machines in all FMSs are able to produce several types of parts simultaneously. This is due to the tooling system on the machine that allows for quick changeover and the ability to hold a number of tools per machine hence capable of performing a number of different operations. A FMS is also characterized by non-preemptive operations, which means that each machine has to finish the operation it has started before freeing the part from *Chretienne et. al. (1995)* [33]. Following the reasons mentioned, the flexibility characteristics of FMS allows for the choice of one or more machine for each operation and one or more processes to manufacture each part type, thus different part types can be produced simultaneously. These mentioned characteristic have a great influence on the productivity and workloads of the machines/work stations.

In the case of acyclic Flexible Manufacturing Systems, a global optimization of the demand must be found through the effective scheduling of all operations. Cyclic Flexible Manufacturing Systems (CFMS) however, are based on the cyclic version of the deterministic FMS where a framework schedule is repeated for a number of occurrences in order to reduce the number of scheduled operations and to fulfill the required demand. By computing a single schedule in a cycle, this approach in solving the cyclic version, reduces the combinatorial complexity. This particular cycle will then be scheduled and repeated until all quantities of the part types are fulfilled.

As like any other shop environment, FMSs are greatly influenced by circumstances and disruptions caused by unforeseen machine breakdowns, increased order priorities, rush order arrival and order cancellations.

In order to formulate the FMS problem, the objective is to find a feasible schedule for a given set of part types so that some criteria can be optimized. The criterion may be chosen from the number and type of jobs, number of tasks in each job, number and types of machines available, processing and setup times of tasks on machines, order due dates, release time of jobs into the shop floor and performance criterion to be chosen.

Among the common objective function associated with scheduling in cyclic FMS are: minimum mean flow time for part types to be processed:

$$\text{Mean flow time} = \frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{N_j} C_{i;j} \tag{3.30}$$

minimum mean tardiness:

$$\text{Mean tardiness} = \frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{N_j} C_{i;j} - d_{i;j} \tag{3.31}$$

maximum average resource utilization for the machines in the FMS.

$$\text{Average Utilization} = \frac{1}{M}\sum_{k=1}^{M}\sum_{i=1}^{N_j} \frac{\text{Total busy Time}}{max(C_j)} \tag{3.32}$$

Hillion and Proth (1989) [72] concluded that it is always possible to maximize the productivity of a cyclic FMS, by fully utilizing the bottleneck resource. This can be completed by acquiring enough Work in Progress (WIP) in the system. Hence by keeping WIP values at a minimum, productivity is maximized. The Work in Progress (WIP) are the parts awaiting to be processed while the machines are busy.

The cyclic behavior of the FMS will reduce the complexity of the general scheduling problem. By working to find a solution that will minimize the Work in Progress (WIP), we will satisfy the economical constraints related to storage space and cost while adhering to resource constraints of the shared resources in the FMS.

We can base our approach on a fixed cyclic time horizon. Using this horizon as a hard constraint in pursuing our objective as maximizing throughput is done by keeping the cycle time of the system to the minimum. We try to solve the minimization of the Work in Progress (WIP)

from the determination of a cyclic characteristic that is based on the optimal speed of the system. By settling on the optimal speed of the system we maximize throughput of the system.

3.6.1 The Proposed Approach

3.6.1.1 The Proposed Model

This problem is slightly different from the classical cyclic job scheduling where the cycle time is to be minimized subject to some conjunctive and disjunctive constraints. In this problem we want to minimize the WIP subject to the same constraints, but the cycle time is known and fixed and it is deduced from the most loaded machine in the system. Therefore, the load of a machine k in the cyclic FMS is denoted by L_k and is given by:

$$L_k = \sum_{i=1}^{N} \sum_{j=1}^{N_i} \delta_{i,j}^{k} p_{i;j} \qquad \forall k \in \{1, 2, \cdots M\} \tag{3.33}$$

where $\delta_{i,j}^{k}$ is a Kronecker symbol indicating whether the operation $O_{i;j}$ is assigned to the machine k or not ($\delta_{i,j}^{k} = 1$ if $m_{i,j} = k$).

The *cycle time* C_T of the system is defined as the maximum time needed for the critical machine to process all operations assigned to it without any slack time. All subsequent patterns of a schedule should be repeated based on this minimal cycle time.

$$C_T = \max_{k=1,2,\ldots,M} (L_k) \tag{3.34}$$

To further describe our approach in more detail, we use the FMS example from Figure 3.2. In this example, there are three jobs with Job A comprising of 6 operations, Job B from 5 operations and Job C with 4 operations. The operations in the example are labelled as $(Job, Operation)$ in the Gantt chart. Using our approach, by maximizing the makespan, we need to locate the critical machine:

Step 1. *Locate critical machine* (or bottleneck machine). Using the above equation 3.33. We find that the loads are $L_k = 6, 10, 4, 9, 6$. So taking the maximum load of the 5 machines, we see that *Machine 2* has the most load of 10 unit time. So the bottleneck machine is *Machine 2*.

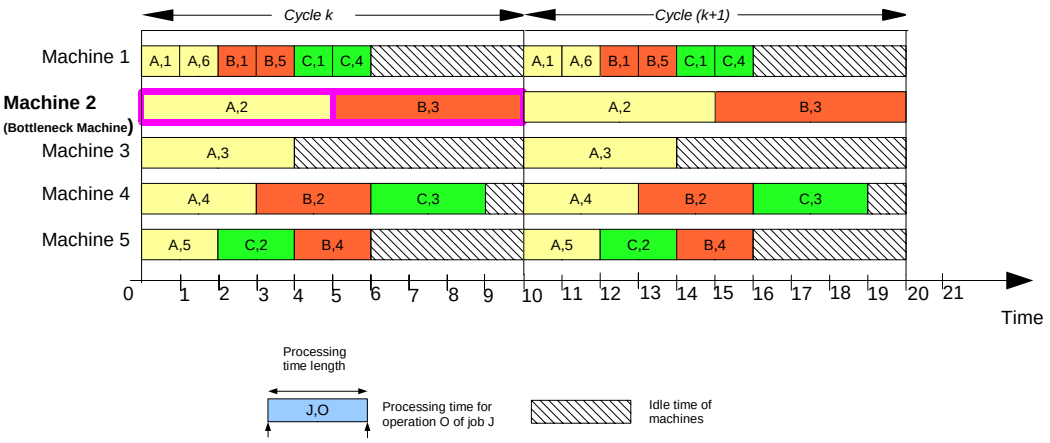


Figure 3.2: FMS Example with *Machine 2* as Bottleneck Machine.

Step 2. *Identify cycle time for FMS*. Now we know that the bottleneck machine is Machine 2, using equation 3.34, cycle time is the sum of processing time of all operations to be processed on the bottleneck machine. Cycle time is 10 unit of time. So using this cycle time, we can identify the Work in Progress of the FMS by scheduling all operations within this time range.

3.6.1.2 The Constraint Cycle

In addition to precedence and resource constraints, which characterize the formulation of the scheduling problem, we introduce the notion of a *constraint cycle* as we are only focusing on the cyclic nature of the FMS scheduling problems. The *constraint cycle* will guarantee that the starting times of all operations are within the cycle time. Some operations may start within the current cycle and end within the following cycle. These operations are called *overlapping operations*. Therefore, an operation $O_{i;j}$ is said to be overlapping if the completion time exceeds the cycle time range, expressed by $S_{i;j} + p_{i;j} > C_T$.

3.6.1.3 Disjunctive Constraints

Similar to the approach proposed in CJSSP, the *disjunctive constraints*, also called resource constraints, are defined between the operations assigned to the same machine. Given two operations $O_{i;j}$ and $O_{i';j'}$ assigned to the same machine k; $m_{i;j} = m_{i';j'} = m$. If the operation $O_{i;j}$ is processed before $O_{i';j'}$, then $O_{i;j}$ cannot be an overlapping operation by being loaded

on the machine k; $S_{i;j} + p_{i;j} \leq S_{i';j'}$.

In the case where the operation $O_{i,j}$ is performed after $O_{i';j'}$, the inequality $S_{i';j'} + p_{i';j'} \leq S_{i;j}$ will hold true, so in this case $O_{i',j'}$ can be an overlapping operation. For every machine m, there exists disjunctive sets $h = 1, 2, \cdots \zeta_m$ where h represents sets of couples of operations assigned to machine m. Therefore, its conformity with respect to the start time of the next occurrence of $O_{i',j'}$ has to be checked; $\left(S_{i,j} + p_{i,j}\right) \bmod C_T \leq S_{i',j'}$. Based on this analysis, disjunctive constraints can be defined as follows:

$$\begin{aligned}
&(\delta_{ij,i'j'})[(S_{i';j'} + p_{i';j'} \leq S_{i;j}) \wedge \\
&\qquad (\omega_{i;j}) \left\{ (S_{i;j} + p_{i;j}) \bmod C_T \leq S_{i';j'} \right\}] \\
&(1 - \delta_{ij,i'j'})[(S_{i;j} + p_{i;j} \leq S_{i';j'}) \wedge \\
&\qquad (\omega_{i';j'}) \left\{ (S_{i';j'} + p_{i';j'}) \bmod C_T \leq S_{i;j} \right\}]
\end{aligned}$$

where parameter $\omega_{i;j}$ indicates whether the operation $O_{i;j}$ is overlapping or not; ($\omega_{i;j} = 1$ if $S_{i;j} + p_{i;j} > C_T$), and $\delta_{ij,i'j'}$ indicates which operation of $O_{i;j}$ and $O_{i';j'}$ is started first within a given schedule ($\delta_{ij,hl} = 1$ if $S_{i';j'} < S_{i;j}$).

3.6.1.4 Conjunctive Constraints

Although the FMS environment allows for flexible routing of processing, the consideration of conjunctive constraints may not be a major constraint due to the simultaneous processing ability of the machines. We consider this constraint in preparation to measure the WIP in the system. *The conjunctive constraints* (or precedence constraints), have been defined between the operations of a job. For each job in the system an order in which the operation should be executed is defined. For the sake of clarity, the operations of a job are indexed according to their order of execution. Defining the occurrence of an operation as the instance when the operation is being processed ($k = 1, 2, \ldots \infty$). So two consecutive operations $O^k_{i;j}$ and $O^k_{i;j+1}$ with start times $S_{i;j}$, $S_{i;j+1}$ and with the same occurrence (k) are linked by the following precedence constraint:

$$\begin{aligned}
&S^k_{i;j} + p_{i;j} \leq S^k_{i,j+1} \\
&\forall i \in \{1, 2, \cdots N\}, \forall j \in \{1, 2, \cdots N_i\}
\end{aligned} \tag{3.35}$$

The effect in violation of the conjunctive constraints on the WIP will be describe further in the chapter.

3.6.1.5 The Objective Function

The Work in progress (WIP) is defined to be the sum of WIPs from each individual job in the system. The WIP E_i of a job i is the number of jobs of type i which have started and have not yet terminated during the current cycle time.

Here we associate the use of the conveyor or pallets as the transportation system with the WIP of the FMS. Note that in cyclic task scheduling the system will deliver one piece of each product type or job. This will make available the conveyors or pallets carrying the terminated occurrences of each job. These conveyors or pallets can then be used to start new occurrences of different jobs in the system during the current or next cycle. A measurement of the number of WIPs depends on the number of dedicated conveyors or pallets required.

Several assumptions has been made in modelling the cyclic FMS. As regards to the FMS production characteristics, such as operating times are deterministic, and all parts or jobs are available at time zero. We also assume that a dedicated conveyor or pallet are allocated for a part, one machine at any one time to eliminate the existence of structural deadlocks. Although generally the machines in FMS can process a variety of processes simultaneously, in our model, each machine is unique and has its own set of dedicated operations.

In order to accurately measure the WIP, the conjunctive constraints are considered in the objective function. If a conjunctive constraint is not satisfied, the two operations involved are not considered to be in the same occurrence. The WIP will be increased by one, as each occurrence deals with one piece of the product. Let $C_{i,j}$ be the WIP variable between two successive operations $o_{i,j}$ and $o_{i,j+1}$. $C_{i,j}$ is defined as follows:

$$C_{i,j} = \begin{cases} 0 & \text{if } S_{i,j} + p_{i,j} \leq S_{i,j+1} \\ 1 & \text{otherwise} \end{cases} \tag{3.36}$$

$$\forall i \in \{1, 2, \cdots N\}, \quad \forall j \in \{1, 2, \cdots N_i\}$$

The WIP variable, C_{i,N_i}, between the last and the first operations should

also be considered. C_{i,N_i} will be used to model the notion of *linked occurrences*. Two occurrences are said to be linked when one occurrence is finished and another occurrence of the same job is started during the same cycle.

Some operations may overlap between two consecutive cycle-times. An extra WIP should be added when the overlapping operation is not finished before the start of the next operation. Let $\tilde{C}_{i,j}$ be the WIP variable between an overlapping operation j and its successor $j+1$. It is given by the following:

$$\tilde{C}_{i,j} = \begin{cases} 0 & \text{if } 0 \leq S_{i,j} + p_{i,j} - C_T \leq S_{i,j+1} \\ 1 & \text{otherwise} \end{cases} \tag{3.37}$$
$$\forall i \in \{1, 2, \cdots N\}, \quad \forall j \in \{1, 2, \cdots N_i\}$$

When the last operation of a job is overlapping, the constraint $\tilde{C}_{i,N_i}$ is defined for the same reasons as C_{i,N_i}.

Hence in order to calculate the overall WIP in the FMS, we sum all the WIP from each job as denoted as follows:

$$\equiv WIP = \sum_{i=1}^{N} \sum_{j=1}^{N_i} \left(C_{i,j} + \omega_{i,j}\tilde{C}_{i,j}\right) \tag{3.38}$$

The following Figures 3.3, 3.4, 3.5, 3.6, 3.7 and 3.8 show six common examples of WIP calculated from the different FMS cases.

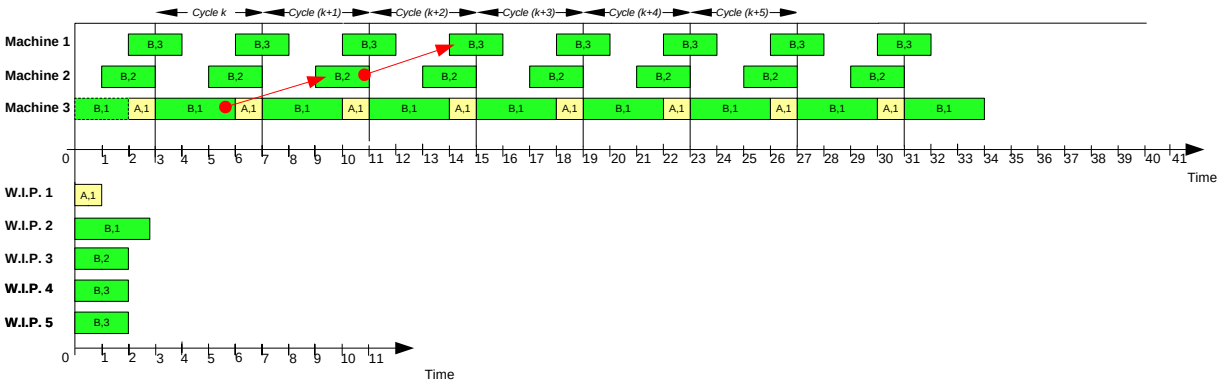


Figure 3.3: Number Of Work In Progress In Case 1

In Fig. 3.9, the FMS example has 3 jobs (A, B, C) running on 5 machines illustrating a cyclic aspect of a scheduling problem. Job A has

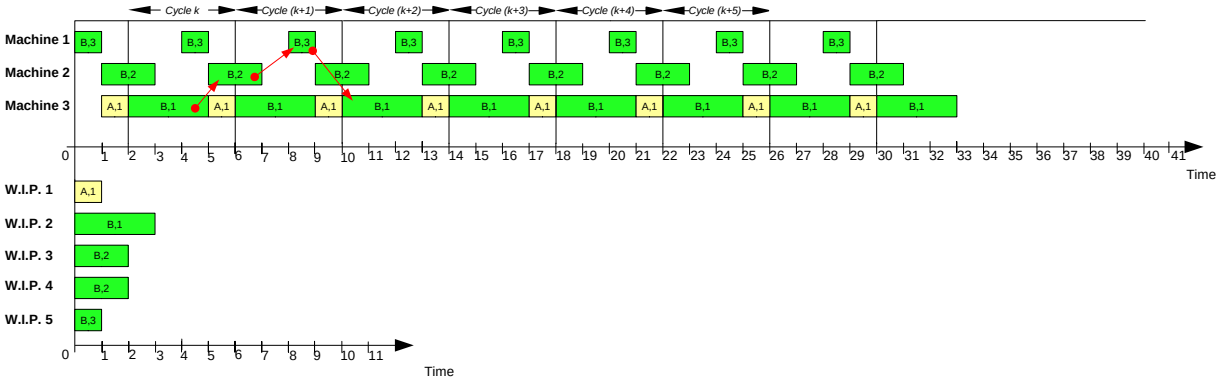


Figure 3.4: Number Of Work In Progress In Case 2

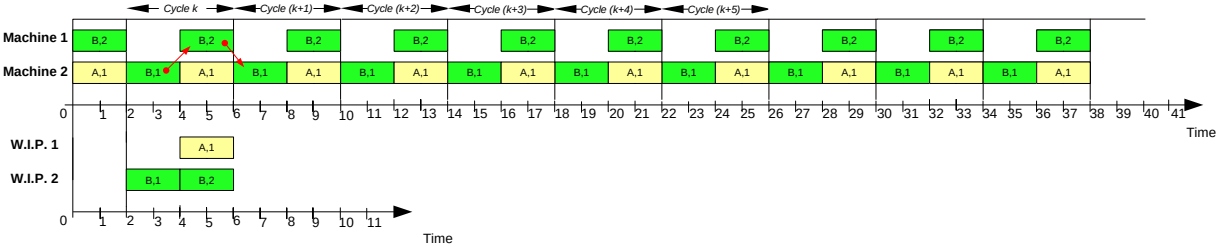


Figure 3.5: Number Of Work In Progress In Case 3

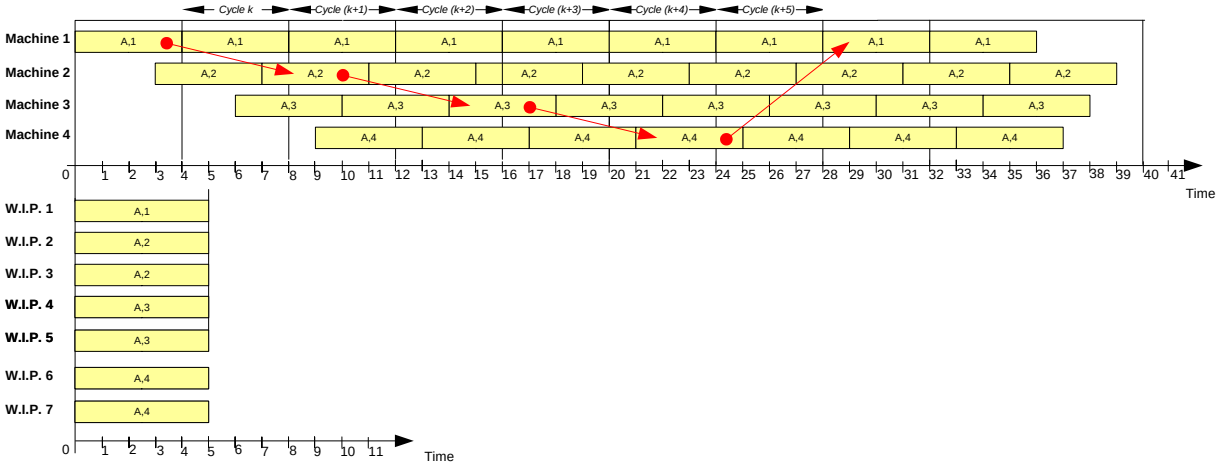


Figure 3.6: Number Of Work In Progress In Case 4

6 operations $(A_1, A_2, \cdots, A_6)$ with their corresponding processing times $(1, 5, 4, 3, 2, 1)$. The job B consists of 5 operations $(B_1, B_2, \cdots, B_5)$ with the following processing times $(1, 3, 5, 2, 1)$. Job C has 4 operations $(C_1, C_2, \cdots, C_4)$ having the corresponding processing times $(1, 2, 4, 1)$ respectively. The WIP is calculated as the total number of $C_{i,j} + \omega_{i,j}\tilde{C}_{i,j}$ occurrences of each job during that cycle. In the example, we have two occurrences of both jobs A and B and only one occurrence of job C and

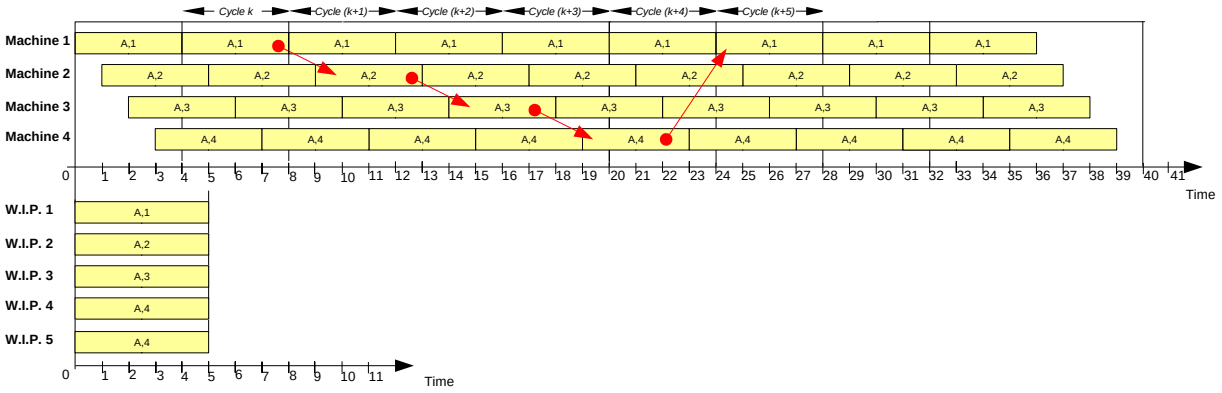


Figure 3.7: Number Of Work In Progress In Case 5

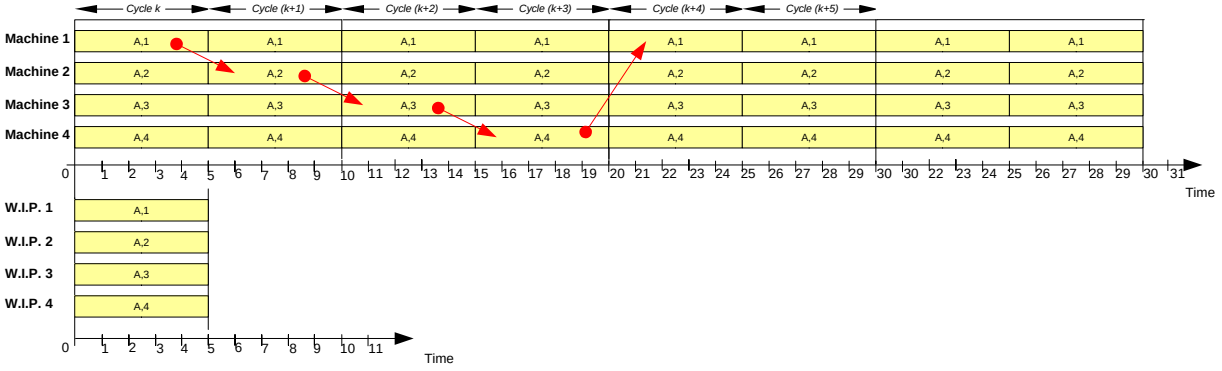


Figure 3.8: Number Of Work In Progress In Case 6

the cycle time is 10, given by machine 2. The value of the WIP is 5 because the conveyors or pallets freed during the cycle are used to start one occurrence of each job during the same cycle. The occurrences $n-2$ and n of each job are linked occurrences because the $(n-2)^{th}$ occurrence finished before the nth occurrence started. In this case the linked occurrences will be counted only once in WIP.

Korbaa et. al. (2002) [98] found that by considering each job separately, the lower bound for WIP can be calculated as from *Campos et. al. (1992)* [24]. The lower bound is the sum of the number of cycles needed for all jobs without taking into account the conjunctive constraints. For each job, the number of cycles needed to complete one occurrence, is calculated. This can be expressed by the following:

$$WIP \geq \sum_{i=1}^{N} \left\lceil \frac{1}{C_T} \sum_{j=1}^{N_i} p_{i,j} \right\rceil \tag{3.39}$$

Given a schedule within a cycle time, the WIP of the system is the sum of

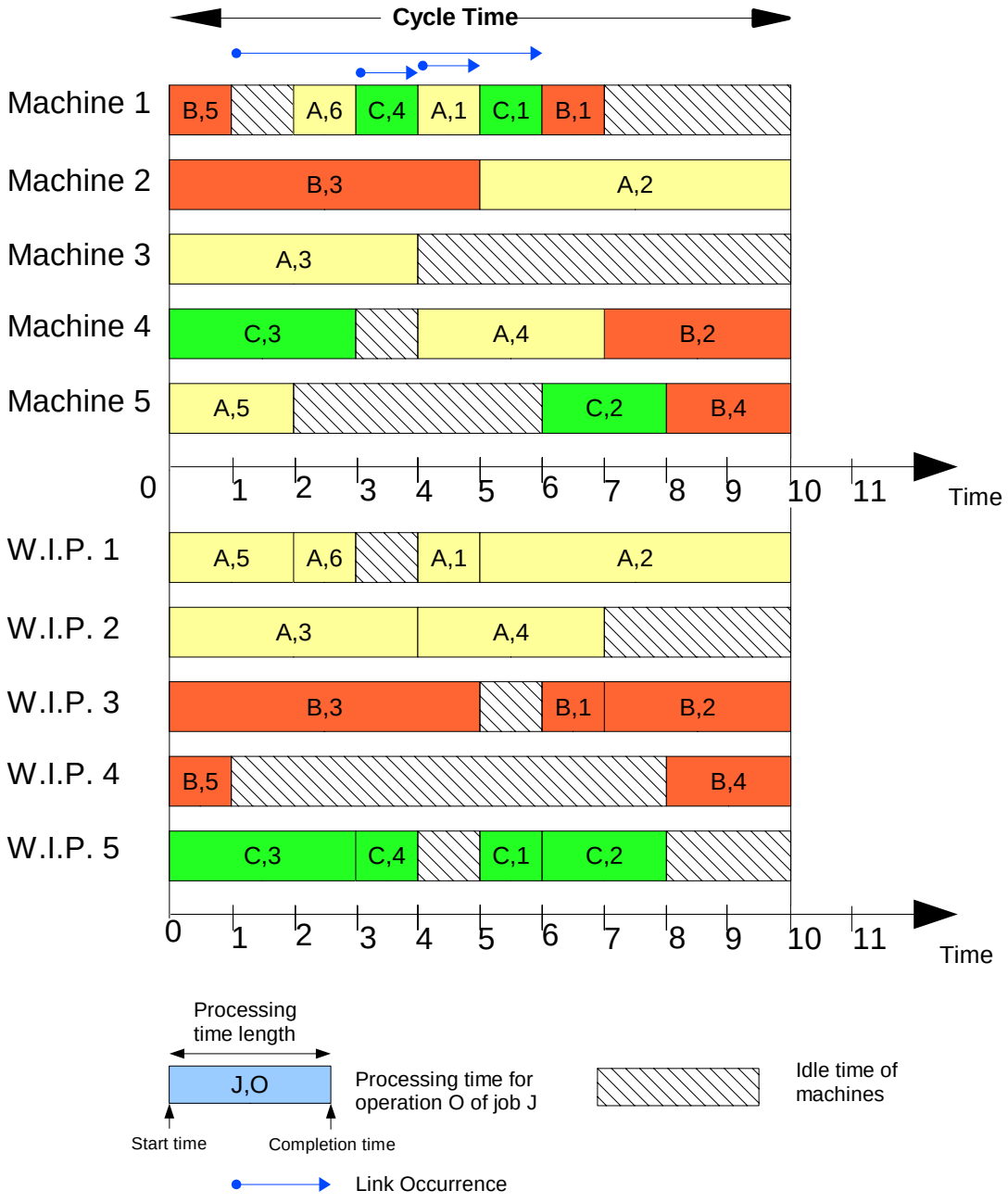


Figure 3.9: Cyclic FMS example with three *Linked Occurrences*.

all Work In Progress of each job E_i. The formulation of the cyclic scheduling FMS problem can then be summarized as follows:

$$\min_{\{S_{i,j}\forall N,N_i\}} f(S) \equiv WIP = \sum_{i=1}^{N}\sum_{j=1}^{N_i}\left(C_{i,j} + \omega_{i,j}\tilde{C}_{i,j}\right) \tag{3.40}$$

Subject to:

$$\begin{aligned}
(\delta_{ijhl})(S_{h,l} + p_{h,l} &\leq S_{i,j}) \\
(1 - \delta_{ijhl})(S_{i,j} + p_{i,j} &\leq S_{h,l}) \\
(\delta_{ijhl})\omega_{i,j}(S_{i,j} + p_{i,j} - C_T &\leq S_{h,l}) \\
(1 - \delta_{ijhl})\omega_{h,l}(S_{h,l} + p_{h,l} - C_T &\leq S_{i,j}) \\
0 \leq S_{i,j} &< C_T
\end{aligned}$$

$$i = 1, 2, \cdots, N, \quad j = 1, 2, \cdots, N_i$$

The problem (3.40) can be transformed into an unconstrained optimization problem form by defining a penalty function of the constraints. Let d_k be the number of operations assigned to machine k. Let o_i^k be the operation i on machine k. The four disjunctive constraints can be further combined into two general forms, first the non-overlapping disjunctive constraints to obtain constraint $r_{i,j}^k$ and second, the overlapping disjunctive constraints represented by $R_{i,j}^k$. The constraints can be rewritten as follows:

$$\begin{aligned}
r_{i,j}^k &= \max\{0,\ \delta_{i,j}^k \left(S_j^k + p_j - S_i^k\right) \\
&\quad +(1 - \delta_{i,j}^k) \left(S_i^k + p_i - S_j^k\right)\} \\
R_{i,j}^k &= \max\{0,\ \delta_{i,j}^k \omega_{i,j}^k \left(S_i^k + p_i - S_j^k - C_T\right) \\
&\quad +(1 - \delta_{i,j}^k) \omega_{i,j}^k \left(S_j^k + p_j - S_i^k - C_T\right)\}
\end{aligned} \tag{3.41}$$

$$k = 1, \ldots, M \quad i = 1, \ldots, d_k - 1 \quad j = i + 1, \ldots, d_k$$

Given the following relaxed problem, which accommodates the penalty function to relax the constraints, the energy function $L(S, \mu)$ is

$$\begin{aligned}
L(S, \mu) &= \sum_{i=1}^{n} \sum_{j=1}^{m_i} \left(C_{i,j} + \omega_{i,j} \tilde{C}_{i,j}\right) \\
&\quad + \tfrac{1}{2} K \sum_{k=1}^{M} \sum_{h=1}^{\zeta_k} \mu_{h,k} \left(r_h^k(S) + R_h^k(S)\right)^2
\end{aligned} \tag{3.42}$$

where K is the penalty parameter for disjunctive constraints, ζ_k disjunctive sets for machine k, and $\mu_{h,k}$ is Lagrange multiplier for disjunctive set(h, k). The above form can then be used in mapping to a neural network in order to be solved in ***Section*** 4.2.

Hsu et. al. (2007) [78] found that number of admissible solutions for each problem is given by

$$N_s = \prod_{k=1}^{m} \varphi k \tag{3.43}$$

where m is the machines in the FMS and

$$\varphi_k = \sum_{i=1}^{n} \sum_{j=1}^{n_j} \left\{ (p_{i,j} - 1) \bullet \frac{(nf_k + n_k - 1)}{nf_k} \right\} \tag{3.44}$$

Here, the number of operations per machine k, using the following: $\sum_{i=1}^{n} \sum_{j}^{n_j} \delta_{i,j}$ where $\delta_{i,j} = 1$ if $k = m_{i,j}$ and calculating the number of operations possible to fill the cycle time:$nf_k = \tau - W_k$ where in the cycle time τ, the workload of the machine, $W_k = \sum_{i=1}^{n} \sum_{j=1}^{n_j} \delta_{i,j} * p_{i,j}$ where $\delta_{i,j} = 1$ if $k = m_{i,j}$ but $\delta = 0$ otherwise.

3.7 Solving The Cyclic Scheduling Problem

3.7.1 Petri Nets

The Petri Net (PN) approach has also been used for scheduling problems. The modeling of the cyclic scheduling problem is done through the objects *places* and *transition* of the Petri Net. Tokens in the places are moved along the circuit when the transitions are fired and the state of the system can be known from the location of the various tokens.

Timed Event Graph, one of the sub class of the Timed Petri Nets [126], [95] has been used for the cyclic scheduling problem. This PN has exactly one input and one output transition and the weight on each arcs (connecting places to transitions) equals one.

Hillion and Proth (1989) [72] described three circuits that are possible to be used for modelling the scheduling problem: **process circuits** where the tokens model the products, **resource circuits** where in each circuit one token models a machine, and **hybrid circuits** which contain parts of the process and resource circuits. The operating sequences of a job may

be represented by some elementary circuits, and this will also give the optimal schedule when PN reaches the optimal state. The order of loading the operations onto the resources are determined from linking the transitions by circuits. Further description of the use of Petri Net in solving the cyclic scheduling problems can be found in *Hanzalek (1998)* [68], *Nakamura et. al. (2000)* [127] and *Lee et. al. (2001)* [105]

3.7.2 Genetic Algorithm

Another optimization technique that has been used to solve cyclic scheduling problems is the genetic algorithm technique [27], [77]. The schedules must firstly be encoded appropriately using either direct encoding or indirect encoding. In direct coding [157] the chromosome represents the schedule that may require complex genetic operators. However the indirect encoding [14] only uses certain rules for constructing the schedule required.

Selection of the proper encoding must not only map the whole search space, yet be able to represent the optimal solutions [78]. In cyclic scheduling, a finite string of the chromosome corresponds to a point in the search space whereas the sub-chromosome (or gene) is an entity of the schedule e.g. operation number or machine number. The optimization process begins with a preset iteration threshold or a number of generations known in advance. A set of solutions are selected at random to form the initial population. A fitness value is calculated for each individual that reflects the quality of the solution i.e. makespan, completion time or cycle time.

Using a probability proportional to fitness, a set of individuals (or parents) are selected. Then using genetic operators e.g. mutation or crossover, the set of individuals are reproduced giving new sets of solutions. New solutions (or children) may takeover the previous solution and the new fitness of the individuals is calculated. This continues until the threshold value is reached.

The *crossover* involves creating two new chromosomes from two parents. By swapping gene in sides before and after the cutting side among the parent chromosome, the new chromosomes are created. *Mutation* involves a simple swap that exchanges the positions or values of some

genes of the chromosome. Both genetic operators are described in detail in *Hsu et. al. (2008)* [78].

Nakamura et. al. (2006) [126] quoted that solving the cyclic scheduling problem is effective as better fitness chromosomes are more frequently produced for every generation, whilst less fit chromosomes occur less. However the success of this approach does depends on the initial population size, number of new individuals in each generation, the number of generations to be created, probability used in the genetic operators and selection of variant of genetic operator of the cyclic scheduling problem.

3.7.3 Tabu Search

An extension to local search (LS) metaheuristic, tabu search approach was proposed by *Glover (1989)* [54], [55]. Incorporating short term memory ability into the deterministic oriented search procedure, the Tabu search has a Tabu list of restricted solutions. This list forbids search steps in the neighbourhood from returning into either duplicate solution or less favourable solutions. However, this approach also includes the flexibility of escaping from the restriction through the aspiration criteria that allows a move into the Tabu if the step attains a pre-determined level of quality.

A search space must first be defined to cover all possible solutions possible and the neighbourhood solution will contain the next solution generated. There is a limit to the number of moves this approach is allowed to make through a predefined maximum number of iterations. The Tabu list is initially empty but will be populated as the search progresses, and may be updated if the list is finite in size.

Brucker and Kampmeyer (2005) [19] used the Tabu search approach to minimize the cycle time for the cyclic job shop and flow shop. The search space explored, adhered to the precedence and disjunctive height values from the disjunctive graph while the neighborhoods populated using the findings strategies of the best-fit and first-fit strategy. This best-fit strategy used, selected a neighbor with the best solution value by systematically analyzing all neighbors while the first-fit strategy chose the first neighbor with a better solution value than the current one.

3.7.4 Artificial Neural Networks

There is a lack of research using artificial neural network (ANN) specifically into the area of cyclic scheduling, although the area of scheduling has attracted more focus. From Section 3.4, most ANN researches have focused on single machine, parallel machines, flexible manufacturing systems, job shop scheduling and its extensions. The common types of ANN approach include Hopfield Network, back-error propagation network and augmented neural network. In next section, we will describe in more detail the types of neural network approach, specifically applied in solving the scheduling problem, and their research evolution.

3.7.5 Other Approaches

Other approaches that have tackled and successfully solved the mentioned cyclic scheduling classes include branch and bound. Various successful heuristics were also researched and applied. A summary of the approaches are shown in Table 3.3. This summary is based on: (a) types of cyclic scheduling problems and (b) types of approaches (on problems) over the years.

3.8 Neural Network Approach To Scheduling Problems

The artificial neural network (ANN) as it is commonly known is based on the human brain that has information processing abilities. The structure of the ANN is similar to information processing systems, comprising of largely interconnected processing units (or neurons) connected in a network. ANN mimics the human ability to learn through examples in order to solve problems. The artificial neural networks are categorized based on:

- Types of connections between neurons,
- The types of processing done by the neurons,
- The way the information is transmitted through the network,
- The method by which the network learns and the learning rate used.

Basic Cyclic Scheduling	
Branch and Bound	*Chretienne (1991)* [32] *Hanen and Munier (1995)* [65]
Basic Cyclic Scheduling with Linear Constraints	
Branch and Bound Tabu Search	*Munier (1996)* [124] *Brucker and Kampmeyer (2005)* [20]
Periodic Job Shop	
Branch and Bound	*Roundy (1992)* [136] *Draper et. al. (1999)* [39] *Hanen (1994)* [64] *Seo and Lee (2002)* [141] *Nakamura et. al. (2006)* [126]
Cyclic Flexible Manufacturing Systems	
Benchmarks Cyclic FMS	*Lee et. al. (2001)* [105] *Chaieb et. al. (2001)* [28] *Korbaa et. al. (2002)* [98] *Lee (2002)* [103] *Trouillet et. al. (2002)* [149] *Toguyeni and Korbaa (2005)* [148] *Lee and Korbaa (2006)* [104] *Korbaa and Camus (2006)* [95] *Trouillet et. al. (2007)* [150] *Hsu et. al. (2008)* [78]
Cyclic robotic production line	
	Kats and Levner (2002) [92] *Dawande et. al. (2005)* [38] *Gultekin et. al. (2006)* [60]
Cyclic scheduling Flow shop	
Stochastic Flowshop Blocking Flowshop Hybrid Flowshop	*Lee and Seo (1998)* [107] *Kamal Abadi et. al. (2000)* [1] *Munawar et. al. (2003)* [123]
Cyclic Open shop	
	Kubate and Nadolski (2005) [99]
Complexity and Classification on Cyclic Scheduling Problems	
Cyclic Production System Performance Tradeoff	*Middendorf and Timkovsky (2002)* [122] *Herrmann (2003)* [69] *Kampmeyer (2006)* [89] *Marchetti and Munier (2006)* [117]

Table 3.3: Researches into Cyclic Scheduling Problem

Generally the neurons in the network will transmit information by calculating and generating overall output values. This is done by combining an input function, activation function and output function on the net input of each neuron to generate an output value. The connections between neurons in the layers have associated numerical values or "weights". These adjustable weights holds the learned information when training is done to configure the correct output required.

The neurons in the network may be grouped as layers or unlayered. Layered neural networks consist of interconnection of input layers, hidden layers and output layers. The information being processed by the ANN are pass through the layers. However depending on whether the neural network is a feedforward or feedback (recurrent) type, the information may only pass through the network once. The latter allows data to be looped back from output to input layers giving "feedback".

The neural network architecture is known prior to any use, consisting of the structure of connections, the number of layers and the number of neurons in each layer. And also prior to training, the weights in the interconnection are initialized to random values or predefined values. These will then be adjusted accordingly until the network has learned the relationship between variables.

The learning process, is commonly known as training, is done through introducing an example of output to ANN. This involves iteratively changing the initial weight values of the interconnections then using the final weight values as the trained ANN for testing. *Supervised learning* of the network depends on the required sample input and desired output data. In the case of *backpropagated learning*, the error between output and desired values are looped back into the preceding layers to change the value of the weights, thus minimizing the error. Once the squared error values are minimized to acceptable values, the values of the weights are fixed and the network is ready for use, as it is now trained.

In the case of *unsupervised learning* (or self organizing learning), there are no desired outputs present. This means that the network will try to self organize input data and discover the correlations between input and output data. Normally learning rules of correlation (synaptic weights adjusted according to Hebb's learning rules) and competitive (output neurons compete until there is a winner) were implemented to achieve this.

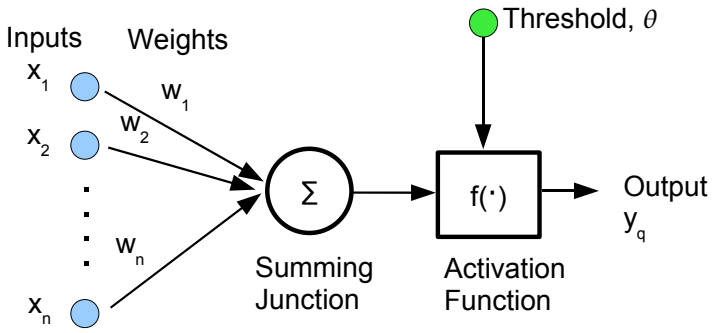


Figure 3.10: Nonlinear model of artificial neuron

The extension to supervised learning is the reinforcement learning where in this special case of supervised learning, the exact desired output is unknown but the outcome is based on whether or not the actual output is correct.

The most important part of the learning is the learning rules. These rules will determine how the weights are adjusted. Among the common learning rules are the Hebbian rule, anti-Hebbian rule and Competitive learning rule.

The basic model of an artificial neuron is shown in Figure 3.10. The output y_q of neuron q is calculated from the transfer (or activation) function

$$y_q = f\left(\sum_{j=1}^{n} w_j x_j - \theta\right) \tag{3.45}$$

with inputs x, weights w and threshold θ of that neuron.

The activation function [135] used in neural network consists of either the (i.) linear function, (ii.) hard limiter function, (iii.) symmetric hard limiter function, (iv.) signum function, (v.) saturating linear function, (vi.) symmetric saturating linear function, (vii.) binary sigmoid function or (viii.) hyperbolic tangent sigmoid function.

3.8.1 Feedforward Network

3.8.1.1 Single Layer Perceptron

Single layer feedforward perceptron generally comprises of 2-states and discrete time state neurons. These are the simplest neural networks. Associated with each input, x to the neuron is the synaptic weight, w. The

weights have both positive and negative values, influencing the associated inputs as exhibitory or inhibitory. The introduction of threshold θ will determine if the output of the neuron, y will fire (exhibitory state) when $y = 1$ or be quiet (inhibitory state),$y = 0$.This simple network has not been found to be successful in solving scheduling problems, due to its requirement for supervised training and limited optimization abilities.

3.8.1.2 Multi-layered Perceptron

Among the most common architecture of neural networks used in solving the scheduling problem is the multi layered perceptron (MLP) network. Neurons are organized into layers with unidirectional connections between layers. There normally exist a few hidden layers in the network where signals are propagated forward from the input layer to the output layer. This type of network may have full connections between neurons or be partially connected. This network is normally trained by the back-error propagation (BEP) (or back propagation) learning algorithm.

Back propagation provides the means for adjusting the weights in an MLP when presented with a set of training data. The gradient of the error function is calculated and these error values are then propagated backwards through the network layers. There will also be small changes made to the weights in each layer. The cycle is repeated until the overall error value is below some pre-determined threshold.

Figure 3.11 shows an example of a MLP network. The output of the network is defined by:

$$y_i^n = g_i(\sum_j W_{(ij)} g_j(\sum_j w_{(jk)} x_k^n)) \tag{3.46}$$

with weights in hidden layers,$W_{(ij)}$ efficiently updated using last steps error adjustments:

$$\Delta W_{(ij)}(t+1) = -\eta \frac{\partial E}{\partial W_{(ij)}} + \alpha \Delta W_{(ij)}(t) \tag{3.47}$$

where η is a learning constant. The measure of error is calculated be-

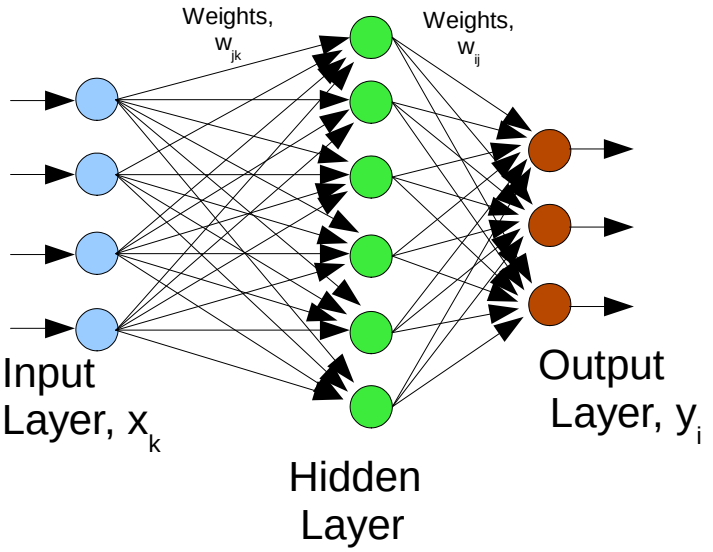


Figure 3.11: Multi Layer Neural Network

tween desired output d_i and output values:

$$E = \frac{1}{2}\sum_{in}(d_i^n - x_i^n)^2 \tag{3.48}$$

The Back propagation network generally requires a training set with a desired output values in order to determine the values of the weights. BPN generally are used efficiently as recall/generalization for solving scheduling problems.

3.8.2 Recurrent Neural Network

The recurrent neural network is a modification to the feedforward neural network with the network recurrently working on the internal states. This types of network will utilize feedback that will include the initial and past state, based on a serial processing nature. The common recurrent neural networks are the Hopfield networks, competitive networks, Self-Organising-Map (SOM) and Constraints Satisfaction Adaptive Neural Networks.

3.8.2.1 Hopfield Networks

The *Hopfield and Tank (1985)* [76] network has a single layer of neurons that feeds the output state back to inputs recurrently. As there is only one layer, all neurons function as inputs and outputs. There are weights associated between neurons to represent the strength between neurons, where large weight values dictate strong connection between neurons. The threshold/bias in the Hopfield network for each neuron controls the excitation of each neuron state. The Hopfield network has a discrete or a continuous version.

The discrete version of Hopfield Network is based on time step change in neuron output:

$$u^{(k+1)} = Tv^{(k)} + i^b \tag{3.49}$$

where the states of the neurons, u depends on output v, offset bias i^b while T containing the weights of the connections between neurons. The output function of the Hopfield network is commonly defined as:

$$g(u_i^{(k+1)}) = sign(u_i^{(k+1)}) \tag{3.50}$$

When neurons are updated stochastically (or asynchronously) and the connection matrix is symmetric (i.e. weights matrix of connections is symmetric), *Hopfield (1984)*[75] showed that Step version of Hopfield network has Lyapunov function that represents the gradient descent of the quadratic energy function of:

$$E = -\frac{1}{2}v^t Tv - (i^b)^t v \tag{3.51}$$

The continuous version of Hopfield Network is represented as follows:

$$\dot{u} = -\frac{u}{\tau} + Tv + i^b \tag{3.52}$$

with an output function:

$$g(u_i) = tanh(\frac{u_i}{u_0}) \tag{3.53}$$

and with Lyapunov Function:

$$E = -\frac{1}{2}v^t T v - (i^b)^t v + \frac{1}{\tau}\sum_i^N \int_0^{v_i} g^{-1}(x)dx \tag{3.54}$$

The recurrent characteristic of the Hopfield network has been successfully used to solve optimization problems as the network dynamics allow the network to achieve a stable state.

Back propagation networks generally require a training set in order to determine the values of the weights prior to being used.

3.8.2.2 Competitive Networks

This type of neural networks differs from Hopfield networks in the sense that although the input neurons are all connected to the output layer neurons, there exist inhibitory links between all output neurons in the same layer. As such, a competitive nature occurs among output neurons where the most active neuron will be the only neuron to remain active, while all the other neurons in the layer will slowly be deactivated. The competitive network is commonly comprosed of two neural networks, as shown in Figure 3.12:

1. *Hemming network* calculates a weighted sum of the input values,
2. *Maxnet neurons* in the competitive layer compete against each other by sending out inhibiting signals to each other. All neurons converge to zero except for the neurons with the maximum initial value. In this way the Maxnet network identifies the neuron with the maximum value.

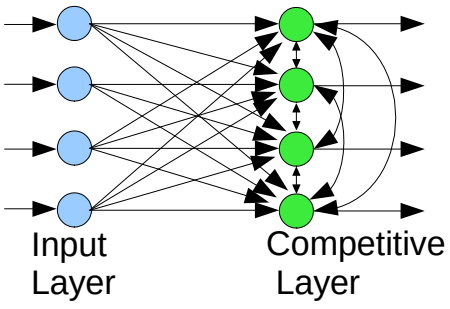


Figure 3.12: Example of Competitive Network

The *Winner-take-all rule* is applied to the neurons of a particular column x in the matrix:

$$s_{xy} = \begin{cases} 1 & \text{if } Net_{xy} = Max_{i=1} Net_{iy} \\ 0 & \text{otherwise} \end{cases}$$

where $Net_{xy} = \sum_{x}^{N} \sum_{y}^{(N)} W_{xI,yJ} s_{xy} - \vartheta_{xy}$ and the $Max_{i=1} Net_{iy}$ denotes the maximum total neuron output.

The Competitive Hopfield Neural Network (CHNN) [31] is a competitive version of the Hopfield network. This network is based on the algorithm:

Step 1. Randomly set the initial neurons states,

Step 2. Define the weight and threshold values according to Hopfield functions,

Step 3. Calculate the total neuron input, impose the winner-take-all rule to decide the output neuron state based on the initial value,

Step 4. Replace random initial state with output neuron states obtained from previous states.

Step 5. Repeat iteration using *Step 3* and *Step 4* until no change in state in any iteration is found,

One of the major advantages of competitive networks is that the competing nature of the output neurons allows for the unsupervised network to detect irregularities and compensate for corrections in the input vectors.

3.8.2.3 Self-Organising-Map (SOM) Networks

The Self-Organising-Map (SOM) Network [93]is different from other neural networks in terms of its ability to classify the inputs of the network. SOM achieved this by taking inputs in the form of vectors and classifying these inputs into different groups. This is done with the intention for each group to have some similarities to their input values.

A weight vector is initially defined for each classification group. The dynamic of SOM is based on an comparing each input vector to all weight vectors. So the weight vector representing the classification group most similar to the input vector currently considered, is classified as the winner. As such, the weights in this particular weight vector will be adjusted

accordingly. Neighbouring weight vectors adjacent to the winner also have their weights adjusted to be more similar to the input vector of interest, but to a lesser degree. This process is repeated until a convergence of the weight vectors are found.

An example of SOM used for solving a scheduling problem can be found in *McMullen (2001)* [120]. The SOM network in this case was developed for solving JIT production-sequencing problem with set-ups minimization and material usage stability. Although the experimental results were promising, the inferiority of SOM network developed was the large number of epochs and iterations chosen, necessary to ensure that weight convergence is obtained. This approach was competitive with the search heuristics such as simulated annealing, tabu search and genetic algorithms (GAs).

3.8.2.4 Constraints Satisfaction Adaptive Neural Network

Witkowski et. al. (2004) [155] used the Constraints Satisfaction Adaptive Neural Network (CSANN) to solve a production scheduling problem. This particular type of neural network was mapped from the constraints of a scheduling problem into its architecture and attempted to remove the violation of the mapped constraints in order to satisfy the constraints. The CSANN adaptively adjusts its connection weights and bias of neural network depends on the violations of the constraints present during processing.

Yang and Wang (2000) [158] proposed the CSANN which contains three kinds of neurons (or units): *ST*-units, *SC*-units and *RC*-units for the job shop scheduling problem. Each ST-unit represents the start time of an operation, while SC-units and RC-units represent whether the sequence constraints and resource constraints are respectively satisfied. As the ST-units are dependent on the weighted activations of the SC-units, RC-units and also the previous activation state of the ST-units, all these allow for feedback adjustments on the network. The two layers of the CSANN consist of ST-units while the second layer comprises of RC-units and SC-units representing the constraints. The dynamic of the network will adjust the weighs to reduce all the RC-units and SC-units to zero activations (i.e. no violations in constraints) hence converging to an optimal feasible solutions.

3.9 Evolution of Neural Network use in Scheduling Problems

Several researchers [121], [137], [79] have tried to review the use of neural networks for solving the scheduling problem. However, this may not specifically imply the use of neural network in the cyclic scheduling problems though. Many of this review done focused on the types of neural network used in either for production or manufacturing scheduling problems.

Here we list the characteristics of the artificial neural network (ANN) that have been proven effective in solving the scheduling problems:

- ANN is able to capture the complex relationship between input and output variables used to define the scheduling problem. This includes how these variables relate to performance measures and the operational policy of manufacturing systems. Using these and also the connections between the job characteristics and performance measures in the scheduling system, ANN is able to find the near-optimal solutions for the scheduling problems.
- In the case of a static scheduling environment, it is possible to obtain optimal or near optimal schedules through the use of mathematical modelling and dynamic programming.However ANN can allow for dynamic scheduling cases, thus eliminating rescheduling when changes in variables are presented.
- Although simulation softwares are widely used to simulate and some form of solution is deduced from the effort, some ANN may be used as an alternative to simulation software.
- Back-Propagation Network would be able to select the appropriate scheduling rules or manufacturing strategy in order to achieve accurate estimation of parameters in the scheduling problems. The parameters that can be affected include estimating system performance measures such as mean utilization, mean job tardiness, mean flow time, etc.
- Optimizing network, namely recurrent network i.e. Hopfield network and its extension are involved directly in the optimization of the scheduling problem. This is achieved by mapping both the

scheduling objective functions to be optimized and constraints of the problems on to these networks.

- The competitive neural networks can detect irregularities and correlations in input information and adapt the output responses accordingly.

The evolution of research in neural network applied on scheduling problems can be generally categorized according to the architecture of the network and type of scheduling problem being solved. Using this two main criteria, the networks can be categorized mainly into:

1. Hopfield network and its extension,
2. Competitive network, and
3. Back-propagation network.

3.9.1 Hopfield Network And Variations

Hopfield Network [76] is one of the most successful neural network approach in solving scheduling problems. This is especially due to the modelling of the scheduling problem that is defined by the quadratic form of the problem, with the energy function stabilizing at minimum point. The effective cost function also contributes to the Hopfield network success. Although the network performed gradient descent on energy function to update solutions, it can also easily be trapped in local minima state. However, the solutions to the scheduling problem can be determined from critical points determined by critical values of penalty and network parameters.

Since *Hopfield and Tank (1985)* [76] successfully applied the Hopfield network to solve the Traveling Salesman Problem (TSP) using the quadratic form of the optimization problem, this network has been extended to solve the job shop problem as found in *Foo et. al. (1988)* [47], [48]. *Foo and Takefuji (1988a)* [47] introduced a two dimensional Hopfield network to model the job shop. The $(mn + 1) \times (mn)$ matrix was mapped from the n jobs and m machines. Simulated annealing(SA) was however used to assist in optimizing the solution. These stochastic optimization techniques was selected to prevent the network from being trapped in the local minima, but able to arrive at near optimum solutions. *Foo and Takefuji (1988c)* [49] further improved from previous

work with integer linear programming neural network (ILPNN). Rather than use the quadratic version of the Hopfield energy function, a linear function was used. The number of neurons in the system was reduced to $\frac{(nm)(nm+1)}{2}$. Using the energy function in the form of:

$$E = \frac{A}{2} \sum_{x=1}^{mn} \sum_{i=1}^{mn+1} \sum_{j=1, j\neq i}^{mn+1} (v_{xi} v_{xj}) + \frac{B}{2} \left(\sum_{x=1}^{mn} \sum_{i=1}^{mn+1} v_{xi} - mn \right)^2 \tag{3.55}$$

with m machines, n number of jobs, v representing the output of the Hopfield Network and A,B as positive constants.

To eliminate the limitation associated with the quadratic function of Hopfield network, *Zhou et. al. (1991)* [162] utilized the linear cost function rather than the quadratic cost function in Hopfield. The linear cost function improved the scaling properties of the network and reduced need for integer linear programming methods that requires excessive control variable. This feature is also a simpler integer programming representation of problem with the number of neuron required equaling the number of operations hence the interconnection between neurons grows linearly with total number of operations.

Research into improving the convergence properties of the Hopfield includes modifying the energy function as described by *Aiyaer et. al. (1990)* [3]. Aiyaer et. al. investigated into the findings of valid subspace approaches relating to the performance of the Hopfield model. Effort was also done to improve the Hopfield convergence to valid solutions by alteration in penalty parameter and reducing search space through precalculations of the solutions in the job shop scheduling problem [154].

Although Hopfield network's output will converge to a solution, it may not guarantee good and feasible solution. As such researchers have added the stochasticity characteristics in Hopfield Network. *Arizono et. al. (1992)* [11] solved the single machine scheduling problem by incorporating the gaussian machine to minimize the total actual flow time. The stochastic characteristics used to avoid convergence to local minima included a noise-like zero-mean normal random variable. This is valued at $N(O, Temp^2/n)$, where *Temp* is a temperature parameter in the Gaussian machine model which allows for change in temperature of overall decrement in the energy function. This research was applied in Just In Time (JIT) scheduling and was successful for problem size of 50.

Huang and Chen (1999) [80] applied the combined Hopfield neural network and the normalized mean field annealing technique, applied to the n job m machines scheduling problem including resource and timing constraints. The mean field annealing (MFA) algorithm here was derived from Simulated Annealing by applying the mean field approximation technique. The MFA formulation is a stochastic neural network based on the Boltzmann state-transition rule, that included a spin interconnection and input parameter that replaced the weights and threshold of the Hopfield network. However the approach was not suitable for large problems.

Another extension of Hopfield network was proposed by *Satake et. al. (1994)* [140] to minimize the makespan of the job shop scheduling problems. Only one constraint is included in the energy function, while the threshold values represent the other constraints of the system. The threshold values are revised at each transition of neurons, and the Boltzmann machine effect was also integrated into the dynamics of the discrete Hopfield model with the simulated annealing methodology giving optimal or near optimal solutions.

Liansheng et. al. (2000)[112] introduced a unified Hopfield neural network algorithm that included a petri net model to represent active model, with an additional priority penalty term. Their modifiable goal and constraint function was used to solve different job shop schedule mode problems that included priority, dynamic scheduling and JIT scheduling.

Akyol et. al. (2005)[5] described a gradient based extension to Hopfield network for a multi machine scheduling problem. The six neural networks developed mainly modelled the penalties from the energy function with the objective of minimizing the weighted earliness and tardiness of the problem. The lack of simulation results included meant that the proposed approach was not fully evaluated.

Chen and Dong (1999) [30] used the Hopfield network as the optimization tool in solving the scheduling problems in a specific Surface Mount Technology (SMT) production problem. The problem aimed to minimize the total setup cost in producing different products. This was achieved through a nonlinear mixed integer programming model that was then converted into a continuous nonlinear programming formulation with added constraint equations. This model was then mapped into the Hopfield network and a sequential algorithm simulating the network was

developed to successfully solve a 4 lot example problem.

Wang et. al. (2003) [152] further improved the Hopfield network to solve the job shop scheduling problem by introducing a new energy calculation function for the Hopfield Network. This included all the constraints within the permutation matrix of $mn \times (mn + 1)$ where m jobs, n operations that represented the schedule. A simulated annealing feature was added to help the system stabilize and successfully avoid local minimum when tested on a 4 job, 3 machine job shop problem.

Maheswaran et. al. (2004) [116] used Hopfield network to solve the weighted total tardiness problem on a single machine. A binary representation of the schedule that relates to the probability of whether the operation start time is greater than the total tardiness factor TF was introduced. Their approach was tested on 10 job problems, and found to outperform the dispatching rules of earliest due date and weighted shortest processing times.

Akyol and Bayhan (2007) [6] proposed a dynamical gradient Hopfield neural network comprising of two maximum neural networks, of type piecewise linear and log-sigmoid network that interacted with each other. A time varying penalty coefficient is also included when applied to solving the non-identical multi machine scheduling problems, was able to minimize the sum of weighted earliness and tardiness of the problems.

3.9.2 Competitive Networks

Fang and Li (1990) [41] applied the competition neural network on single machine total tardiness problems. Limiting the single neuron activation per row and per column, the equation of motion was developed for the energy function that allows for convergence of the neural state.

Another researcher *Sabuncuoglu and Gurgun (1996)* [138] solved the single machine scheduling problem and job shop scheduling problem using the competitive characteristics. In the network, the neurons representing the jobs will compete with each other in order to be sequenced in the schedule. The performance of the competitive network was compared with the Wilkerson and Irwin (WI) algorithm, by comparing mean tardiness and the computation time. The proposed network produced better quality solutions than WI.

Chen and Huang (2001) [31] used the competitive Hopfield neural networks (CHNN) in multiprocessor scheduling problems. A 3-dimension Hopfield Network containing job, time and processors was developed and based on dateline and limited resource execution time constraints. The job shop problem was initially composed as a Hopfield Energy function prior to combining with a competitive learning mechanism. Through the experimental results, the CHNN was able to obtain feasible schedules from randomly initialized schedule and the rate of convergence is initial-state dependent, sometimes with a random distribution.

3.9.3 Back-propagation Networks

The back-propagation networks (BPN) consists of feedforward network or multilayer perceptrons comprising of sets of neurons connected by weight links. The dynamics of this type of network generally involved:

1. initializing the weights randomly before training,
2. applying the inputs and desired outputs,
3. comparing the calculated output against the desired output for error calculation,
4. the errors calculated are then propagated backward through network and weights adjusted by magnitude correlating to negative gradient of error function (normally equal to sum of squared error),
5. the steps are repeated to minimize the difference between actual and desired output

Successful application of BPN network in job shops scheduling includes *Chryssolouris et. al. (1991)* [34] who combined BPN and simulation to determine the number of machines and operational policy required for each work station in manufacturing systems. The performance measures used included the mean flow time and mean tardiness in job shop scheduling environment.

Feng et. al. (2003) [43] used BEP network to design, develop and implement a scheduling system, incorporating special data encoding for processing times and sequences with heuristic to revise initial output in a real job shop scheduling problem. Their research was able to solve a 6 job 5 machine problem.

The use of Back propagation network in the single machine scheduling problem was researched by *Sabuncuoglu et. al. (1996)* [138], for finding the relationship between problem data (e.g. processing times, due dates, etc.) and properties of optimal solutions (e.g. schedules). The approach managed to solve the common due dates, total tardiness, and flow time of the scheduling problems but the results greatly depended on the size of the problem.

Researchers also improved the effectiveness of the BPN by modifying the network, e.g. *Jain and Meeran (1998)* [84] modified the BPN for minimizing makespan in job shop problems. Although the modified BEP network optimized the job shop scheduling problem, the network has additional features to assist the search i.e. momentum parameters, jogging parameter and learning parameter to avoid local minima, eliminated the generalized learning capabilities to map the input and output for NP-hard problem. The limitation of this modified BEP network is that size of network still increases with the size of the problem but works more effectively than the three other dispatches rules (SPT, MWR, FCFS, shifting bottleneck procedure from *Adam et. al. (1988)* [2] and in less computational time.

Sabuncuoglu and Touhami (2002) [139] used BPN as a simulation metamodel in estimating manufacturing system performances of a job shop problem They however used training sets from simulation software e.g. ARENA, SIMAN, ProMODEL and deduced that the results were as good as from simulation software. The simulation metamodel was the subset of the actual simulation work possible on a system.

Another use of the BPN is for the design of manufacturing systems. *Cakar and Cil (2004)* [4] used the performance measures of mean flow time, mean tardiness, maximum completion time, machine utilization rate of each work center and percentage of late parts as inputs to the network. Their research were aimed at using the BPN to generate the required number of machines in each work center as the optimized result for that particular manufacturing system.

So from the review of work researched on back-error propagation network, the descriptions of this network includes:

- has good generalization to correlate relationship between input and output variables in scheduling problems.

- generally does not optimize except in the case of modification *Jain and Meeran (1998)* [84].
- can be used as a metamodel simulation model for design of manufacturing systems.

However the back-error propagation network falls short in terms of requiring large training sets and are commonly incorporated in heuristic or metaheuristic techniques. Also the BPN network risks being trapped in local minima when trained by gradient based search technique. The Back propagation networks solve the scheduling problems above based on a generalization phase, (except the study by *Jain and Meeran (1998)* [84]) and are not directly involved in the optimization problem.

3.10 Summary

In this chapter, we describe the modelling of the elements associated with the scheduling problems. In order to fully incorporate the real life conditions in manufacturing environment the problem formulation must be modelled correctly. All constraints should also be included. Without a full formulation of the scheduling problem, and constraints, the solving approach may be limited in terms of problem variants and its performance.

A description of the complexities associated with the different types of scheduling problem has also been given. Finally we summarized some of the approaches that are either approximation or iterative approaches that had been used for the scheduling problem.

We then introduced the parameters associated with cyclic job shop scheduling problem necessary for an accurate modelling of the scheduling problem. It was vital for us to define the constraints between operations and mathematically model these constraints.We later described our approach in modelling the cyclic job shop scheduling problem with the objective of achieving minimum cycle time of the problem. Hence we have prepared the modelling of the cyclic job shop in the linear programming form which will then be utilized to be solved by our later recurrent neural network approach.

We also described the approach in calculating the Work in Progress when modelling the cyclic Flexible Manufacturing System(FMS) based

on overlapping and non-overlapping conjunctive constraints. We also showed several examples from figures on how this approach can be utilized. Finally we presented an unconstrained optimization problem form of the cyclic FMS problem, by incorporating the objective of minimizing WIP subjected to described constraints.

Finally in this chapter we have described several popular neural network models. We also described the neural network's characteristics and some of the architecture required for solving optimization problems. The evolution of research of neural network in solving the scheduling problem mainly concentrated on the Hopfield Network, Competitive networks and multi-layered back propagated neural network. As found from the research done on neural network, Hopfield network and competitive network are capable of generating feasible and optimal schedules. This of course involved supervised learning using already known optimal schedule for selected scheduling problems. Multi-layered back propagated neural networks use more of its generalization characteristics. This network is mainly used to select from scheduling policies or predefined rules. The neural network approach depends on the availability of ready optimal solutions in the form of schedules for particular problems in some cases. Although neural network has been proven successful in solving scheduling problem, the lack of emphasis on cyclic scheduling problems has prompted this research.

Part II

Approach

CHAPTER

FOUR

Approaches in Solving the Cyclic Scheduling Problems

In this chapter, we present the Recurrent Neural Network incorporating combined special phases as a very viable approach in solving cyclic scheduling-related problems. We then extend this Recurrent Neural Network to include the Lagrange relaxation and Lagrange multipliers. We will also present the advanced Hopfield Network model to be used as a viable approach to solve the cyclic FMS problem.

4.1 Recurrent Neural Network Model and CJSSP

4.1.1 Recurrent Neural Network Model

This recurrent neural network model is an extension from the *Cichocki et. al. (1996)* [35]. Our approach will include the Competitive Dispatch Rule Phase (CDRP) and PostProcessing phase that will be shown to improve the results obtained. Both phases proposed are intended to allow for flexibilities when solving cyclic scheduling problems and also compliment the Recurrent Neural Network's solutions.

We will combine this Recurrent Neural Network approach with the modelling approached used in the cyclic job shop scheduling problems, in solving the cyclic job shop problem and achieving the optimal solutions.

4.1.1.1 Motivation

The Recurrent Neural Network was chosen as a viable approach in solving the cyclic scheduling problem, based on the following characteristics:

1. has the advantages of feedforward and feedback neural network,
2. ability to optimize a problem,
3. allows for unsupervised approach,
4. considers the previous state of the systems together with the initial state,
5. allows for infusion of additional functions into the activation function.

4.1.2 Recurrent Neural Network Dynamics and Architecture

The Recurrent Neural Network model [35] greatly depends on the mathematical modelling of the problem to be solved. In this case, in order to solve the cyclic job shop problem, we need to convert the mathematical modelling of the cyclic scheduling problem into the form of an equivalent Linear Programming (LP) problem. The linear programming form generally consists of the cost function declared, while subjected to a limited number of inequality constraints. Precise modelling of the problem represented by the linear programming form will then be mapped to the appropriate energy function to be solved using the Recurrent Neural Network approach.

Initially, the input layer of the RNN functions to take in input schedule. As this RNN is recursive in the sense that the decision variables (i.e. start times of the operations) are recursively updated or improved, the input layer functions as integrator layer for this previous state value. The calculations of the decision variables are streamed through the network, where some change is applied to the initial solutions. The penalty function in the network that encompasses the constraints will help to dictate the direction of change for the output values of the decision variables (i.e. start times).

The recurrent neural network is modelled from the energy function deduced from the cyclic scheduling problem itself. This energy function

encompasses the two other functions which are the cost function and penalty function. The combination of both functions helps the recurrent neural network to optimize the solutions to the cyclic scheduling problems, while considering all conditions to the problem. The cost function is generally the function that is required to be minimized or maximized. As such, from the model, the objective function of the cyclic scheduling problem is chosen as the cost function. Ignoring the effect of the penalty function for example, a cost function with high value will give a high level of energy associated with the RNN model. This in turn is good if the objective was to maximize the objective of the problem. However, a low level of energy in the RNN model from a low valued cost function, will mean that the RNN model is far from the optimal solution required to maximize the outcome.

In this recurrent neural network, the energy function with K as penalty factor, is described as follows:

$$E(\widetilde{S}) = C(\widetilde{S}) + KP(\widetilde{S}) \tag{4.1}$$

Considering that the cost function $C(\widetilde{S})$ and penalty function $P(\widetilde{S})$ are composed of a set of decision variables for the cyclic scheduling problem. The set $\widetilde{S} = \{S_{i;j} | i = 1, 2, \ldots, N_j | j = 1, 2, \ldots, N\}$.

So when applied in solving the cyclic job shop problem, parameter $C(\widetilde{S})$ is mapped from the objective function vector $\sum_{j=1}^{N} \sum_{i=1}^{N_j} (S_{i;j}^{k+1} - S_{i;j}^{k})$. The objective of this neural network model is to minimize the overall energy function. Hence, if ignoring the parameter $KP(\widetilde{S})$, from Equation 4.1, the minimum energy value, $E(\widetilde{S})_{min}$ is found from $C(\widetilde{S})_{min}$.

In the case of the choice of penalty function, this depends on the constraints associated with the cyclic scheduling problem. In the case of optimal solution associated with low valued energy state, a high valued penalty function is not acceptable. This is because a high value penalty function dictates large violations of the constraints, hence will result in infeasible solutions. The best case is if the penalty function value is equal to zero, that means that no constraints are violated.

The penalty function $P(\widetilde{S})$ is a combination of constraint equations in the system, if there are a total of N_c constraints to be modeled. The penalty

function can be expressed as follows:

$$P(\widetilde{S}) = \sum_{c}^{N_c} \Phi[r_c(S_{i;j})] \tag{4.2}$$

The expression $r_c(S_{i;j})$ represent each constraints equation, while function $\Phi[r_c(S_{i;j})]$ conserves the feasible form of these constraint. So the penalty function $P(\widetilde{S})$ will add a positive penalty value into the energy function if any of the constraints are violated.

$$\Phi[r_c(S_{i;j})] = \begin{cases} \frac{1}{2}(r_c(S_{i;j}))^2 & \text{if } r_c(S_{i;j}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

The above characteristics can be modeled by using a linear activation function to guarantee positive linear values for function $\Phi[r_c(S_{i;j})]$.

Again, mapping the constraint function from the constraints of the cyclic job shop problem modelled is equivalent to $P(\widetilde{S}) = \sum_{c}^{N_c} \Phi[r_c(S_{i;j})]$ where N_c is the total number of conjunctive and disjunctive constraints that exist in the problem.

We then use the steepest descent method to update the solution of decision variable with learning rate μ:

$$S_{i;j}^{k+1} = S_{i;j}^{k} - \mu \frac{\partial E(S_{i;j})}{\partial S_{i;j}} \tag{4.4}$$

The $\frac{\partial E(S_{i;j})}{\partial S_{i;j}}$ can be expressed from

$$\begin{aligned} \tfrac{\partial E(S_{i;j})}{\partial S_{i;j}} &= \tfrac{\partial}{\partial S_{i;j}} \left\{ C(\widetilde{S}) + K \textstyle\sum_{c}^{N_c} \Phi[r_c(S_{i;j})] \right\} \\ &= \tfrac{\partial C(\widetilde{S})}{\partial S_{i;j}} + K \textstyle\sum_{c}^{N_c} \tfrac{\partial \Phi[r_c(S_{i;j})]}{\partial S_{i;j}} \\ &= \widetilde{c}_{i;j} + K \textstyle\sum_{c}^{N_c} \tfrac{\partial \Phi[r_c(S_{i;j})]}{\partial r_c(S_{i;j})} \tfrac{\partial r_c(S_{i;j})}{\partial S_{i;j}} \end{aligned} \tag{4.5}$$

where denoting function $\Psi[(S_i)] = \frac{\partial \Phi[r_c(S_{i;j})]}{\partial r_c(S_{i;j})}$ with $\Phi[r_c(S_{i;j})]$ defined from Equation 4.3:

Updating the solution of decision variable involves the use of the steepest descent method and learning rate μ. This update on the decision variable will utilize a dynamic learning rate from *Darken et. al. (1990)* [37],

defined by:

$$\mu = \frac{\mu_0}{log(1+K)} \tag{4.6}$$

here parameter K is set at a fixed value to allow for the model to reduce the learning rate at particular intervals. Generally the learning rate is set at a high value and is dynamically reduced.

The output layer that reveals a certain solution is then recursively applied back to the input layer. So in the next state, the improvement of the solutions depends on the change applied, considering the difference in correlated values between previous state input schedule and generated output schedule.

Network Output: The output of the network is obtained by:

$$S_j^{k+1} = S_j^k - \mu_j \left\{ \widetilde{c}_j + K \sum \Psi[(S_i)] \frac{\partial r_c(S_{i;j})}{\partial S_{i;j}} \right\} \tag{4.7}$$

The solution being updated with every iteration is subjected to the inclusion of a *schedule perturbation phase*. Prior to starting the Recurrent Neural Network a threshold value is defined, and with every iteration, the energy value is saved in a vector set defined by $(E^{(n-2)}, E^{(n-1)}, E^{(n)})$. By evaluating these values against the threshold value, the *schedule perturbation phase* will be activated based on their differences in value. Section 4.1.4 will describe this phase in more detail.

The continuous evaluation of the acceptance of the decision variable depends on the limitations set on the RNN model. Two distinct limitations are the maximum iterations allowable and maximum computational time allowed. These characteristics are introduced to limit the system from entering a continuous non-improving state if non-optimal solutions can be reached.

So if S_{output} is the generated schedule from the output layer, $Iter_{max}$ is the maximum iteration allowed and $CompTime_{max}$ is the maximum computational time allowed per execution. So if $Iter_{current} < Iter_{max}$ and $CompTime_{current} < CompTime_{max}$, then the S_{output} is recursived into the input layer. But if $Iter_{current}$ is equal to $Iter_{max}$ or $CompTime_{current}$is equal to $CompTime_{max}$, then the S_{output} is taken as a solution for the execution.

4.1.3 Recurrent Neural Network Parameters Selection

The selection of penalty factor K is generally chosen to be a large positive value. This factor is used to approximate the penalty function to the linear programming form of the mathematical modelling of the cyclic scheduling problem.

Learning Rate μ is another factor that affects the update on the decision variable. The good selection of value will help the recurrent neural network to reach the solution accurately. A low valued learning rate will slow down the seek rate of the network. From Equation 4.6, the learning rate is dynamically changing from large and fixed value. The μ will decrease at fixed interval K from the adaptive adjustments. Typically $\mu_0 > 0$ is set.

4.1.4 Schedule Perturbation Phase

In order to improve the solutions found as well as with the aim to improve the capability of the Recurrent Neural Network described above, we incorporate a *schedule perturbation phase* into the Recurrent Neural Network. This proposed perturbation phase is similar to the associated perturbation in the Simulated Annealing approach. However in our approach, the perturbation phase is only activated based on a specific condition. This condition is when the system is identified to be trapped in a local minima state. Trapped in local minima is defined as stabilizing into a state that may not necessarily be the global optimum state.

We introduce this addition to the Recurrent Neural Network model in solving the cyclic scheduling problems because the constrained problem contains local minimas that jeopardize the search for optimal solutions. This schedule perturbation phase functions explicitly as follow:

Phase 1 Assess for local minima reached. This is done by comparing the energy value attained from previous iterations. Setting a threshold value that constitute when this condition has been met, consequently activating the perturbation phase.

Phase 2 Generate a corresponding perturbed factor,

Phase 3 Unsettle the current state of the solution by incorporating the perturbed factor into solution,

Phase 4 Release new generated solution to the Recurrent Neural Network to be further iterated.

Phase 1 constitutes the use of a set containing previous solutions, and energy value. This set will contain three previous solutions with energy values $E^{(n-2)}, E^{(n-1)}, E^{(n)}$. When $E^{(n-1)} - E^{(n-2)} < Threshold \wedge E^{(n)} - E^{(n-1)} < Threshold$, the perturbation phase will be activated.

Utilizing the linear function $\Gamma(-\pi p_{i;j}, \pi p_{i;j})$, a perturbed factor for the processing time of the operation $O_{i;j}$ is generated. The π is a constant value with $p_{i;j}$ as processing time for a particular operation.

$$p^*_{i;j} = p_{i;j} + \Gamma(-\pi p_{i;j}, \pi p_{i;j}) \tag{4.8}$$

$$S^*_{i,j} = S_{i;j} + p^*_{i;j}(G(1 - \Omega)) \tag{4.9}$$

Here in the first stage of the perturbation algorithm the processing time is updated to a particular range. This range is a subset of the processing time range that is normally allocated to that cyclic scheduling problem. The chosen parameters π will not invoke a perturbed factor that may jeopardize the whole current schedule but merely spike the schedule to push the state out of the identified local minima.

A large perturbation may invoke or undo search steps done, thus rendering the optimization steps prior to perturbation unbeneficial. This large perturbation on the schedule may also increase the optimization effort after the new perturbed schedule has been generated.

Too small of a perturbed step however, may not be beneficial in releasing the system state out of the local minima. So a balance has to be found that will release the system from trapped local minima, and enough to allow for continuing effort in the optimization approach to seek out new optimal solutions.

The second stage of Equation 4.8 involved a particular decision variable updated with the perturbed factor. We introduce a stochastic characteristic into the application of the perturbed factor, using $G(1 - \Omega)$ as a stochastically affect to determine if the perturbed factor is incorporated or not. This is helped by the Ω parameter which is a random number in the range of $[0 : 1]$. The *schedule perturbation phase* is shown in Figure 4.1.

Input: List of Schedule S at iteration n, Set $E^{(n-2)}, E^{(n-1)}, E^{(n)}$, parameter π, parameter *Threshold*
Output: Schedule with perturbed start times

if $(E^{(n-1)} - E^{(n-2)} < Threshold) \wedge (E^{(n)} - E^{(n-1)} < Threshold)$ **then**
  foreach *operation o in List S* **do**
    Apply processing time, $p^*_{i;j} = p_{i;j} + \Gamma(-\pi p_{i;j}, \pi p_{i;j})$;
    Apply start times,$s^*_{i,j} = s_{i;j} + p^*_{i;j}(G(1 - \Omega))$;
    Update start time to List S ;
  end
end

Figure 4.1: Perturbation Schedule Algorithm

4.1.5 Competitive Dispatch Rule Phase (CDRP) Phase

Although the Recurrent Neural Network has been infused with the perturbation phase, to reduce the trapping of the system in local minima, we also introduce a preprocessing phase known as **Competitive Dispatch Rule Phase (CDRP)** prior to feeding the initial schedule into the RNN model. The purpose of this *CDRP* is to:

1. generate an initial schedule that is viable,
2. generate an initial schedule that has minimum constraints violations,
3. reduce the search space for the Recurrent Neural Network approach.

This preprocessing phase allows for selection of competitive dispatch rules applied on the initial schedule prior to being fed into the RNN model. This **Competitive Dispatch Rule Phase (CDRP)** comprises of a selection from:

Weighted shortest processing time (WSPT) The WSPT rule is one of the simplest and most effective in design. This rule is commonly used to minimize total completion time, mean flow time and percentage of tardy jobs [36]. The WSPT rule has been identified in the literature to perform very well in terms of minimizing the weighted mean flow time, as well as reducing the percentage of tardy jobs, especially under highly loaded conditions. Comparative studies have listed WSPT as one of the most consistent rules in solving job shop problems.

Input: List of Schedule S
Output: Schedule with WSPT Algorithm

Initialize $StartTime\ S$;=0 ;
Initialize $CompletionTime_{job=j}$; = 0;
foreach *job j from List S* **do**
    foreach *operation o in job j* **do**
        Find the operation o with smallest processing time, $p_{i;j}$;
        Place in assigned machine $M_{m=o}$ from List M ;
        $StartTime = \max\{CompletionTime_{job=j}, CompletionTime_{M_{m=o}}\}$;
        Update $CompletionTime_{job=j} = StarTime + p_{i;j}$;
        Update $CompletionTime_{M_{m=o}} = StarTime + p_{i;j}$;
        Assign operation to beginning of list S;
    end
end

Figure 4.2: Weighted Shortest Processing Time Algorithm

Weighted longest processing time (WLPT) As opposed to the WSPT rule, this rule will load the operations with the largest processing time from the jobs onto respective machines.

The two dispatching rules were selected based on getting the best options from the dispatching rules characteristic based on properties associated with the job or operation.

Competitive Dispatch Rule Phase (CDRP) works through a parallel single run of each dispatch rule. This will generate three viable schedules. The competitive characteristic based on the objective function of the cyclic scheduling problem will then select the best initial solution. This objective function may vary depending on the type of the cyclic scheduling problem being considered (i.e. minimum cycle time for CJSSP or minimum WIP for CFMSSP) to choose the best among the two schedule generated to be fed into the Recurrent Neural Network.

An example of the CDRP used for a cycle job shop is shown in Figure 4.4. It consists of 4 jobs. Applying the CDRP shows that schedule from WSPT rule in this case has cycle time of 23, thus being selected to be fed into the RNN model.

Alternatively to using the CDRP, we also employ the Randomized Schedule Generation Phase (RSGP). This schedule generation phase is based on allocating a start time to the individual operations based on a certain predefined range. The Algorithm 4.5 describes this phase.

Input: List of Schedule S
Output: Schedule with WLPT Algorithm

Initialize $StartTime\ S$;=0 ;
Initialize $CompletionTime_{job=j}$; = 0;
foreach *job j from List S* **do**
    foreach *operation o in job j* **do**
        Find the operation o with largest processing time, $p_{i;j}$;
        Place in assigned machine $M_{m=o}$ from List M ;
        $StartTime = \max\{CompletionTime_{job=j}, CompletionTime_{M_{m=o}}\}$;
        Update $CompletionTime_{job=j} = StarTime + p_{i;j}$;
        Update $CompletionTime_{M_{m=o}} = StarTime + p_{i;j}$;
        Assign operation to beginning of list S;
    end
end

Figure 4.3: Weighted Longest Processing Time Algorithm

4.1.6 Schedule Postprocessing Phase

This Postprocessing phase (PPP) is introduced to compliment the solutions obtained through the Recurrent Neural Network (RNN) model. Due to some limitations associated with the quality of solutions (that does not guarantee feasibility), it is important for the entire scheduling process to adapt through this proposed phase.

In addition to guaranteeing feasible solution, we introduce the Postprocessing Phase (PPP) that aims to achieve the following stages:

1. Checking and eliminating any minor violations associated with resource (or disjunctive) constraints. This is completed using the ***Adhere Disjunctive Algorithm***.
2. Checking and eliminating any minor violations associated with precedence (or conjunctive) constraints. This is achieved through the ***Adhere Conjunctive Algorithm***.
3. Transitioning any optimal solutions obtained through RNN, to the earliest start time state using ***Algorithm Compact Schedule***

The effect of introducing this **Postprocessing Phase (PPP)** is vital in ensuring the generation of assignment of operation start times that adhere to discrete time.

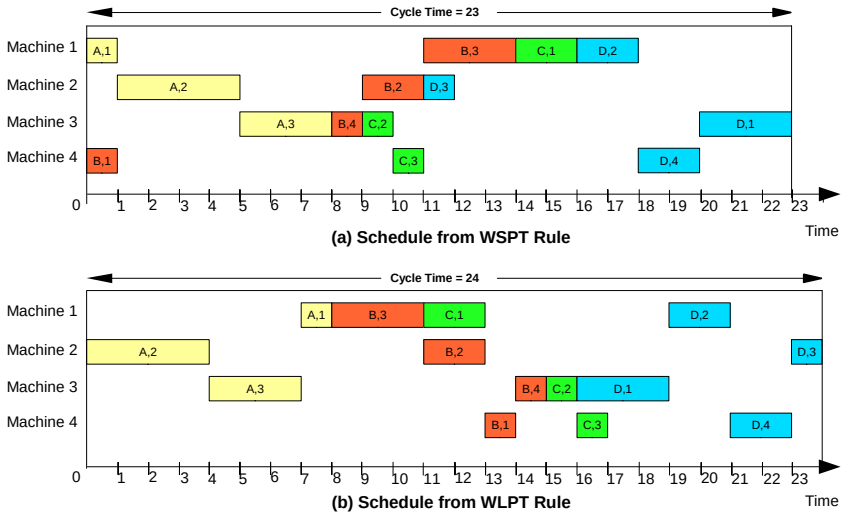


Figure 4.4: CDRP illustrated using (a) WSPT rule and (b)WLPT rule. Schedule from WSPT is chosen in this case.

Input: List of Operations, List of Jobs, List of Machines, List of Constraints
Output: Schedule of Start Times, $\widetilde{S}$

```
Initialization of processing range, Start Time Range ;
foreach job j from Job List do
    foreach operation o from Operation List do
        Get Random start time from Start Time Range ;
        Assign random start time S_o to operation o ;
        Update Schedule of Start Times, S̃ ;
    end
end
```

Figure 4.5: Randomized Schedule Generation Algorithm

4.1.7 The Evaluations

In order to evaluate the accuracy and optimization abilities of the Recurrent Neural Network with special phases, we tackle the three benchmark problems from *Fisher and Thompson (1963)* [45] denoted as FT06, FT10 and FT20. These three different problems have sizes as 6 jobs $\times$ 6 machines, processing time in interval $[1, 10]$, 10 jobs $\times$ 10 machines with Processing time in interval $[1, 99]$ and 20 jobs $\times$ 5 machines with processing times in interval $[1, 99]$. Table 4.1 shows the results found using the Recurrent Neural Network (RNN). The results clearly show that the RNN is effective to be used to solve the three problem with best optimal results obtained. The optimal schedule for large problems with 100 oper-

Input: Start Times
Output: Schedule adhering to Disjunctive Constraints

for *Machine m:=1* **to** M **do**
  for $OperationOnMachine_m\ O(m) = 1$ **to** $N(m-1)$ **do**
    if $StartTime_{O(m)+1} - (StartTime_{O(m)} + p_{O(m)}) < 0$ **then**
      Δ:=$StartTime_{O(m)+1} - (StartTime_O(m) + p_O(m))$;
      for $OperationOnMachine_m\ O(m) = 1$ **to** $N(m)$ **do**
        $StartTime_O(m+1)$:=$StartTime_O(m+1) + \Delta$;
      end
    end
  end
end

Figure 4.6: Adhere Disjunctive Algorithm

Input: Start Times
Output: Schedule adhering to Conjunctive Constraints

s^*:=s;
for *Job J:=1* **to** N **do**
  for $Operation\ O_j = 1$ **to** N_j **do**
    if $StartTime_{O(j)+1} - (StartTime_{O(m)} + p_{O(m)}) < 0$ **then**
      Δ^*:=$StartTime_{O(j)+1} - (StartTime_O(j) + p_O(j))$;
      for $Operation\ O_j = 1$ **to** N_j **do**
        $StartTime_O(j+1)$:=$StartTime_O(j+1) + \Delta^*$;
      end
    end
  end
end

Figure 4.7: Adhere Conjunctive Algorithm

Problem	N	M	O	$CT_{optimal}$	CT^{best}_{RNN}	$\%Dev$	$Iteration_{best}$	CPU(s)
FT06	6	6	36	55	55*	0	336	0.62
FT10	10	10	100	930	930*	0	18815	34.62
FT20	20	5	100	1165	1165*	0	8092	14.89

Table 4.1: Solution found for benchmarks FT06, FT10, FT20 [45] from RNN model

ations (i.e problems FT10 and FT20) were solved between 14 and 34s. To further simulate our RNN approach against other benchmarks for cyclic job shop scheduling problems, more experimentation results will be discussed in Chapter 5.

4.2 Lagrangian Relaxation Recurrent Neural Network and The CFMSSP

4.2.1 Lagrangian Relaxation Approach

As a mathematical technique used to solve constrained optimization problems, one of the earliest uses of Lagrangian relaxation in solving scheduling problems involved *Luh and Hoitomt (1993)* [114].

The Lagrangian relaxation technique (prior to addition to Recurrent neural network) will decompose each problem into the smaller subproblem whether at job-level or operation-level [46]. The relaxing of complicated coupling constraints e.g. machine capacity constraints using Lagrange multipliers, will allow for the smaller subproblems which are much easier to be solved, before solving the upper level harder problems. Once the subproblems have been solved, the Lagrange multipliers are adjusted at a higher level to satisfy the constraints of the harder problem. The additional cost of solving the subproblems, will only add minor additional computational time but will allow for better feasible solutions to be found.

The Lagrangian relaxation component in the recurrent neural network maximizes the corresponding dual of the cyclic scheduling problem. This dual problem is generated from associating the constraints with the Lagrange multipliers set and including these to the objective function. To solve this dual form of the problem, the Lagrange multipliers are selected and initialized allowing the problem to be separated. These multipliers are then minimized with respect to problem variables and updated. The second stage then will hold the problem variables as constant and dual function maximized with respect to dual variables. The maximization phase will help to satisfy the constraints associated with the problem. The process is repeated to conclude when a maximum dual and minimum decision variables are found. Normally the Lagrangian multipliers

are updated using subgradient technique.

The limitation of introducing Lagrangian relaxation into the recurrent neural network is the possible generation of infeasible near optimal or optimal solutions. Hence the use of our proposed **Postprocessing Phase** described in ***Section*** 4.1.6 will be helpful.

Early Lagrange relaxation work came from researchers *Fisher (2004)* [46], *Hoogereen and Van De Velde (1995)* [74]. Several researchers like *Luh et. al. (2000)* [115], *Fang et. al. (2000)* [42] have successfully solved the general job shop scheduling problem while *Zhang et. al. (2000)* [161], *Tang et. al. (2006)* [146] have applied this Lagrangian Relaxation approach to mixed model assembly line and flow shop scheduling quite successfully.

In this section, we introduce the combined Lagrangian relaxation approach into our Recurrent neural Network known as Lagrangian Relaxation Recurrent Neural Network (LRRNN). We will describe this approach in solving the cyclic flexible manufacturing system scheduling problem specifically based on the modelling technique specifically for cyclic FMS.

4.2.2 Lagrangian Relaxation Recurrent Neural Network Dynamics and Architecture

The Lagrangian Relaxation Recurrent Neural Network (LRRNN) model utilizes the addition of Lagrange relaxation and Lagrange multipliers into the energy equation of the Recurrent neural network. Here we will redefine the Lagrangian function and its corresponding constraints. The Lagrange multipliers function as scaling factors in the function. The general Lagrangian function is defined by:

$$L(\widetilde{S}, \lambda) = C(\widetilde{S}) + \sum_{c}^{N_c} \lambda_c r_c(S_{i;j}) \tag{4.10}$$

This Lagrangian function is the combination of the objective function, $C(\widetilde{S})$ to be minimized/maximized and the appended constraints denoted in penalty function $\sum_{c}^{N_c} \lambda_c r_c(S_{i;j})$ with Lagrange multipliers λ_c.

Since this LRRNN approach is considered to solve the CFMSSP, the objective function and constraints here relate to the linear programming form of Equation 3.40 and 3.41:

$$\min_{\{S_{i,j}\forall N,N_i\}} f(S) \equiv WIP = \sum_{i=1}^{N}\sum_{j=1}^{N_i}\left(C_{i,j} + \omega_{i,j}\tilde{C}_{i,j}\right)$$

Subject to:

$$\begin{aligned}
(\delta_{ijhl})(S_{h,l} + p_{h,l} &\leq S_{i,j}) \\
(1-\delta_{ijhl})(S_{i,j} + p_{i,j} &\leq S_{h,l}) \\
(\delta_{ijhl})\omega_{i,j}(S_{i,j} + p_{i,j} - C_T &\leq S_{h,l}) \\
(1-\delta_{ijhl})\omega_{h,l}(S_{h,l} + p_{h,l} - C_T &\leq S_{i,j}) \\
0 \leq S_{i,j} &< C_T
\end{aligned}$$

$$i = 1, 2, \cdots, N, \quad j = 1, 2, \cdots, N_i$$

The constraints $r_c(S_{i;j})$ that are considered in scheduling problem include the precedence constraints and disjunctive constraints that will be relaxed using the nonnegative Lagrange multipliers λ_c. Somewhat different from the Recurrent Neural Network approach where the penalty factor is fixed, here the Lagrange multipliers λ_c will change allowing for a dynamic Recurrent Neural Network.

When we consider the above Equation 4.10 to solve the CFMSSP, we have the objective function $C(\widetilde{S}) = \sum_{i=1}^{N}\sum_{j=1}^{N_i}\left(C_{i,j} + \omega_{i,j}\tilde{C}_{i,j}\right)$with a slightly modified penalty function, as follows:

$$L(S_n, \lambda) = C(\widetilde{S}) + \sum_{i=c}^{N_c}\lambda_c\varphi[r_c(S_n)] \tag{4.11}$$

where penalty function is defined as

$$\varphi[r_i(S_n)]\begin{cases} = 0 & \text{if} \quad r_i(S_n) \leq 0 \\ > 0 & \text{if} \quad r_i(S_n) > 0 \end{cases} \tag{4.12}$$

$$and \quad S_n \geq 0; \forall n \geq 0 \tag{4.13}$$

In the above, there will a total of N decision variables in the form of start times for all the operations from all the jobs and N_c unknowns constraints. For Lagrangian in Equation 4.11, the stationary conditions can

be found from:

$$\begin{aligned}\frac{\partial L(S_n,\lambda)}{\partial S_n} &= \frac{\partial}{\partial S_{i;j}}\left\{C(\widetilde{S}) + \sum_c^{N_c} \lambda_c r_c(S_n)\right\} \\ &= \frac{\partial C(\widetilde{S})}{\partial S_n} + \sum_c^{N_c} \lambda_c \frac{\partial r_c(S_n)}{\partial S_n} \\ &= 0\end{aligned} \tag{4.14}$$

$$\begin{aligned}\frac{\partial L(S_n,\lambda)}{\partial \lambda_c} &= \frac{\partial}{\partial \lambda_c}\left\{C(\widetilde{S}) + \sum_c^{N_c} \lambda_c r_c(S_n)\right\} \\ &= \frac{\partial r_c(S_n)}{\partial \lambda_c} \\ &= 0\end{aligned} \tag{4.15}$$

Both Equations (4.18) and (4.19) will give a total of $(N + N_c)$ equations to be solved.

The optimal solution $\widetilde{S^*}$ will give the solution with the minimum WIP value when ignoring the constraints.

4.2.2.1 Lagrangian Relaxation Recurrent Neural Network Dynamics

Updating the solution of decision variable and Lagrange multipliers involve the use of steepest descent method as in recurrent neural network.

$$S_n^{k+1} = S_n^k - \mu \frac{\partial L(\widetilde{S}, \lambda)}{\partial S_n} \tag{4.16}$$

$$\lambda_c^{k+1} = \lambda_c^k + \mu_\lambda \frac{\partial L(\widetilde{S}, \lambda)}{\partial \lambda_c} \tag{4.17}$$

$$\begin{aligned}\frac{\partial L(S_n,\lambda)}{\partial S_n} &= \frac{\partial}{\partial S_n}\left\{C(\widetilde{S}) + \sum_c^{N_c} \lambda_c r_c(S_n)\right\} \\ &= \frac{\partial C(\widetilde{S})}{\partial S_{i;j}} + \sum_c^{N_c} \lambda_c \frac{\partial r_c(S_n)}{\partial S_n}\end{aligned} \tag{4.18}$$

$$\begin{aligned}\frac{\partial L(S_n,\lambda)}{\partial \lambda_c} &= \frac{\partial}{\partial \lambda_c}\left\{C(\widetilde{S}) + \sum_c^{N_c} \lambda_c r_c(S_n)\right\} \\ &= \frac{\partial r_c(S_n)}{\partial \lambda_c}\end{aligned} \tag{4.19}$$

here both learning rates, $\mu > 0$ and learning rates for the Lagrange multi-

pliers, $\mu_\lambda > 0$ will assist in accurate convergence to an optimal solution.

The solutions generated at every iteration are generally infeasible. This is due to the continuous Lagrange multipliers used and relaxed constraints. As such, we use the *AdhereDisjunctive Algorithm* to modify, if required, the sequence found into a feasible one.

So from the modelling of the cyclic scheduling problem, the inclusion of Lagrangian relaxation helps in solving the problem more efficiently through solving the smaller operation level subproblems first before tackling larger job-level problems. This will help the recurrent neural network in considering less constraints. We consider that each cyclic scheduling problem compose of resource/disjunctive constraints and precedence constraints. Here we use Lagrange multipliers to relax the resource constraints for each machine, while the precedence constraints can be relaxed, using another set of distinct multipliers.

From the Lagrange Relaxation of the Recurrent Neural Network, the Lagrange multipliers are iteratively adjusted and the subproblems are iteratively solved. This iterative process continues until the model converges to an optimal solution, with minimum or zero violation in penalty function. In a case when an infeasible solution is found, the **PostProcessing Phase** will generate a feasible solution.

4.2.3 Modified Competitive Dispatch Rule Phase (MCDRP)

Prior to applying the LRRNN approach on the cyclic flexible manufacturing systems scheduling problem, we employ the **Modified Competitive Dispatch Rule Phase (MCDRP)**. Although this MCDRP is similar to the CDRP used with RNN, but it is specific for the cyclic flexible manufacturing systems based on the minimum cycle time constraint. The **MCDRP** will generate and evaluate two schedules prior to feeding the initial schedule into the LRRNN model. The main purpose of this *MCDRP* is to reduce the search space for the LRRNN approach but also to present an initial schedule with minimum constraints violations.

Although similar to CDRP in cyclic job shop scheduling problems, this phase differs in considering the operations based on each machine with no slack time to appear between operations. This **Modified Competitive Dispatch Rule Phase (MCDRP)** still comprises of a selection from WSPT

(Weighted shortest processing time) and WLPT (Weighted longest processing time) rules. A further description can be found from Figure 4.9 and 4.8.

Modified Competitive Dispatch Rule Phase (CDRP) runs a parallel single run of each dispatch rule and will generate two viable schedules. The schedule with the minimum objective function will be chosen to be fed into the Lagrangian Relaxation Recurrent Neural Network.

Input: List of Schedule S
Output: Schedule with WSPT Algorithm
1 Initialize $StartTime\ S$;=0 ;
2 Initialize $CompletionTime_{M_{m=o}}; = 0$;
3 **foreach** *machine m from M* **do**
4 **foreach** *operation o in job j on machine m* **do**
5 Find the operation o with shortest processing time, $p_{i;j}$;
6 $StartTime = CompletionTime_{M_{m=o}}$;
7 Update $CompletionTime_{M_{m=o}} = StartTime + p_{i;j}$;
8 Assign operation to beginning of list S;
9 **end**
10 **end**

Figure 4.8: Weighted Shortest Processing Time Algorithmfor CFMS.

Input: List of Schedule S
Output: Schedule with WLPT Algorithm
1 Initialize $StartTime\ S$;=0 ;
2 Initialize $CompletionTime_{M_{m=o}}; = 0$;
3 **foreach** *machine m from M* **do**
4 **foreach** *operation o in job j on machine m* **do**
5 Find the operation o with largest processing time, $p_{i;j}$;
6 $StartTime = CompletionTime_{M_{m=o}}$;
7 Update $CompletionTime_{M_{m=o}} = StartTime + p_{i;j}$;
8 Assign operation to beginning of list S;
9 **end**
10 **end**

Figure 4.9: Weighted Longest Processing Time Algorithm for CFMS.

Figure 4.10 shows an example of applying the MCDRP where 4 jobs exist in the CFMS. MCDRP shows that the WSPT rule will generate a schedule with cycle time of 8 and WIP of 10, while WLPT has WIP of 10 also, hence any of the schedules can be selected.

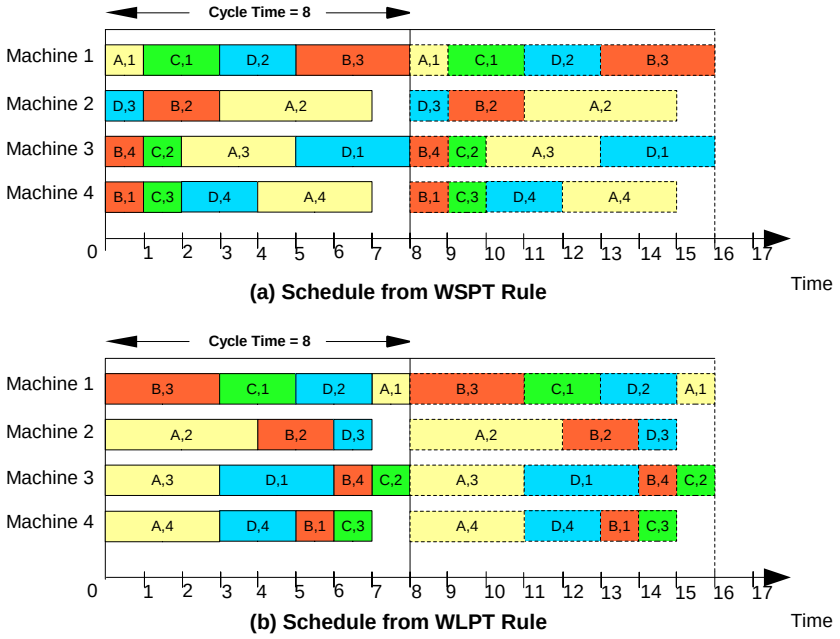


Figure 4.10: MCDRP illustrated using (a) WSPT rule and (b)WLPT rule.

4.2.4 The Evaluations

The Lagrangian Relaxation Neural Network with special phases is used to solved the Cyclic flexible manufacturing systems (FMS) problem. We chose the problem from *Hillion et. al. (1987)* [70]. The FMS in this problem consisted of 4 machines with 4 jobs. Each job consists of 4,4,3,4 operations respectively. The objective in this problem is to solve the CFMS problem to obtain the minimum Work in Progress (WIP) possible based on the minimal cycle time constraints. In this case *Machine 1* and *Machine 3* are bottleneck machines hence the schedule has cycle time of 8. The MCDR phase has generated two schedules as shown in Figure 4.10. Using initial Lagrange multipliers values of 0.50, learning rate of 100.00 and learning rate for Lagrange Multipliers of 100.00, with maximum iteration set as 10000, an optimal solution was obtained, reducing the initial schedule to the optimal schedule with minimum WIP of 5. Table 4.2 shows the best results in solving the HIL87 problems obtained in 31.8 seconds, compared to the genetic algorithm by *Hsu et. al. (2007)* [78], while Figure 4.11 shows the optimal schedule with the minimum WIP of 5.

Although our LRRNN approach was only simulated for HIL87 problem only in this section, further experimental simulations conducted on a number of other cyclic flexible manufacturing scheduling problems will

FMS Benchmarks										
Problem	N	M	O	CT	$W.I.P._{LB}$	GA WIP_{best}	GA CPU_{best}	LRRNN WIP_{best}	LRRNN $Iteration_{Best}$	LRRNN CPU_{best}
HIL87	4	4	15	8	5	5	~1	5	9654	31.8

Table 4.2: Best Result for FMS test problems HIL87 by *LRRNN*.

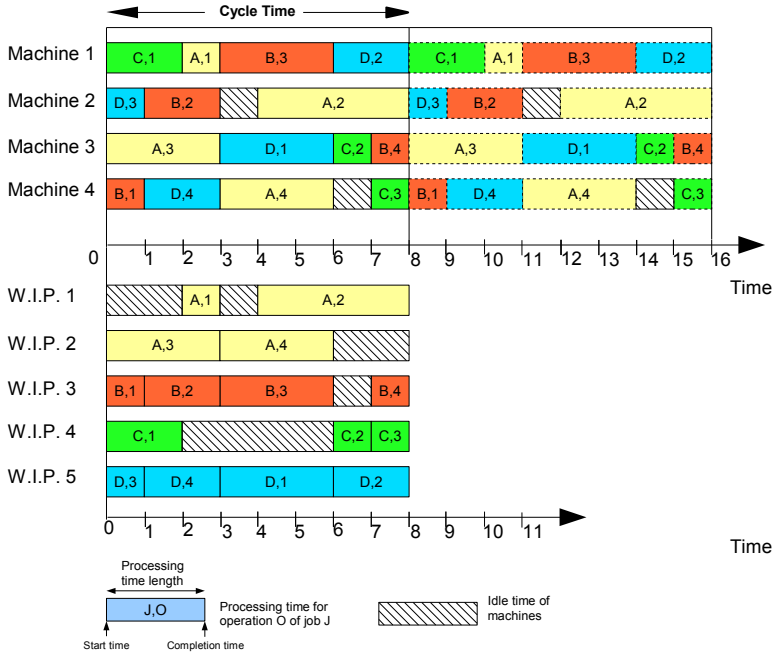


Figure 4.11: Optimal solution for CFMS problem HIL87 [70]

be described in Chapter 5.

4.3 Solving The CJSSP With Linear Constraints

The *Basic Cyclic Scheduling problem with Linear Precedence Constraints (BCSL)* has been researched by *Munier (1996)* [124]. This only covers the case of cyclic scheduling problems without any resource constraints. From modeling the problem as a linear graph, Munier deduced the expansion of the graph into equivalent uniform graph. This expansion graph would preserve the same number of constraints as the linear constraints. n_i and n_j duplicate generic operations are created from operations i and j, based on the conditions:

$$\frac{n_i}{\beta} = \frac{n_j}{\beta'} = s \tag{4.20}$$

where s, as the equivalent uniform constraints expanded from linear constraints (i, j).

As already briefly described in ***Section*** 2.6.2, the *basic cyclic scheduling (BCL)* problem with linear precedence constraints differs in the form of linear form of precedence constraints. The uniform precedence constraints would be the special case of linear precedence constraints.

The linear precedence constraint that govern the generic tasks i and i' can be defined as follows from *Munier (1996)* [124]:

$$S_i^{(\beta_{i,i'}n+\delta_{i,i'})} + p_i \leq S_{i'}^{(\beta'_{i,i'}n+\delta'_{i,i'})} \quad \forall n > 0 \tag{4.21}$$

The $\beta_{i,i'}$ and $\beta'_{i,i'}$ are two positive natural numbers while $\delta_{i,i'}$ and $\delta'_{i,i'}$ are just two natural numbers. If the above Equation 4.21 holds true and linear precedence constraints between operations i and i' are denoted by parameters $(p_i, \beta, \delta, \beta', \delta')$, this means that $\beta n + \delta$ occurrence of operation i must be completed before occurrence $\beta' n + \delta'$ of operation i'.

This linear relationship can be represented graphically by a directed graph $G = (T, E)$ with nodes T and arcs E as in Figure 4.12. The nodes represent the operations while the arcs represents the linear constraints. An example from Figure 4.13 shows the linear precedence constraints effect on the cyclic occurrences of both operations i and i'. Here the linear precedence constraints denoted by $(2, 3, 1, 2, 0)$ or $S_i^{(3n+1)} + 2 \leq S_{i'}^{(2n)} \quad \forall n > 0$.

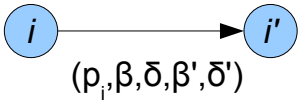


Figure 4.12: Linear Precedence Constraint.

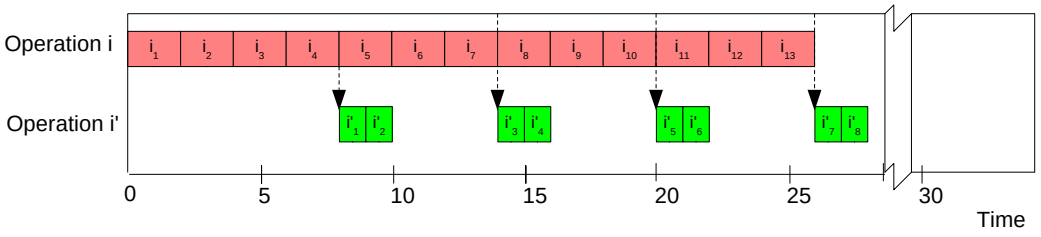


Figure 4.13: Gantt Chart for Linear Precedence Constraints (2,3,1,2,0)

Hanen and Munier (2008) [67] proposed the reduction of the BCS with linear precedence constraints into BCL with uniform precedence constraints. Hence according to *Munier (1996)* [124], some linear precedence cyclic scheduling problems (with objective of minimizing the cycle time) can be reduced into uniform scheduling problems, especially from the form of:

$$Minimise \quad \tau \quad s.t. \tag{4.22}$$

$$S_i^{(\alpha_{i,i'}n+\beta_{i,i'})} + p_i \leq S_{i'}^{(\gamma_{i,i'}n+\delta_{i,i'})} \quad \forall n > 0 \tag{4.23}$$

into the equivalent form of (in the directed graph):

$$Minimise \quad \tau \quad s.t. \tag{4.24}$$

$$S_{i'} - S_i \geq L_{i,i'} - \tau H_{i,i'} \quad \forall (i,j) \in E \tag{4.25}$$

provided the graph G is strongly connected. This means that for all cycles c of graph G, the combined weight of the arcs $\prod \frac{\gamma}{\alpha} = 1$. This is assuming that the weight for each arc (i, i') is $\frac{\gamma_{i,i'}}{\alpha_{i,i'}}$.

From [67], it is proven that the periodic schedule holds true for linear precedence constraints for all arcs $(i, i') \in E$ if:

$$S_i^0 + (\alpha_{i,i'}k + \beta_{i,i'})\tau_i + L_{i,i'} \leq S_{i'}^0 + (\gamma_{i,i'}k + \delta_{i,i'})\tau_i' \tag{4.26}$$

where S_i^0 is the start time for operation i, $S_{i'}^0$ is the start time for operation i', τ_i' and τ_i, are two positive numbers while $L_{i,i'}$ represents the length of the arc or processing time of operation i.

Several researchers that had investigated the linear precedence constraints in cyclic scheduling problem are *Munier (1996)* [124], *Hanen and Munier (2008)* [67] and *Cavory et. al. (2005)* [27].

4.3.1 Cyclic Job Shop with Linear Constraints

The Cyclic Job Shop Scheduling problem is the extension to the BCS with linear precedence constraints. This problem have resource constraints where the processing of operation on each machine cannot overlap at any point in time. Here the operations and jobs are processed cyclically in order to fulfill the demand quantities. The linear constraints only apply to the precedence constraints.

Based on the linear precedence constraints between operations i and i', the ordering of the start times relating to both the operations can be denoted by:

$$S_i^{(\beta n+\delta)} + p_i \leq S_{i'}^{(\beta' n+\delta')} \quad \forall n > 0 \tag{4.27}$$

where both operation O_i and $O_{i'}$ are ordered through parameters $(\beta n + \delta)$ and $(\beta' n + \delta')$. So occurrence $(\beta' n + \delta')$ of Operation O_i' cannot start until occurrence $(\beta n + \delta)$ of Operation O_i has been completed, while all disjunctive constraints are met.

An extension to the basic cyclic scheduling problem, *Brucker and Kampmeyer (2005)* [20] formulated the cyclic scheduling problem with limited resources and with linear precedence constraints. Here the constraints of a machine only being able to process one operation per instant of time is added into the problem.

Whereas in the case of *Cavory et.al. (2005)* [27], petri-net modelling and approaches from *Hanen and Munier (2008)* [67] of reducing the graph from linear constraints to be unitary were combined to formulate the mixed integer linear program of the cyclic job shop with linear constraints.

When representing the cyclic job shop in the linear graph, the linear precedence constraints are represented by the arc with parameters $(p_i, \beta, \delta, \beta', \delta')$, between two operations that are represented by two nodes. The corresponding resource constraints can be shown as well as directed (once a schedule is determined) arcs between operations to be processed on the same machines.

Among the properties of the cyclic operations in the CJSSP with linear constraints are that the tasks cannot be interrupted once it has been placed on the machine to be processed (commonly known as non-preemptive characteristics). The operations are also non-reentrant where the operations are not repeated on the same machine at any point in time, after being processed on a particular machine. Also all information as regards to release time, machine required to process the operation and processing time required for each operation is known prior to starting the scheduling.

The main objective of solving this CJSSP with linear precedence constraints is to find a cyclic schedule that will not only adhere to all precedence and disjunctive constraints but also aim to have the minimum cy-

cle time.

Munier and Hanen (2008) [67] found that for an infinite existence of the cyclic schedule to exist, certain parameters must be satisfied. This is in relation to the linear parameters between operations for each job. For each job, the $\prod \frac{\beta}{\beta'}$ must be equal to 1.

Also in order to eliminate the possibility of deadlock in the system, the condition upon which an operation can start come into play. *Munier and Hanen (2008)* [67] deduced that the start for operation j from equation 4.27 for previous occurrence of $\beta' + \delta' - 1$ can be anytime > 0 and already been processed prior to the effectiveness of the Equation 4.27.

4.3.2 Modelling CJSSP with Linear Constraints

In order to accurately model the cyclic job shop problem, the objective function is set as to minimze the cycle time, τ. This is subjected to linear precedence constraints between operations, generally held by parameters $(p_i, \beta, \delta, \beta', \delta')$, with p_i as processing time of initial operation. And by considering the disjunctive constraints in the machines where only a single operation can be processed at any period of time, hence denoted by:

$$(S_i^k + p_i \leq S_j^l) \vee (S_j^l + p_j \leq S_i^k) \tag{4.28}$$

for when operations i, j ($i \neq j$)are to be processed on the same machine m. Combining all the above, the cyclic job shop problem can be depicted in the form of:

$$Minimise \quad \tau$$

Subject to:

$$\begin{aligned} S_i^{(\beta_{i,i'} n + \delta_{i,i'})} - S_{i'}^{(\beta'_{i,i'} n + \delta'_{i,i'})} + p_i &\leq 0 && \forall n > 0 \\ \delta_{ikjl}(S_i^k - S_j^l + p_i) &\leq 0 && S_i, S_j \in T; k, l \in Z \\ (1 - \delta_{ikjl})(S_j^l - S_i^k + p_j) &\leq 0 && S_i, S_j \in T; k, l \in Z \\ S_{i,j} &\geq 0 \end{aligned}$$

The constrained problem has been shown to be equivalent to a mixed integer linear program. *Brucker and Kampmeyer (2005)* [20] proofed this

equivalence based on the Extended Euclidean Algorithm. Hence using their results, our approach in solving this cyclic job shop with linear constraints involves the reduction of the linear constraints into their equivalent uniform constraints through using a delinearization technique. Preserving all disjunctive constraints, we then proceed to solve this equivalent cyclic job shop problem using the Lagrangian Relaxation recurrent Neural Network (LRRNN) approach.

4.3.2.1 Conditions for Linear Graph

For a linear graph to exist that can accurately depict a cyclic scheduling problem with linear precedence constraints, several conditions must be fulfilled. These are mainly:

1. No deadlock in graph during the cyclic processing of the operation. This is achieved through the first occurrence of the operation to be started at anytime [67].
2. To guarantee no overlap between executions of a particular generic task in a cyclic schedule.
3. Consistency of the graph that will allow for infinite schedule to be created. Munier and Hanen (1997) has proven this case where $\prod \frac{\beta}{\beta'} = 1$ can guarantee this for a particular circuit of linking linear constraints.
4. Enclosed circuit depicting a cyclic behaviour where the schedule generated is cyclic in nature.

4.3.3 Solving CJSSP with Linear Constraints

Phases in solving the CJSSP with linear precedence constraints involve:

Step 1 Assess the CJSSP with linear constraints

Step 2 Generate duplicate operations through delinearilization

Step 3 Generate associated precedence constraints and disjunctive constraints

Step 4 Combine all duplicate operations and constraints to form CJSSP with uniform precedence constraints

Step 5 Solve the CJSSP with Lagrangian Relaxation Recurrent Neural Network (LRRNN) approach.

4.3.3.1 Delinearization Algorithm

Before we can implement the delinearization algorithm, the conditions from the linear precedence constraints parameters of $(p_i, \beta, \delta, \beta', \delta')$ that will allow for delinerization can be found from *Munier (1996)* [124]. Here the equivalence of the uniform constraints to the linear constraints depends on the condition that must be respected as:

$$\frac{nuc_i}{\beta} = \frac{nuc_{i'}}{\beta'} = s \tag{4.29}$$

where nuc_i and nuc'_i are the number of uniform generic operation i and i' respectively to be generated.

In this Delinearization algorithm, we will determine the number of equivalent number of generic tasks in a linear constraint from operation i to i', with linear precedence constraint $(p_i, \beta, \delta, \beta', \delta')$. Here based on a job with N operation required to complete the job. The steps are as follows:

Step 1 Start at last precedence constraint of the particular job, e.g operation $O_{(n)}$ and $O_{(n-1)}$ Based on Equation 4.29, taking the number of uniform generic operation $O_{(n)}$ as $nuc_{O_{(n)}} = \beta'$, we obtain the number of uniform generic operation $O_{(n-1)}$ as $nuc_{O_{(n-1)}} = \beta \times \frac{nuc_{O_{(n)}}}{\beta'}$. Here $nuc_{O_{(n)}}$ is defaulted at 1 for a single occurrence of operation i', unless specified otherwise.

Step 2 Take previous linking constraint, between operation $O_{(n-1)}$ and $O_{(n-2)}$ where by considering $nuc_{O_{(n-1)}} = \beta'$ this time, we obtain the number of uniform generic operation $O_{(n-2)}$ as $nuc_{O_{(n-2)}} = \beta \times \frac{nuc_{O_{(n-1)}}}{\beta'}$. So the value of $nuc_{O_{(n-1)}}$ is found from the previous step.

Step 3 Repeat above step for the next previous linking constraint for the job until the last linear precedence constraint between operation $O_{(1)}$ and $O_{(2)}$ is found.

Here in each step of the delinearization algorithm, the equivalent nuc_i uniform precedence constraints are generated while each identical operation i generated will inherit the equivalent disjunctive constraints.

An example to show the above condition can be fulfilled, is shown in Figure 4.14. Here this particular job has 5 operations with linear constraints

between each pair of operations. Working backwards from operation $(G,4)$ to $(G,5)$ with linear constraints $(2,1,0,1,0)$ determined that one operation of $(G,4)$ is required for one operation $(G,5)$ to be started. Considering the previous linear constraint between operation $(G,3)$ to $(G,4)$, with linear constraint $(2,2,0,1,0)$ will show that two operations $(G,3)$ are to be completed before we can start on operation $(G,4)$. Then further considering linear constraint $(2,1,0,1,0)$ between operations $(G,2)$ to $(G,3)$ will require four operations $(G,2)$ to fulfill the two operation $(G,3)$. Now since we already require four operations of $(G,2)$, taking the last linear constraint between operation $(G,1)$ to $(G,2)$, we found that for two operations $(G,2)$, two operations $(G,1)$ must be completed. So we then require a total of four operation $(G,1)$. In Figure 4.14, the delinearized form of the graph will have uniform precedence constraints between the operations.

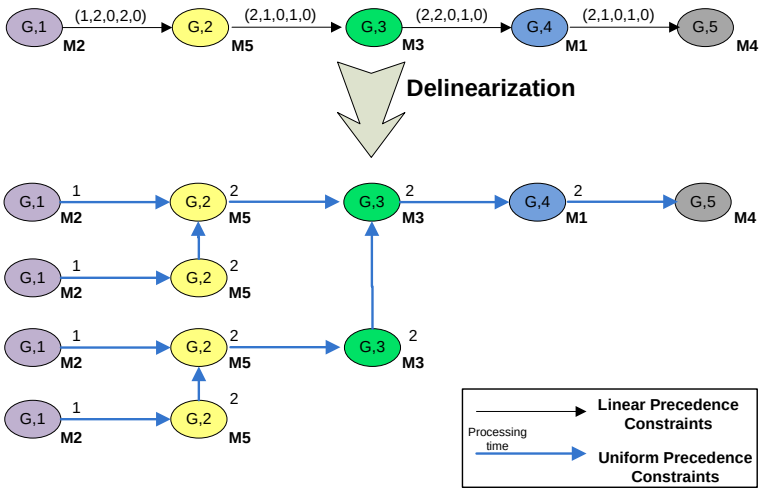


Figure 4.14: Delinearization Algorithm Done On Five Operations.

After the delinearization process is completed and the uniform precedence constraints and disjunctive constraints are determined and declared, next we will try to solve this cyclic job shop scheduling problem. Here the objective is to find the minimum cycle time for the cyclic scheduling problem. We then revert to our previously described Lagrangian Relaxation Recurrent Neural Network (LRRNN) approach to solve the completed uniform version of the Cyclic Job Shop Scheduling problem with linear constraints.

The already described Lagrangian Relaxation Recurrent Neural Network

(LRRNN) approach can be summarized by the following steps:

Step 1. Apply the delinerization algorithm on all the jobs in the cyclic job shop problem with linear precedence constraints between operations. This will result in an equivalent cyclic job shop with identical uniform operations, uniform precedence constraints and disjunctive constraints.

Step 2. Construct an energy function for the considered problem using a penalty function approach based on Equation 4.30.

$$L(\widetilde{S}, \lambda) = C(\widetilde{S}) + \sum_{c}^{N_c} \lambda_c r_c(S_{i;j}) \tag{4.30}$$

where penalty function defined as

$$\varphi[r_i(S_n)] \begin{cases} = 0 & \text{if} \quad r_i(S_n) \leq 0 \\ > 0 & \text{if} \quad r_i(S_n) > 0 \end{cases} \tag{4.31}$$

$$and \quad S_n \geq 0; \forall n \geq 0 \tag{4.32}$$

Step 3. Initialize schedules based on the *Competitive Dispatch Rule Phase (CDRP) Phase* from Section 4.1.5 and select the schedule based on a minimum cycle time.

Step 4. Select required initial Lagrange multipliers λ^0, learning rate μ^0 and learning rate for Lagrange multipliers μ_λ^0 and maximum iteration permitted.

Step 5. Start iteration of LRRNN by updating the Lagrange multipliers using Equation 4.34 and start times for operations using Equation 4.34:

$$S_n^{k+1} = S_n^k - \mu \frac{\partial L(\widetilde{S}, \lambda)}{\partial S_n} \tag{4.33}$$

$$\lambda_c^{k+1} = \lambda_c^k + \mu_\lambda \frac{\partial L(\widetilde{S}, \lambda)}{\partial \lambda_c} \tag{4.34}$$

Step 6. Check if perturbation phase described in Section 4.1.4 is activated with every iteration and activated if conditions are met.

Step 7. Repeat *Step 5* to *Step 6* until required criterion (i.e maximum iteration) is met.

Step 8. If the energy has stabilized to an optimal solution, employ the Post-Processing phase as described in Section 4.1.6 to ensure feasibility and optimality.

4.3.4 The Evaluations

In order to evaluate and validate our modelling and approach in this special cyclic scheduling case, we tested our approach on several problems. These problems were taken from *Dupas (2001)* [40]. The problems are detailed by m_n where m is the number of machines for the CJSSP while n is the jobs. The problems tested consisted of:

1. Problem 5_5: This is CJSSP with 5 jobs and 5 machines, with associated 5 generic operations per job.
2. Problem 5_10: This is CJSSP with 10 jobs and 5 machines, with associated 5 generic operations per job.
3. Problem 6_6: This is CJSSP with 6 jobs and 6 machines, with associated 6 generic operations per job.
4. Problem 6_12: This is CJSSP with 12 jobs and 6 machines, with associated 6 generic operations per job.

For all the above problems, we model the problems, apply the delinearization algorithm, and the corresponding uniform precedence and resource constraints. We then solve the problems using the Lagrangian Relaxation Recurrent Neural Network approach. The objective is to delinear the linear precedence constraints associated with the CJSSP and obtain cyclic schedule with the minimum cycle time for each problem of the cyclic scheduling problem. We consider the resource constraints in each problems as well.

The Problem 5_5 can be illustrated from Figures 4.15. The linear constraints between operations are denoted $(p_i, \beta, \delta, \beta', \delta')$ where p_i is the processing time of the operation. A circuit will denote a job comprising of operations with their corresponding precedence constraints. For Problem 5_5, the delinearization of the linear precedence constraints resulted in additional equivalent operations to satisfy the required constraints. For example, in job A, the application of the algorithm resulted in an extra 4 operations, in job B with an extra 11 operations, job C has an extra 2

Problem	Jobs	Machines	Operations	Lower Bound	Cycle Time GA [27]	Rand [27]	CSP [18]	COP [18]	LRNN	$Iteration_{Best}$	CPU(s)
Problem 5_5	5	5	25	15.90	26.10	26.5	26.4	25.4	26.00	912913	3005.1
Problem 5_10	10	5	50	19.73	62.44	63.4	N/A	N/A	65.00	1404480	4623.3
Problem 6_6	6	6	36	17.57	32.72	32.9	37.0	36.83	34.00	3245755	10684.4
Problem 6_12	12	6	72	20.29	69.9	70.2	N/A	N/A	72.00	4897424	16121.4

Table 4.3: Solution found for benchmark *Dupas (2001)* [40]

operations, job D has an extra 5 operations and job E has an extra 2 operations. A cyclic schedule is then to be generated that will solve and fulfill all the linear constraints. The Problem 5_10 is shown in Figures 4.16.

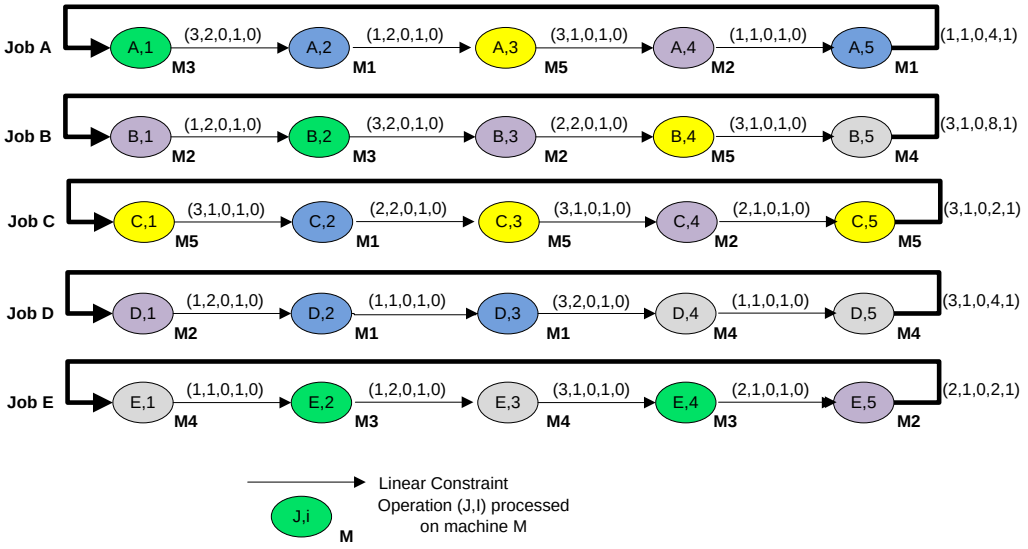


Figure 4.15: Problem 5_5 of CJSSP with Linear Constraints

From Table 4.3, we compare the results obtained from our approach to several previous techniques that included Genetic Algorithm [27], Constraint Satisfaction Problem (CSP) and Constraint Optimization Problem (COP) [18]. The lower bound [27] quoted is defined without considering any resource constraints in the cyclic job shop, and is not necessarily the most optimal solution. As there is no exact exact optimal cycle time quoted in *Dupas (2001)* [40] specified for problems, the results do show that our approach is comparable in terms of results obtained against other approaches used.

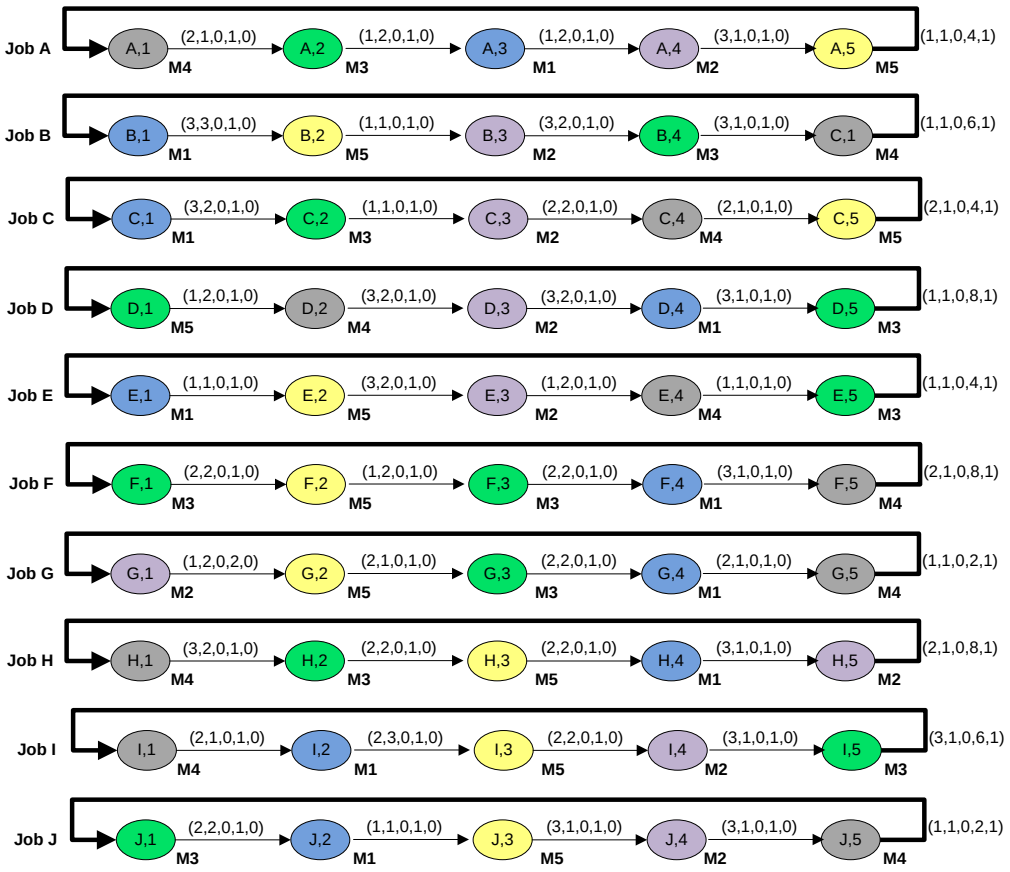


Figure 4.16: Problem 5_10 of CJSSP with Linear Constraints

4.4 Advanced Hopfield Network Model and CFMSSP

The recurrent nature of the Hopfield Network has many advantages in solving optimization problems, notably the Traveling Salesman Problem. *Hopfield and Tank (1985)* [76] successfully demonstrated these advantages. In the case of scheduling problems, *Foo and TakeFuji (1988)* [49, 47, 48] managed to solve the job shop problem with the Hopfield network. However *Cavalieri (1998)* [25] proposed solving the FMS problem with a modified Hopfield Network that encompasses the additional Petri Nets and varying bias current units. A summary of advancement of Hopfield neural network researched in solving the scheduling problem has already been described in Section 3.9.1 in Chapter 3.

In this section we examine and propose the Hopfield Network as a vi-

able approach to solving the cyclic FMS problem. This includes using the Hopfield Network characteristics of solving optimization problems and combining cyclic problem formulations, scheduling generating algorithms, and search space optimization into inner workings of the network.

Prior to applying the Hopfield Network to solving this constrained scheduling optimization problem, several considerations must be achieved. First and foremost the construction of the Hopfield network must accurately represent the problem. In other words, a corresponding relationship between neuron, neurons states and the parameters to be optimized, must be built. Secondly, an energy function must be built that truly reflects the objective function and the constraints and being able to reduce this function to the symmetric quadratic form. Thirdly, the weights that influence the connection and threshold associated with the activation of the neurons must be determined. Both parameters relate the Hopfield energy function to the particular energy function. Next is determining the dynamics which the weight coefficient in the penalty function and corresponding threshold for each neuron are to be updated. This is important to locate the optimal solution with every change in the change in neuron state in corresponding time steps.

Convergence of the energy function can be described from the dynamics of the energy function. The recurrence of the dynamic feedback along with the successive iterations of the change in the output state will bring the energy function to an equilibrium state [76]. Here the equilibrium state will equate to the optimum solution of the scheduling problem. This can also be classified by the state when all outputs eventually become constant, after successive iterations. As such, the Hopfield network is worked through by the following two main steps of storage phase and searching phase:

1. define the memory synaptic weights and threshold values;
2. according to values calculated in the neurons, with each iteration, the neuron state will be updated based on the initial value until no state change occurs at any iteration.

4.4.1 Advanced Hopfield Network Architecture

The architecture of Advanced Hopfield Neural Network comprises of a recurrent form of the neural network. The output states of the single layered neurons are fed back into the input of the network. Connections between neurons are based on the Hopfield's weights matrix which is symmetric. In each neuron, as the output values are applied, an external fixed bias is also included giving a new output value. The change in output and corresponding input is iteratively measured, where the dynamics of the neurons follow a gradient descendant of the quadratic energy function, known as the Lyapunov function until an equilibrium state is achieved.

We propose selecting the parallel dynamics of the network against the sequential (each neuron changed at each iteration) or stochastic dynamics (a neuron at random is chosen to change its state). The neurons in our case are updated simultaneously as this will reduce computational time in achieving the optimal solution.

The stabilized outputs of the Hopfield network will give the solution to the optimization problem. Generally the influencing conditions of the optimization are identified and will be expressed in the Hopfield energy function. These influencing conditions in the cyclic scheduling problem will include constraints and other parameters relevant to schedule.

4.4.2 Problem Formulation to be Solvable

Several assumptions to the system are vital to accurately describe the cyclic FMS problem. Most importantly, each job, j has a finite number of operations, i_j to be processed prior to completing the whole job. Each operation will be required to be processed on particular machines with limited machines available in the system. The processing time for each operation is finite, fixed and known prior to solving the cyclic FMS problem. The concept of work in progress and approach in calculating this parameter has been described in ***Section*** 3.6.1.5. A *Disjunctive set* exists, that holds sets of operations that are to be scheduled on a particular machine. Each pair of operations in this set will have disjunctive constraints.

The objective in solving this cyclic FMS problem is to find an optimum schedule. This optimum schedule will adhere to all constraints relat-

ing to resource relationships and can be turned into a cyclic schedule that will then determine a minimum work-in-progress. This optimized schedule will be obtained from the Hopfield network simulation results.

The formulation of the cyclic scheduling FMS problem can be summarized as follows from Section 3.6.1:

$$\min_{\{S_{i,j}\forall N,N_i\}} f(S) \equiv W.I.P. = \sum_{i=1}^{N}\sum_{j=1}^{N_i}\left(C_{i,j} + \omega_{i,j}\tilde{C}_{i,j}\right) \tag{4.35}$$

Subject to

$$\begin{aligned}
(\delta_{ijhl})(S_{h,l} + p_{h,l} &\leq S_{i,j}) \\
(1-\delta_{ijhl})(S_{i,j} + p_{i,j} &\leq S_{h,l}) \\
(\delta_{ijhl})\omega_{i,j}(S_{i,j} + p_{i,j} - C_T &\leq S_{h,l}) \\
(1-\delta_{ijhl})\omega_{h,l}(S_{h,l} + p_{h,l} - C_T &\leq S_{i,j}) \\
0 \leq S_{i,j} &< C_T
\end{aligned}$$

$$i = 1, 2, \cdots, N, \quad j = 1, 2, \cdots, N_i$$

Mapping the above equations into the form containing all the objective function and constraints, we obtain the following:

$$\begin{aligned}
r_{i,j}^k &= \max\{0,\ \delta_{i,j}^k\left(S_j^k + p_j - S_i^k\right) \\
&\quad +(1-\delta_{i,j}^k)\left(S_i^k + p_i - S_j^k\right)\} \\
R_{i,j}^k &= \max\{0,\ \delta_{i,j}^k\omega_{i,j}^k\left(S_i^k + p_i - S_j^k - C_T\right) \\
&\quad +(1-\delta_{i,j}^k)\omega_{i,j}^k\left(S_j^k + p_j - S_i^k - C_T\right)\}
\end{aligned} \tag{4.36}$$

$$k = 1, \ldots, M \quad i = 1, \ldots, d_k - 1 \quad j = i+1, \ldots, d_k$$

Given the following relaxed problem, which accommodates the penalty function to relax the constraints, the energy function L is

$$\begin{aligned}
L &= \sum_{i=1}^{n}\sum_{j=1}^{m_i}\left(C_{i,j} + \omega_{i,j}\tilde{C}_{i,j}\right) \\
&\quad +\tfrac{1}{2}K\sum_{k=1}^{M}\sum_{h=1}^{\zeta_k}\mu_{h,k}\left(r_h^k(S) + R_h^k(S)\right)^2
\end{aligned} \tag{4.37}$$

or

$$
\begin{aligned}
L \;=\; & \\
& \sum_{i=1}^{n}\sum_{j=1}^{m_i}\left(C_{i,j}+\omega_{i,j}\tilde{C}_{i,j}\right) \\
& +\tfrac{1}{2}K\sum_{k=1}^{M}\sum_{h=1}^{\zeta_k}\mu_{h,k}\left(\max\{0,\ \delta_{i,j}^{k}\left(S_j^k+p_j-S_i^k\right)\right. \\
& \left.+(1-\delta_{i,j}^{k})\left(S_i^k+p_i-S_j^k\right)\right\} \\
& +\max\{0,\ \delta_{i,j}^{k}\omega_{i,j}^{k}\left(S_i^k+p_i-S_j^k-C_T\right) \\
& \left.+(1-\delta_{i,j}^{k})\omega_{i,j}^{k}\left(S_j^k+p_j-S_i^k-C_T\right)\right\}\Big)^2
\end{aligned}
\tag{4.38}
$$

Successful application of Hopfield network in solving the cyclic FMS problem requires the mapping of the Equation 4.37 to the Hopfield energy function.

4.4.2.1 Energy Function

The Hopfield network's algorithm is based on a gradient-type technique. From the theory of dynamic systems, the Lyapunov function discovered from Hopfield, or the energy function shown in Equation 4.39 has verified that stable states exist and can be achieved. As such this function has been used in Hopfield neural network to guarantee the convergence of the network system.

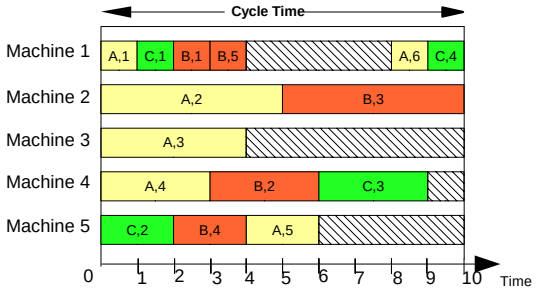
$$
E_{hopfield}=-\frac{1}{2}\sum_{x=1}^{N}\sum_{y=1}^{N}\sum_{I=1}^{N+1}\sum_{J+1}^{N+1}Ws_{xI}s_{yJ}+\sum_{x=1}^{N}\sum_{I}^{N+1}\vartheta_{xI}S_{xI} \tag{4.39}
$$

Where s_{yJ}, s_{xI} denotes neuron states, the weight between two particular neurons is defined as W and the threshold is defined as ϑ. This weight will constitute the strength of connection between neurons, and in each individual neuron, the threshold will determine whether the neuron is active or non-active.

However, even before the Hopfield network can be mapped, we need to formulate the cyclic flexible manufacturing systems problem in the form solvable by Hopfield network. This format of the schedule generally required us to define the problem in a two dimensional matrix. This matrix

would generally consist of $1, 0$ values. Each value or element of the matrix would define some sort of relationship that can be decoded into a solution or schedule.

Based on the matrix $N \times N + 1$ as *rows* $\times$ *column* or $x \times I$, the matrix would represent the schedule. N here is the total number of operations in the scheduling problem. Even though the jobs in the scheduling problem comprise of predetermined operations, in this matrix these operations are re-classified consecutively. The first column (denoted by ST in Figure 4.17(b)) of the matrix would indicate the operations that will be firstly scheduled on the individual machines. The corresponding operations in a columns would indicate the preceding operations to the row of the matrix. From this matrix we can deduce the sequence of all the operations on their individual machines. For example the following schedule can be defined as the matrix form as shown in Figure 4.17.



(a) A schedule for the cyclic FMS problem

	ST	A1	A2	A3	A4	A5	A6	B1	B2	B3	B4	B5	C1	C2	C3	C4
A1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
A2	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
A3	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
A4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
A5	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
A6	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
B1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
B2	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
B3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
B4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
B5	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
C1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
C3	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
C4	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

(b) Matrix showing corresponding schedule

Figure 4.17: A cyclic FMS schedule represented in (a) Gantt chart and its corresponding (b) 2D matrix form.

From this matrix a sequence can be obtained for each machine. This sequence has no start time allocated to each operation yet, merely an order in which to arrange the operations. Here we introduce a procedure to generate the complete schedule from the sequence. We name it as *Schedule-Generating Algorithm*. This is described in Figure 4.18. As such the Hopfield matrix would be used indirectly to define a cyclic schedule consisting of a particular start time for each operation in the system. Relatively speaking, a feasible schedule will be obtained from elements defined in the Hopfield matrix and the Work-In-Progress value can be calculated based on parameters described in Section 3.6.1.5.

Procedure in obtaining the schedule can be outlined as follow:

1. Sequence obtained from within elements s_{xI} of the Matrix. This sequence pertains to individual machines, where working from column elements to corresponding row elements, any element with value 1 will show a preceding relationship that can place the operation on the machines in a particular order.
2. An algorithm will work on the sequence to obtain the start times, S_{ij} of each operation by generating a feasible schedule. This Algorithm 4.18 is known as *Schedule-Generating Algorithm*.This is based on the known processing times of the all the operations. Using this information, the load of each machine can be calculated. In the case of cyclic flexible manufacturing systems, the highest load of all the machines will be used as the cycle time.
3. Calculation of the relevant value of the parameters associated to the work-in-progress, C,$\tilde{C}$,$r^k_{i,j}$ and $R^k_{i,j}$.

4.4.2.2 Mapping

It is required that the energy function and constraints from the cyclic scheduling problem be mapped into the Hopfield Energy function. This would enable the network to fully iterate in solving the problem discussed. Generally the cyclic scheduling problem will be formulated as the Lyapunov function associated with the Hopfield Network: $L = L_{obj} + L_{constraint^1} + L_{constraint^2} + L_{constraint^3} + \cdots$.

From the matrix defined, some restrictions to the elements of the matrix must be adhered to. This is important to set the matrix accurately to the constraints in the cyclic FMS problem. In the first column ($I = 1$), the total number of elements with value 1 will be equal to the number of machines, M in the system. This will ensure that only a maximum of M number of operations can be sequenced initially on the machines. Any number larger than M number of operations will not denote a feasible solution. Hence:

$$\left(\sum_{x=1}^{N} s_{x1} - M \right)^2 \tag{4.40}$$

Also in each column after the first column, the total number of elements with value 1 is only one. The rest of the elements in that particular column will be 0. This will limit the number of preceding relationship be-

Input: Sequence from matrix, Job List, Operation List, Processing Time, Machine List

Output: Schedule per machine for each job and operation

for *job* $j := 1$ **to** J **do**
  for *operation* $i := 1$ **to** i_j **do**
    if *on Seq* $= 1$ *of sequence list of machine* M_i **then**
      $StartTime :=$
      $\max\{CompletionTime_{operation^{seq-1}}, CompletionTime_{operation_{i-1,j}}\}$;
    end
    else if *on Seq* $= n$ *of sequence list of machine* M_i*, only if* $operation^{seq-1}$ *is scheduled* **then**
      $StartTime :=$
      $\max\{CompletionTime_{operation^{seq-1}}, CompletionTime_{operation_{i-1,j}}\}$;
      $StartTime = CompletionTime_{op_previous}$, if after scheduling, **foreach** *Seq+1 in the Sequence* **do**
        If$StartTime_{Operation^{Seq+1}} =$
        $StartTime_{Operation^{Seq}} + ProcessingTime_{Operation^{Seq}} > CycleTime$
        $StartTime =$
        $\min\{CompletionTime_{operation^{Seq-1}}, CompletionTime_{operation_{i-1,j}}\}$
      end
    else
      keep operations and all subsequent operations on **Unscheduled set**;
    end
  end
end
Retrieve operations in **Unscheduled set**;
Repeat steps ;

Figure 4.18: Schedule Generating Algorithm

tween operations on a particular machine to one. This can be termed as:

$$\sum_{I=1}^{N+1}\left(1-\sum_{x}^{N}s_{xI}\right)^2 \tag{4.41}$$

To ensure that the schedule is a cyclic scheduling type, the total number of preceding relationships is equal to the total number of operations in the system. This means that, by summing all the elements in the matrix except for the first column ($I = 1$), this should equate to N. Thus the following term is required:

$$\left(\sum_{I=2}^{N+1}\sum_{x}^{N}s_{xI}-N\right)^2 \tag{4.42}$$

The above restrictions in the matrix will replace the disjunctive constraints described in Equation 4.38 above.

However, in this cyclic flexible manufacturing system problem, the objective is to minimize the work-in-progress therefore we would include the objective function defined in Equation 4.35 into the combined function. We also would include the constraints that ensure that no overlap between operations occur over cycles.

By combining all the functions detailed above, we have the following:

$$\begin{aligned}
L = \\
&\frac{A}{2}\left(\sum_{x=1}^{N}s_{x1}-M\right)^2+\frac{B}{2}\sum_{I=1}^{N+1}\left(1-\sum_{x}^{N}s_{xI}\right)^2+\frac{D}{2}\left(\sum_{I=2}^{N+1}\sum_{x}^{N}s_{xI}-N\right)^2+\\
&\sum_{i=1}^{n}\sum_{j=1}^{m_i}\left(C_{i,j}+\omega_{i,j}\tilde{C}_{i,j}\right)+\\
&\frac{F}{2}\sum_{k=1}^{M}\sum_{h=1}^{\zeta_k}\mu_{h,k}\Big(\max\{0,\ \delta^k_{i,j}(S^k_j+p_j-S^k_i)+(1-\delta^k_{i,j})(S^k_i+p_i-S^k_j)\}+\\
&\max\{0,\ \delta^k_{i,j}\omega^k_{i,j}(S^k_i+p_i-S^k_j-C_T)+(1-\delta^k_{i,j})\omega^k_{i,j}(S^k_j+p_j-S^k_i-C_T)\}\Big)^2
\end{aligned} \tag{4.43}$$

In order to ascertain the weight and threshold function, we expand the

associated objective functions to their quadratic form:

$$\begin{aligned} L_{constraint^1} &= \frac{A}{2}\left(\sum_{x=1}^{N} s_{x1} - M\right)^2 \\ &= \frac{A}{2}\left(\sum_{x=1}^{N}\sum_{y=1}^{N} s_{x1}s_{y1} - 2M \times \sum_{x=1}^{N} s_{x1} + M^2\right) \\ &= \frac{A}{2}\sum_{x=1}^{N}\sum_{I=1}^{N+1}\sum_{y=1}^{N}\sum_{J=1}^{N+1} \phi_{IJ}^1 s_{xI}s_{yJ} - AM \times \sum_{x=1}^{N}\sum_{I=1}^{N+1} \phi_I^2 s_{xI} + \frac{A}{2}M^2 \end{aligned} \tag{4.44}$$

where

$$\phi_{IJ}^1 = \begin{cases} 1 & \text{if I=1 or J=1} \\ 0 & \text{if otherwise} \end{cases}$$

and

$$\phi_I^2 = \begin{cases} 1 & \text{if I=1} \\ 0 & \text{if otherwise} \end{cases}$$

with

$$\begin{aligned} L_{constraint^2} &= \frac{B}{2}\sum_{I=1}^{N+1}\left(1 - \sum_{x}^{N} s_{xI}\right)^2 \\ &= \frac{B}{2}\sum_{I=1}^{N+1}\left(1 - 2 \times \sum_{x}^{N} s_{xI} + \sum_{x}^{N}\sum_{y}^{N} s_{xI}s_{yI}\right) \\ &= \frac{B}{2} - B\sum_{I=1}^{N+1}\sum_{x}^{N} s_{xI} + \frac{B}{2}\sum_{I=1}^{N+1}\sum_{J=1}^{N+1}\sum_{x}^{N}\sum_{y}^{N} \rho_{IJ}s_{xI}s_{yI} \end{aligned} \tag{4.45}$$

where

$$\rho_{IJ} = \begin{cases} 0 & \text{if I} \neq \text{J} \\ 1 & \text{if I = J} \end{cases}$$

with

$$\begin{aligned} L_{constraint^3} &= \frac{D}{2}\left(\sum_{I=2}^{N+1}\sum_{x}^{N} s_{xI} - N\right)^2 \\ &= \frac{D}{2}\left(\sum_{I=2}^{N+1}\sum_{x}^{N}\sum_{J=2}^{N+1}\sum_{y}^{N} s_{xI}s_{yJ} - 2N \times \sum_{I=2}^{N+1}\sum_{x}^{N} s_{xI} + N^2\right) \\ &= \frac{D}{2}\sum_{I=1}^{N+1}\sum_{x}^{N}\sum_{J=1}^{N+1}\sum_{y}^{N} \varpi_{IJ}^{1} s_{xI}s_{yJ} - DN \times \sum_{I=1}^{N+1}\sum_{x}^{N} \varpi_{I}^{2} s_{xI} + \frac{D}{2}N^2 \end{aligned} \tag{4.46}$$

where

$$\varpi_{IJ}^{1} = \begin{cases} 0 & \text{if I=1 or J=1} \\ 1 & \text{if otherwise} \end{cases}$$

and

$$\varpi_{I}^{2} = \begin{cases} 0 & \text{if I=1} \\ 1 & \text{if otherwise} \end{cases}$$

Equating the function L to Energy function of the Hopfield Network, $E_{hopfield}$:

$$\begin{aligned} L = & \\ & \frac{A}{2}\sum_{x=1}^{N}\sum_{I=1}^{N+1}\sum_{y=1}^{N}\sum_{J=1}^{N+1} \phi_{IJ}^{1} s_{xI}s_{yJ} - AM \times \sum_{x=1}^{N}\sum_{I=1}^{N+1} \phi_{I}^{2} s_{xI} + \frac{A}{2}M^2 \\ & \frac{B}{2} - B\sum_{I=1}^{N+1}\sum_{x}^{N} s_{xI} + \frac{B}{2}\sum_{I=1}^{N+1}\sum_{J=1}^{N+1}\sum_{x}^{N}\sum_{y}^{N} \rho_{IJ} s_{xI}s_{yI} + \\ & \frac{D}{2}\sum_{I=1}^{N+1}\sum_{x}^{N}\sum_{J=1}^{N+1}\sum_{y}^{N} \varpi_{IJ}^{1} s_{xI}s_{yJ} - DN \times \sum_{I=1}^{N+1}\sum_{x}^{N} \varpi_{I}^{2} s_{xI} + \frac{D}{2}N^2 + \\ & \sum_{i=1}^{n}\sum_{j=1}^{m_i}\left(C_{i,j} + \omega_{i,j}\tilde{C}_{i,j}\right) + \\ & \frac{F}{2}\sum_{k=1}^{M}\sum_{h=1}^{\zeta_k} \mu_{h,k}\Big(\max\{0,\ \delta_{i,j}^{k}(S_j^k + p_j - S_i^k) + (1-\delta_{i,j}^{k})(S_i^k + p_i - S_j^k)\} + \\ & \max\{0,\ \delta_{i,j}^{k}\omega_{i,j}^{k}(S_i^k + p_i - S_j^k - C_T) + (1-\delta_{i,j}^{k})\omega_{i,j}^{k}(S_j^k + p_j - S_i^k - C_T)\}\Big)^2 \end{aligned} \tag{4.47}$$

We can take the summarized version of the constraints containing pa-

rameters start times, S from calculation of the constraints. This is due to the fact that the start times are actually generated optimally from the *Schedule-Generating Algorithm* based on the sequence obtained from the Hopfield matrix.

$$
\begin{aligned}
L = & \\
& \frac{A}{2}\sum_{x=1}^{N}\sum_{I=1}^{N+1}\sum_{y=1}^{N}\sum_{J=1}^{N+1}\phi_{IJ}^{1}s_{xI}s_{yJ} - AM \times \sum_{x=1}^{N}\sum_{I=1}^{N+1}\phi_{I}^{2}s_{xI} + \frac{A}{2}M^{2} \\
& \frac{B}{2} - B\sum_{I=1}^{N+1}\sum_{x}^{N}s_{xI} + \frac{B}{2}\sum_{I=1}^{N+1}\sum_{J=1}^{N+1}\sum_{x}^{N}\sum_{y}^{N}\rho_{IJ}s_{xI}s_{yI} + \\
& \frac{D}{2}\sum_{I=1}^{N+1}\sum_{x}^{N}\sum_{J=1}^{N+1}\sum_{y}^{N}\varpi_{IJ}^{1}s_{xI}s_{yJ} - DN \times \sum_{I=1}^{N+1}\sum_{x}^{N}\varpi_{I}^{2}s_{xI} + \frac{D}{2}N^{2} + \\
& \sum_{i=1}^{n}\sum_{j=1}^{m_i}\left(C_{i,j} + \omega_{i,j}\tilde{C}_{i,j}\right) + \frac{F}{2}\sum_{k=1}^{M}\sum_{h=1}^{\zeta_k}\mu_{h,k}\left(r_{h}^{k}(S) + R_{h}^{k}(S)\right)^{2} \qquad (4.48)
\end{aligned}
$$

Thus in order to determine the weights and threshold, the Hopfield energy function is:

$$
E_{hopfield} = -\frac{1}{2}\sum_{x=1}^{N}\sum_{y=1}^{N}\sum_{I=1}^{N+1}\sum_{J+1}^{N+1}W_{xI,yJ}s_{xI}s_{yJ} + \sum_{x=1}^{N}\sum_{I}^{N+1}\vartheta_{xI}S_{xI} \qquad (4.49)
$$

with weights defined between neuron (xI) and (yJ) as:

$$
W_{xI,yJ} = -\left(A\phi_{IJ}^{1} + B\rho_{IJ} + D\varpi_{IJ}^{1}\right) \qquad (4.50)
$$

and threshold for each neuron in the network as:

$$
\vartheta_{xI} = -\left(AM\phi_{I}^{2} + B + DN\varpi_{I}^{2}\right) \qquad (4.51)
$$

So here the function L equals to $E_{hopfield}$ accurate to a constant:

$$
E = L - \sum_{i=1}^{n}\sum_{j=1}^{m_i}\left(C_{i,j} + \omega_{i,j}\tilde{C}_{i,j}\right) - \frac{F}{2}\sum_{k=1}^{M}\sum_{h=1}^{\zeta_k}\mu_{h,k}\left(r_{h}^{k}(S) + R_{h}^{k}(S)\right)^{2} \qquad (4.52)
$$

Here the values of constants A, B, D and F are empirically determined. Each coefficient has a weighting influence on the term it is attached to,

when all these terms are summed up into the overall function. However each term in the function links the constraints appropriately to the inner dynamics of the Hopfield Network.

4.4.2.3 Energy Function Convergence

The dynamic of the energy function can be calculated as:

$$\Delta E_{hopfield} = -\left(\sum_{y=1}^{N}\sum_{J}^{N+1} W_{xI,yJ}s_{yJ} - \vartheta_{xI}\right)\Delta s_{xI} \tag{4.53}$$

Thus, the negative convergence from Equation 4.53 with every iteration of change in parameter s_{xI} will bring the state of the system lower, descending eventually settling in a stable, optimal position.

The element in the matrix will be updated according to :

$$\frac{du_{xI}}{dt} = -\frac{u_{xI}}{\upsilon} + \sum_{y}^{N}\sum_{J}^{(N+1)} W_{xI,yJ}s_{yJ} + \vartheta_{xI} \tag{4.54}$$

where constant υ is a decaying constant.

$$s_{xI}^{(k+1)} = g(u_{xI}^{k} + \frac{du_{xI}}{dt}) \tag{4.55}$$

The above function will guarantee that the value of s_{xI} is fixed either as 0 or 1. In order to restrict the generation of possible schedules in the associated Hopfield matrix in each iteration, it is vital for us to optimally control the assignment of values to the elements, especially in the case where the disjunctive constraints should not be violated. A master set that contains the sets of operations per machine is identified. These operations in the set with disjunctive relationship are enforced. Overall, this will eliminate the possibility of the matrix having unrecognizable disjunctive links between operations that are to be processed on different machines. In order to control this, the function $g(u)$ will identify operations from a disjunctive set, the function $g(u)$ will then null the ability of particular neurons, not in disjunctive set to activate.

$$g(s_{xI}^{k}) = \begin{cases} 0 & \text{if } x_{xI} \text{ is not in Disjunctive set} \\ sgn(s_{xI}^{k}) & \text{if } x_{xI} \text{ is in Disjunctive set} \end{cases}$$

generally s_{xI} in next iteration is determined from:

$$s_{xI}^{(k+1)} = \begin{cases} 1 & \text{if } u_{xI}^{k} > \vartheta_{xI} \\ 0 & \text{if } u_{xI}^{k} < \vartheta_{xI} \\ s_{xI}^{k} & \text{if } u_{xI}^{k} = \vartheta_{xI} \end{cases}$$

4.4.3 Solving Cyclic Scheduling Problem with Advanced Hopfield Network

In order for the mapped Hopfield network to effectively solve the cyclic problem, we include the simulated annealing, based on a schedule decaying Temperature T effect to obtain the optimal solution. This optimal solution will correlate with the minimum energy function. The effect of the simulated annealing will also help the Hopfield network to release the system from local minimum states and transcend to the most global optimum state.

Based on the probability $Pr(x^{(k+1)})$, this will determine if we continue to keep state x or continue to change.

$$Pr(x^{(k+1)}) = \begin{cases} 1 & \text{if } \Delta E_{hopfield} < 0 \\ e^{-\frac{\Delta E}{T}} & \text{if } \Delta E_{Hopfield} \geq 0 \end{cases}$$

In the case when the state changes from a low level to a higher level energy state, $Pr(x^{(k+1)}) = e^{-\frac{\Delta E}{T}}$ is dependent on whether $Pr(x^{(k+1)})$ calculated is greater than a random number, η from a uniform distribution of range $[0, 1]$. If $Pr(x^{(k+1)}) > \eta$, then the new state is accepted. The temperature T is initialized to a high value prior to starting the simulation but changes based on fixed iteration k value:

$$T = \frac{T_0}{log(k)} \tag{4.56}$$

Algorithm 4.19 describes the simulated annealing algorithm used with the Hopfield network. The simulated annealing here will force the system to assess its state, using a high value of initial Temperature T_0, with a slow enough decaying rate, so the numbers of iteration assessed at that particular decaying Temperature can be controlled and evaluated to finally return an optimum solution.

Steps in Advanced Hopfield Network in solving the cyclic FMS scheduling problem can be summarized as follow:

- Initialize neuron matrix with initial schedule form,
- Define initial Temperature, T_0 and predefined value k to modify the Temperature,
- Compute value from energy function,
- Calculate and change the value of the element from the matrix,
- Calculate the change in energy value,
- Evaluate using the Simulated Annealing Algorithm depending on value of change in energy value,
- If energy function improves, accept new state otherwise reuse previous state,
- Repeat computation of energy function value and interchanging until predetermined iteration is achieved,
- Find minimum value for energy function,
- Calculate the minimum work-in-progress schedule.

Input: Schedule S, Change in Energy Value $\Delta\ E$, Temperature T
Output: Schedule conforming to Simulated Annealing effect

$i := 0$;
Obtain schedule S and $\Delta\ E$;
s^*:=s;
if $\Delta\ E$ ¡ 0 **then**
    $Pr(x^{(k+1)}) = 1$;
    s^*:=s;
else
    if *random [0,1]* $< \{exp(-\frac{\Delta E}{T})\}$ **then**
        s^*:=s;
    end
end

Figure 4.19: Simulated Annealing Hopfield Algorithm

4.4.4 Experimental Results

We developed the advanced Hopfield network and simulated the approach using the JAVA language on 2GHz Intel processor, 512Mb mem-

FMS Benchmarks									
Problem	N	M	O	CT	Feasible solutions	$W.I.P._{LB}$	$W.I.P._{best}$ Advanced Hopfield	$Iteration_{Best}$	CPU(s)
HIL87	4	4	15	8	1.6E+7	5	5	9020	16.6s
HIL88	4	3	9	6	1.1E+4	5	5	6530	12.3s
VAL94	3	5	13	11	7.9E+7	5	5	9870	18.2s

Table 4.4: Solution for CFMS benchmarks with Minimum WIP using Advanced Hopfield Network

ory computer. To show the practicality of our approach, we tested 3 major test problems associated with the cyclic FMS problem. The problems are HIL87 [70] with 4 jobs and 4 machines, HIL88 [71] with 4 jobs and 3 machines and VAL94 [151] with 3 jobs and 5 machines.

The parameters selected for the simulation were with values of A=200,B=400,D=100,F=100. These parameters were determined empirically and a decaying rate of 500 iterations was set. The maximum iterations for which the network to iterate to, was set at 10000 iterations. Results from advance Hopfield Neural Network Approach can be seen in solving the cyclic flexible manufacturing system scheduling problem shown in Table 4.4. The results show that our approach was successful in solving the three problems. The optimal solution was fond in the cpu time of between 12s and 18s. Further experimental results are published in Chapter 5 when this Advanced Hopfield network approach is compared to the RNN and LRRNN approach for solving more benchmark problems.

4.5 Conclusions

In this chapter, we proposed the Recurrent Neural Network approach. A description of the dynamics of the network is given that is based on the modelling of the cyclic job shop scheduling problem found in Section 3.5. This RNN approach utilizes the schedule generated from the Competitive Dispatch Rule Phase (CDRP). We applied the CDRP as an option to ensure a viable schedule is fed into the recurrent neural network. This will also reduce the search space to which the RNN to look for the optimum solution.

Given that the recurrent neural network have the characteristic of being trapped in local minima states during iteration, we include a Schedule Perturbation Phase into the activation function. This will ”bounce” the

energy state into a new state to continue in the search space, once it is detected that the energy state has stabilized below the predefined threshold value. Finally we included the Postprocessing Phase that will ensure that the end optimal schedule generated, is feasible in terms of non-violation of constraints.

In this chapter we also described the Lagrangian Relaxation approach in solving constrained optimization problems. We also developed the combined *Lagrangian Relaxation Recurrent Neural Network (LRRNN)* technique specifically for solving the cyclic flexible manufacturing system scheduling problem.

We then described the Modified Competitive Dispatch Rule Phase (MDCDP) to present initial schedules that are viable to the LRRNN. Combining this phase with the **Postprocessing Phase**, both approach will be able to the handle the difficulties of infeasibilities in solutions. We also simulated the approach on a benchmark HIL87 and the numerical results for the problem showed that LRRNN is able to generate optimal schedules, compared to the genetic algorithm approach. Although this LRRNN has been developed to solve the CFMSSP, it is easily extended to solve the cyclic job shop problem.

We have also described the cyclic job shop with linear constraint with a delinearization approach. We then applied the Delinearization algorithm on four test problems. By analyzing the numerical results of the model and solving approach against the Genetic Algorithm approach [27], constraint satisfaction problem (CSP) approach [18] and a constraint optimization Problem (COP) approach [18], we deduced that this approach is comparatively efficient and able to solve the cyclic scheduling problems with linear precedence constraints.

Finally in this chapter, we introduced the advanced Hopfield network. The energy function, related parameters and dynamics associated with this neural network specified to solve the cyclic flexible manufacturing systems scheduling problem were described in detail.

Our motivation in applying the Advanced Hopfield network in this particular cyclic scheduling problem stemmed from the lack of research on cyclic FMS problem from this type of neural network.

We are able to solve the cyclic scheduling problem using new advanced Hopfield approach. Using Lyapunov equation of the Hopfield network,

the most effective weights and bias values are deduced. We also developed a new binary representation of the cyclic FMS scheduling problem in a matrix form that the Hopfield network can be mapped to. We have shown the application of the *Schedule-Generating Algorithm* that can decode the matrix into a viable schedule. This has thus allowed the Work in progress of the FMS to be calculated. Also by incorporating the Simulated Annealing effect into the inner dynamics of the Hopfield Network, the iterated solutions obtained through the Hopfield Network, will avoid the numerous local minima but eventually converge to a global minima.

Also we verified our approach on three important test problems and the results found confirmed its accuracy. So our approach uses the internal optimization of the Hopfield network to solve the cyclic scheduling problem. However one concern associated with our approach is the size of the network that depends on the number of operations of the problem. Further research into the effectiveness of out Hopfield approach in the case of larger and more complex test problems will be discussed in **Chapter** 5

Part III

Experiments

CHAPTER

FIVE

Experimental Results

5.1 Benchmark Problems

Several benchmark problems were used in evaluating and comparing our proposed approaches. These benchmark problems consist of mainly cyclic modification on acyclic benchmark problems. The reason this is done is because of the lack of actual benchmark problems representing the cyclic scheduling problem.

Brucker et. al. (2005) [19] described how the cyclic job shop scheduling problem is comparative to the acyclic scheduling problem with the addition of source and sink operations in disjunctive graph. When we consider the precedence graph of Vertices T and Arcs E, the CJSSP has additional precedence constraints between the sink and source operation to denote a cyclic state. The delay $L_{i;i'}$ is equivalent to the processing time of the operation and Height $H_{i;i'} = 0$ for all arcs. However setting $H_{sink:source} = 1$ will set all operations in the $k - th$ period to be completed before operations of period $k + 1$ can start. Hence, the cyclic problem to minimize the cycle time is equivalent to the non-cyclic makespan minimization problem. This will allow us to compare CJSSP results with the non-cyclic scheduling benchmark results. The objectives when solving these benchmarks are to seek the best optimal cycle time in the case of CJSSP problems and the best optimal WIP in the case of CFMSSP.

5.2 Test Data for Cyclic Job Shop Scheduling Problem

For CJSSP problems, the benchmarks were formulated from papers published by *Fisher and Thompson (1963)* [45], *Lawrence (1984)* [102], *Adams et. al. (1988)* [2], *Applegate and Cook (1991)* [10] and *Storer et. al. (1992)* [145]. To fully understand the significance of testing our approach to solve the cyclic scheduling problem, we list some descriptions of the benchmark problems as follow:

FT06,FT10,FT20 These *Fisher and Thompson (1963)* [45] benchmarks consist of 6 Jobs$\times$6 Machines,10 Jobs$\times$10 Machines and 20 Jobs$\times$5 Machines. The processing times for the operations are in the range of [1, 99]. Table 5.1 shows the details for these problems.

LA01-LA40 These 40 problem instances from *Lawrence (1984)* [102] consist of 10 Jobs$\times$5 Machines,15 Jobs$\times$15 Machines,20 Jobs$\times$5 Machines,10 Jobs$\times$10 Machines,15 Jobs $\times$10 Machines,20 Jobs$\times$10 Machines,30 Jobs$\times$10 Machines and 15 Jobs$\times$5 Machines. The processing times for the operations are in the range of [5, 99]. Table 5.2 shows the details for these problems.

ABZ5-ABZ9 *Adams et. al. (1988)* [2] generated 5 problem instances consisting of 10 Jobs$\times$10 Machines and 20 Jobs$\times$15 Machines. The processing times for the operations are in the range of [50, 100],[25, 100] and [11, 40]. Table 5.3 shows the details for these problems.

ORB01-ORB10 The 10 instances from *Applegate and Cook (1991)* [10] consist of 10 Jobs$\times$10 Machines. The processing times for the operations are in the range of [1, 99]. Table 5.4 shows the details for these problems.

5.3 Preparation of Benchmark Problems

Given the information regarding the:

i Number of jobs per cyclic scheduling problem,

ii Number of machines per cyclic scheduling problem,

iii Processing times of each operation,

we deduced the precedence constraints and disjunctive constraints between operations. Initializing the cyclic scheduling problems based on the specification of the authors involves two stages. Firstly, using Competitive Dispatch Rule Phase (CDRP), a schedule is selected for the individual cyclic job shop scheduling problems. This schedule lies in a preset time range and forms the initial solution for the cyclic scheduling problem to be worked on.

Several assumptions are taken when solving the cyclic job shop scheduling problems:

1. Once the operation has been started on the machine, it must not be preempted (i.e. no preemption is allowed).
2. The latency considered in each solution is based on latency equal to 1.
3. The solutions were obtained from best, optimal and feasible solutions from 20 runs.

5.4 Experiments Set-up

To fully evaluate our approach, a RNN and LRRNN approach was developed on a JAVA program on 2GHz Intel Pentium 4 processor, 512Mb memory computer running a Linux (i.e Fedora Core 6) operating system. The problems were modelled and instances were loaded onto the RNN and LRRNN model. These solutions were evaluated based on the following criteria:

1. Speed or computational time to obtain optimal solution,
2. The accuracy of solutions found against known optimal results, measured from Percentage deviation of solution (i.e. $MRE(\%)$ where:

$$\%Dev = \left\{ \frac{(\text{Best Solution Found}) - (\text{Optimal Solution})}{(\text{Optimal Solution})} \times 100 \right\} \quad (5.1)$$

3. Percentage improvements between the Recurrent Neural Network (RNN) with special phases and LRRNN approach in terms of CPU processing time and number of iterations to reach optimal solution.

Problem	No. of jobs	No. of machines	No. of operations	No. of Conjunctive Constraints	No. of Disjunctive Constraints
FT06	6	6	36	30	90
FT10	10	10	100	90	450
FT20	20	5	100	80	950

Table 5.1: Benchmark CJSSP Problem from *Fisher and Thompson (1963)* [45]

These problems were tested by initially generating schedules using the Competitive Dispatch Rules Phase (CDRP) prior to being fed into the RNN and LRRNN model. The best solutions were then treated with the *Schedule Postprocessing Phase* to guarantee feasibility.

5.4.1 Termination Criteria

The experiments were executed with several termination criteria. The execution will stop after a maximum iteration is reached. In the cases above, the upper limit for maximum execution was fixed prior to starting the experiments at $10e6$ executions. There is however no computational time limit set for the execution to end.

5.5 Results for Cyclic Job Shop Scheduling Problems

In this section, we present the computational results for the different benchmarks of the cyclic job shop scheduling problem.

We also will show the optimal solution found for each instance of the problem. Furthermore we will also analyze the performance of both approaches. We will compare our solutions against best results from various other researchers.

The best results for the cyclic job shop are given in Table 5.6 showing results obtained for problems LA01-LA40 from *Lawrence (1984)* [102]. Table 5.5 shows best results obtained for problems FT06, FT10 and FT20 from *Fisher and Thompson (1963)* [45]. While Table 5.7 shows results obtained for problems ABZ5-ABZ9 from *Adams et. al. (1988)* [2]. Finally Table 5.8 shows results obtained for the 10 problems ORB01-ORB10 from *Applegate and Cook (1991)* [10]. We have printed the best results in blue if the best result is equal to the known computed lower bound, which means that we have found the optimal solution.

Problem	No. of jobs	No. of machines	No. of operations	No. of Conjunctive Constraints	No. of Disjunctive Constraints
LA01	10	5	50	40	225
LA02	10	5	50	40	225
LA03	10	5	50	40	225
LA04	10	5	50	40	225
LA05	10	5	50	40	225
LA06	15	5	75	60	525
LA07	15	5	75	60	525
LA08	15	5	75	60	525
LA09	15	5	75	60	525
LA10	15	5	75	60	525
LA11	20	5	100	80	950
LA12	20	5	100	80	950
LA13	20	5	100	80	950
LA14	20	5	100	80	950
LA15	20	5	100	80	950
LA16	10	10	100	90	450
LA17	10	10	100	90	450
LA18	10	10	100	90	450
LA19	10	10	100	90	450
LA20	10	10	100	90	450
LA21	15	10	150	135	1050
LA22	15	10	150	135	1050
LA23	15	10	150	135	1050
LA24	15	10	150	135	1050
LA25	15	10	150	135	1050
LA26	20	10	200	180	1900
LA27	20	10	200	180	1900
LA28	20	10	200	180	1900
LA29	20	10	200	180	1900
LA30	20	10	200	180	1900
LA31	30	10	300	270	4350
LA32	30	10	300	270	4350
LA33	30	10	300	270	4350
LA34	30	10	300	270	4350
LA35	30	10	300	270	4350
LA36	15	15	225	210	1575
LA37	15	15	225	210	1575
LA38	15	15	225	210	1575
LA39	15	15	225	210	1575
LA40	15	15	225	210	1575

Table 5.2: Benchmark CJSSP Problem from *Lawrence (1984)* [102]

Problem	No. of jobs	No. of machines	No. of operations	No. of Conjunctive Constraints	No. of Disjunctive Constraints
ABZ5	10	10	100	90	450
ABZ6	10	10	100	90	450
ABZ7	20	15	300	280	2850
ABZ8	20	15	300	280	2850
ABZ9	20	15	300	280	2850

Table 5.3: Benchmark CJSSP Problem from *Adams et. al. (1988)* [2]

Problem	No. of jobs	No. of machines	No. of operations	No. of Conjunctive Constraints	No. of Disjunctive Constraints
orb01	10	10	100	90	450
orb02	10	10	100	90	450
orb03	10	10	100	90	450
orb04	10	10	100	90	450
orb05	10	10	100	90	450
orb06	10	10	100	90	450
orb07	10	10	100	90	450
orb08	10	10	100	90	450
orb09	10	10	100	90	450
orb10	10	10	100	90	450

Table 5.4: Benchmark CJSSP Problem from *Applegate and Cook (1991)* [10]

FT Problems, 6 Jobs × 6 Machines, 36 Operations												
				RNN				LRNN				
Problem	CT_{LB}	DisjConstr	ConjConstr	CT_{Best}	$Iteration_{Best}$	CPU(s)	$MRE(\%)$	CT_{Best}	$Iteration_{Best}$	CPU(s)	$MRE(\%)$	$\%Improvement$
FT06	55	30	90	55	336	0.62	0	55	339	0.62	0%	0%
FT Problems, 10 Jobs × 10 Machines, 100 Operations												
FT10	930	90	450	930	18815	34.62	0	930	19087	35.12	0%	0%
FT Problems, 20 Jobs × 5 Machines, 100 Operations												
FT20	1165	80	950	1165	8092	14.89	0	1165	8210	15.10	0%	0%

Table 5.5: Best Results found for benchmark *Fisher and Thompson (1963)* [45]

La										
		RNN				LRNN				
Problem	CT_{LB}	CT_{Best}	$Iteration_{Best}$	CPU(s)	$MRE(\%)$	CT_{Best}	$Iteration_{Best}$	CPU(s)	$MRE(\%)$	$\%Improvement$
10 Jobs × 5 Machines, 50 Operations										
LA01	666	666	4947	9.12	0.0%	666	5023	9.26	0.0%	0.0%
LA02	655	655	11896	21.93	0.0%	655	12043	22.20	0.0%	0.0%
LA03	597	597	133883	246.80	0.0%	597	135409	249.62	0.0%	0.0%
LA04	590	590	130742	241.01	0.0%	590	132337	243.95	0.0%	0.0%
LA05	593	593	2042	3.76	0.0%	593	2078	3.83	0.0%	0.0%
15 Jobs × 5 Machines, 75 Operations										
LA06	926	926	5026	9.26	0.0%	926	5129	9.45	0.0%	0.0%
LA07	890	890	5929	10.93	0.0%	890	6047	11.15	0.0%	0.0%
LA08	863	863	17746	32.71	0.0%	863	17931	33.05	0.0%	0.0%
LA09	951	951	3337	6.15	0.0%	951	3418	6.30	0.0%	0.0%
LA10	958	958	54967	101.33	0.0%	958	56126	103.46	0.0%	0.0%
20 Jobs × 5 Machines, 100 Operations										
LA11	1222	1562	7970	14.69	27.8%	1222	8334	15.36	0.0%	21.8%
LA12	1039	1039	3416	6.30	0.0%	1039	3542	6.53	0.0%	0.0%
LA13	1150	1150	4829	8.90	0.0%	1150	4955	9.13	0.0%	0.0%
LA14	1292	1487	3691	6.80	15.1%	1292	3778	6.96	0.0%	13.1%
LA15	1207	1402	12132	22.36	16.2%	1207	12735	23.48	0.0%	13.9%
10 Jobs × 10 Machines, 100 Operations										
LA16	945	982	231645	427.02	3.9%	957	235050	433.30	1.3%	2.5%
LA17	784	784	369062	680.34	0.0%	784	375963	693.06	0.0%	0.0%
LA18	848	848	416176	767.19	0.0%	848	422169	778.24	0.0%	0.0%
LA19	842	859	451512	832.33	2.0%	842	459368	846.81	0.0%	2.0%
LA20	902	1052	677898	1249.65	16.6%	908	690032	1272.02	0.7%	13.7%
15 Jobs × 10 Machines, 150 Operations										
LA21	1046	1248	277687	511.89	19.3%	1074	289905	534.42	2.7%	13.9%
LA22	927	1047	718493	1324.48	12.9%	932	748957	1380.64	0.5%	11.0%
LA23	1032	1059	883393	1628.46	2.6%	1054	925707	1706.47	2.1%	0.5%
LA24	935	1024	877567	1617.72	9.5%	944	920392	1696.67	1.0%	7.8%
LA25	977	1049	385552	710.73	7.4%	984	405523	747.55	0.7%	6.2%
20 Jobs × 10 Machines, 200 Operations										
LA26	1218	1322	208088	383.59	8.5%	1224	259669	478.68	0.5%	7.4%
LA27	1235	1276	766889	1413.70	3.3%	1249	1168739	2154.48	1.1%	2.1%
LA28	1216	1287	1197488	2207.47	5.8%	1235	1506440	2777.00	1.6%	4.0%
LA29	1152	1458	1377908	2540.06	26.6%	1249	1824350	3363.04	8.4%	14.3%
LA30	1355	1396	776778	1431.93	3.0%	1355	1022706	1885.28	0.0%	2.9%
30 Jobs × 10 Machines, 300 Operations										
LA31	1784	2189	150373	277.20	22.7%	1855	162613	299.76	4.0%	15.3%
LA32	1850	2478	114252	210.61	33.9%	1947	123312	227.32	5.2%	21.4%
LA33	1719	2416	100510	185.28	40.5%	1840	111707	205.92	7.0%	23.8%
LA34	1721	2441	108363	199.76	41.8%	1849	109880	202.55	7.4%	24.3%
LA35	1888	2511	83628	154.16	33.0%	1994	88930	163.94	5.6%	20.6%
15 Jobs × 15 Machines, 225 Operations										
LA36	1268	1468	1911586	3523.86	15.8%	1299	2455623	4526.75	2.4%	11.5%
LA37	1397	1574	1619023	2984.54	12.7%	1405	2225509	4102.55	0.6%	10.7%
LA38	1196	1369	1647812	3037.61	14.5%	1257	2238717	4126.90	5.1%	8.2%
LA39	1233	1548	1264812	2331.58	25.5%	1302	1843413	3398.18	5.6%	15.9%
LA40	1222	1325	1589312	2929.77	8.4%	1320	1903138	3508.28	8.0%	0.4%

Table 5.6: Best Results found for benchmark *Lawrence (1984)* [102]

ABZ Problems, 10 Jobs × 10 Machines, 100 Operations										
		RNN				LRNN				
Problem	CT_{LB}	CT_{Best}	$Iteration_{Best}$	CPU(s)	$MRE(\%)$	CT_{Best}	$Iteration_{Best}$	CPU(s)	$MRE(\%)$	$\%Improvement$
ABZ5	1234	1455	84898	156.50	17.9%	1297	91367	168.12	5.1%	10.9%
ABZ6	943	1032	675548	1245.32	9.4%	978	721147	1326.91	3.7%	5.2%
ABZ Problems, 20 Jobs × 15 Machines, 300 Operations										
ABZ7	656	748	1229878	2267.18	14.0%	694	1325808	2439.49	5.8%	7.2%
ABZ8	645	745	554231	1021.68	15.5%	697	585157	1076.69	8.1%	6.4%
ABZ9	661	786	588761	1085.33	18.9%	694	620554	1141.82	5.0%	11.7%

Table 5.7: Best Results found for benchmark *Adams et. al. (1988)* [2]

ORB Problems, 10 Jobs × 10 Machines, 100 Operations										
		RNN				LRNN				
Problem	CT_{LB}	CT_{Best}	$Iteration_{Best}$	CPU(s)	$MRE(\%)$	CT_{Best}	$Iteration_{Best}$	CPU(s)	$MRE(\%)$	$\%Improvement$
ORB01	1059	1076	77898	143.60	1.6%	1061	83834	154.25	0.2%	1.4%
ORB02	888	923	85452	157.52	3.9%	888	86067	158.36	0.0%	3.8%
ORB03	1005	1022	94556	174.31	1.7%	1005	101761	187.24	0.0%	1.7%
ORB04	1005	1019	76523	141.06	1.4%	1005	82354	151.53	0.0%	1.4%
ORB05	887	892	39006	71.90	0.6%	887	41557	76.46	0.0%	0.6%
ORB06	1010	1018	77898	143.60	0.8%	1029	83694	154.00	1.1%	-1.9%
ORB07	397	402	66578	122.73	1.3%	397	68708	126.42	0.0%	1.2%
ORB08	899	913	88009	162.24	1.6%	921	95842	176.35	2.4%	-0.9%
ORB09	934	956	75447	139.08	2.4%	966	78125	143.75	3.4%	-1.0%
ORB10	944	959	66591	122.76	1.6%	969	69035	127.02	2.6%	-1.0%

Table 5.8: Best Results found for benchmark *Applegate and Cook (1991)* [10]

As we can see, both RNN and LRRNN approaches compute very good results for test problems FT06,FT10,FT20 and test problems LA01 to LA10. The LRRNN approach managed to compute best results for problems LA11 to LA15, while performed competitively for large problems containing 100 operations and 150 operations as shown in results for LA16 to LA28. In the case of problems ORB01 to ORB10, the LRRNN approach managed to outperform the RNN approach by finding the best results for 5 out of the 10 problems (i.e. ORB02, ORB03, ORB04, ORB05 and ORB07). However both RNN and LRRNN approach under-performed in solving the 5 ABZ problems.

Although our RNN approach performed well for cyclic job shop problems with cycle time up to 1165 (i.e. in the case of problem FT20 and total number of constraints up to 585, this particular approach was underperforming for larger problems with lower bound cycle time above this value. This is obvious in the case of problems with 300 operations (i.e. problems LA31-LA35 where the RNN approach produced an average deviation from the optimal results of 34.4%. However this could have been greatly influenced by the fact that these problems have the largest total number of constraints, among all the test problems tested, comprising of 270 conjunctive constraints and 4350 disjunctive constraints.

The LRRNN approach is most efficient for problems with up to 100 operations but is not competitive in the case of problems with 300 operations (i.e average $MRE(\%)$ of 5.86%) or with lower bound cycle time greater than 1784 (i.e problems [LA31-LA35].

Table 5.9 shows the performance analysis of both our RNN and LRRNN approaches. We find that by analyzing the computational time for our two approaches in reaching the optimal solution, we can see that in the case of the RNN approach, it will require an average of 104 seconds of CPU time. However the RNN approach may retire at an early stage of the iteration if the problem is complex (e.g. stuck at average of 205s for problems LA31-35 with MRE(%) of 34.43%), possibly trapped in a local minima that the perturbationphase is unable to release the model from.

On the other hand, in the case of LRRNN, this approach may also suffer from the same setback of retiring early for very complex problems without reaching the optimum solution (e.g. stuck at average of 219s for problems LA31-35 with MRE(%) of 5.86%). Although this could be problem specific as in the case of nearly similar problems of ABZ7-AB9 (of 300 operations), the LRRNN had taken a average of 1552s to obtain the best solution with MRE(%) of 6.28%. When we analyze the computational time it takes for both approaches to reach optimal solution in small problems, the RNN approach is stronger as this approach will reach optimal solution is less time than the LRRNN approach. This is obvious in the case of FT06, FT10 and FT20 where RNN was 1.61%, 1.44% and 1.48% faster compared to the LRRNN approach.

Overall, the LRRNN approach outperform the RNN approach in all the CJSSP problems by an average of 5.14%. Figures 5.1,5.2 and 5.3 graphically show the deviation from the optimum solutions for all CJSSP benchmarks when comparing RNN and LRRNN approaches.

5.6 Test Data for Cyclic Flexible Manufacturing Scheduling Problem

In order to further analyze our approaches in solving the cyclic flexible manufacturing systems scheduling problem, we list some descriptions of the following eight benchmark problems from FMS literature:

				RNN		LRNN					
Problem	N	M	O	Number of ConjConstr	Number of DisjCons	Average CPU(s)	Average $MRE(\%)_{RNN}$	Average CPU(s)	Average $MRE(\%)_{LRRNN}$	Average $\%Improv$	Average $\%ImprComT$
FT06	6	6	36	30	90	0.62	0.00%	0.63	0.00%	0.00%	-1.61%
FT10	10	10	100	90	450	34.62	0.00%	35.12	0.00%	0.00%	-1.44%
FT20	20	5	100	80	950	14.89	0.00%	15.11	0.00%	0.00%	-1.48%
LA01-LA05	10	5	50	40	225	104.53	0.00%	105.77	0.00%	0.00%	-1.19%
LA06-LA10	15	5	75	60	525	32.08	0.00%	32.68	0.00%	0.00%	-1.87%
LA11-LA15	20	5	100	80	950	11.81	11.81%	12.29	0.00%	11.81%	-4.06%
LA16-LA20	10	10	100	90	450	791.3	4.51%	804.68	0.39%	4.12%	-1.69%
LA21-LA25	15	10	150	135	1050	1158.66	10.35%	1213.15	1.41%	8.94%	-4.70%
LA26-LA30	20	10	200	180	1900	1595.35	9.46%	2131.7	2.32%	7.14%	-33.62%
LA31-LA35	30	10	300	270	4350	205.4	34.41%	219.9	5.86%	28.55%	-7.06%
LA36-LA40	15	15	225	210	1575	2961.47	15.38%	3932.53	4.35%	11.03%	-32.79%
ABZ5-ABZ6	10	10	100	90	450	700.91	13.67%	747.51	4.41%	9.26%	-6.65%
ABZ7-ABZ9	20	15	300	280	2850	1458.07	16.15%	1552.67	6.28%	9.87%	-6.49%
ORB01-ORB10	10	10	100	90	450	137.88	1.67%	145.54	0.98%	0.69%	-5.56%

Table 5.9: Performance Analysis for CJSSP benchmarks for RNN and LRRNN approaches

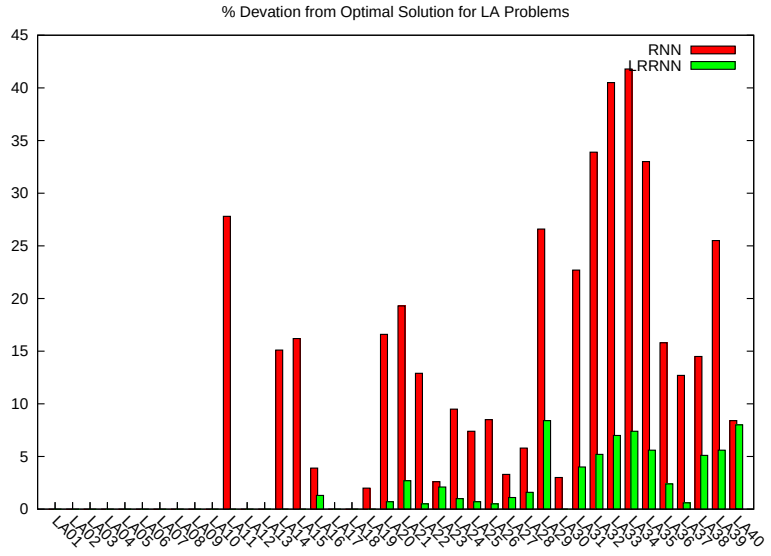


Figure 5.1: Percentage Deviation From Optimal Solution for LA Problems

HIL87 This *Hillion et. al. (1987)* [70] benchmarks consist of 4 Jobs×4 Machines. Jobs (A, B, C, D) require (4, 4, 3, 4) operations each.

HIL88 This 4 Jobs×3 Machines problem comes from *Hillion and Proth (1988)* [71] where Jobs (A, B, C, D) require (3, 2, 2, 2) operations each.

VAL94 *Valentin (1994)* [151] generated a 5 Jobs×3 Machines cyclic FMS problem. The Jobs (A, B, C, D, E) require (3, 3, 3, 2, 2) operations each.

OHL95 The *Ohl et. al. (1995)* [129] problem consists of 2 Jobs×6 Machines. Jobs (A, B) have (6, 4) operations each.

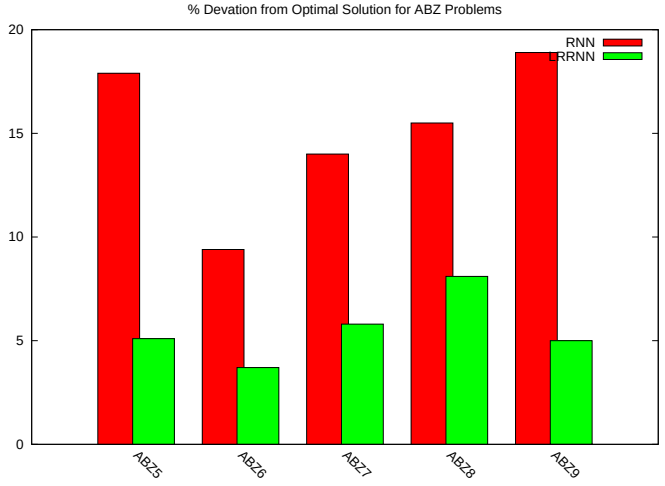


Figure 5.2: Percentage Deviation From Optimal Solution for ABZ Problems

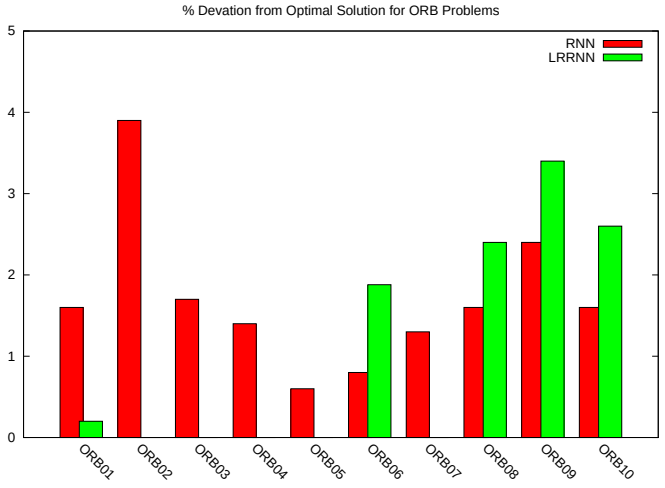


Figure 5.3: Percentage Deviation From Optimal Solution for ORB Problems

KORBAA98a The particular complex cyclic FMS problem from *Korbaa (1998)* [94] consists of 7 Jobs×9 Machines. Jobs (A, B, C, D, E, F, G) require (3, 3, 3, 4, 4,3, 3) operations each.

KORBAA98b This *Korbaa (1998)* [94] problem contains 2 Jobs (A, B) with (9, 14) operations each. This problem contains 2 Jobs×9 Machines.

FT06 An instance from *Fisher et. al. (1963)* [45] consists of 6 Jobs×6 Machines. The processing times for the operations are in the range

of [1, 99]. Each Job (A, B, C, D, E, F) has 6 operations each.

LA04 The instances from *Lawrence (1984)* [102] consists of 10 Jobs$\times$5 Machines. The Jobs (A, B, C, D, E, F, G,H, I,J) have 5 operations each.

In all the above problems, the objective is to find the optimum cyclic schedule with minimum Work in progress (WIP). We performed all the experiments to analyze the following:

- Performance analysis between the Recurrent Neural Network (RNN), Lagrangian Relaxation Neural Network (LRNN) and Advanced Hopfield Network approaches in solving the CFMS problems.
- Speed or computational time to obtain optimal solutions for the RNN, LRRNN and Advanced Hopfield Network.
- The accuracy of solutions found against known optimal results and other known techniques such as Genetic Algorithm approach,

To accurately compare our three approaches of RNN, LRRNN and Advanced Hopfield, some assumptions and set-up were agreed for all three models prior to running the experimental problems. These are:

- No time restrictions on the simulation run.
- Restriction of maximum iteration of 15e51500000 iterations for Problems HIL87, HIL88, VAL94, OHL95, KORBAA98a, KORBAA98b. However due to the complexity of problems FT06 and LA04, a larger maximum iteration is imposed, which is 10e8 (i.e. 100000000) iterations.
- All the cyclic FMS test problems will have a minimum cycle time imposed as hard constraints, determined from the bottleneck machine.
- Application of Modified Competitive Dispatch Rule Phase (MCDRP) for generating the initial solutions for all three approaches.

The RNN, LRRNN and Advanced Hopfield approaches were developed on a JAVA program on 2GHz Intel Pentium 4 processor, 512Mb memory computer running a Linux (i.e Fedora Core 6) operating system.

FMS Benchmarks									
Problem	N	M	O	CT	Feasible solutions	$W.I.P._{LB}$	$W.I.P._{best}$ RNN	$W.I.P._{best}$ LRRNN	$W.I.P._{best}$ AdvHopfield
HIL87	4	4	15	8	1.6E+7	5	5	5	5
HIL88	4	3	9	6	1.1E+4	5	5	5	5
VAL94	3	5	13	11	7.9E+7	5	6	5	5
OHL95	2	6	10	28	3.5E+10	4	5	5	5
KORBAA98a	7	9	23	24	8.6E+10	12	14	14	14
KORBAA98b	2	9	23	24	8.6E+10	9	12	10	9
FT63(FT06)	6	6	36	43	4.4E+38	7	9	7	7
LA84(LA04)	10	5	50	537	3.9E+71	10	12	12	10

Table 5.10: Solution for FMS Test Problems with Minimum W.I.P.

FMS Benchmarks												
Problem	$W.I.P._{LB}$	GA WIP_{best}	GA CPU_{best}	RNN WIP_{best}	RNN $Iteration_{Best}$	RNN CPU_{best}	LRRNN WIP_{best}	LRRNN $Iteration_{Best}$	LRRNN CPU_{best}	Advanced Hopfield WIP_{best}	Advanced Hopfield $Iteration_{Best}$	Advanced Hopfield CPU_{best}
HIL87	5	5	~1	5	4530	13.9	5	9654	31.8	5	9020	16.6
HIL88	5	5	~1	5	3490	10.7	5	7340	24.2	5	6530	12.3
VAL94	5	5	~1	6	13420	41.2	5	31069	102.3	5	9870	18.2
OHL95	4	5	~1	5	1320155	4056.0	5	288716	950.4	5	29098	53.5
KORBAA98a	12	13	~10	14	1327966	4080.0	14	391090	1287.4	14	366848	675.0
KORBAA98b	9	9	~60	12	1230322	3780.0	10	877174	2887.5	9	1020652	1878.0
FT63 (FT06)	7	8	~48hours	9	45111803	138600.0	7	37183062	122400.0	7	44152174	81240.0
LA84 (LA04)	10	12	~72hours	12	60930228	187200.0	12	56300635	185331.6	10	94558000	173880.0

Table 5.11: Comparison of Solutions for FMS test problems

5.7 Results for Cyclic Flexible Manufacturing Systems Scheduling Problems

Table 5.10 summarized the best results found where for problems, the corresponding number of jobs N, machines M, operations O, and minimum cycle time is shown. Table 5.11 analyzes and compare the computational times in which all three approaches managed to find the best optimum results against the Genetic Algorithm approach from *Hsu et. al. (2007)* [78].

The total number of feasible solutions for each problem was calculated by *Hsu et. al. (2007)* [78] from Equation 3.43 and 3.44.

The results show that the Advanced Hopfield approach is the most effective among all three approaches. This is evident as the Advanced Hopfield approach was able to obtain best results (that is equivalent to the optimum lower bound result) in 6 out of the 8 problems tested. The second most effective approach which is the LRRNN approach managed to find the optimum solutions for 4 out of the 8 problems. Where else, the RNN approach only managed to be successful in two smaller problems of HIL87 and HIL88. Compared to the alternative Genetic Algorithm approach, only the Advanced Hopfield approach managed to outperform this particular

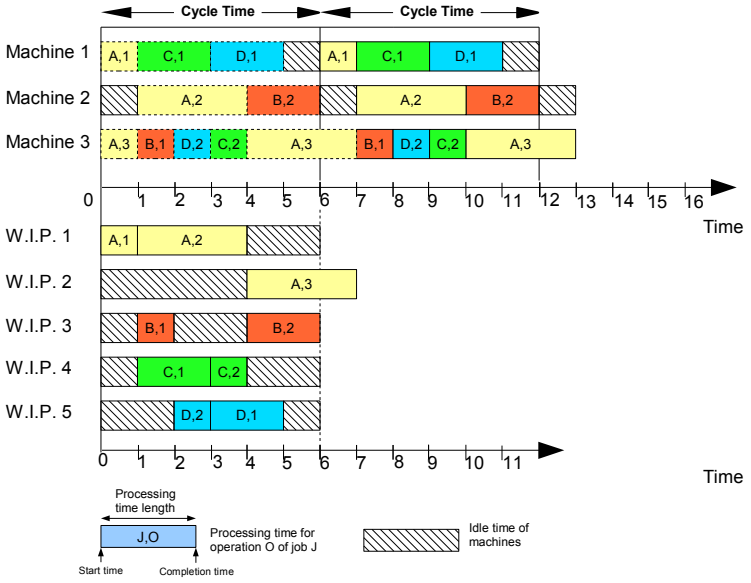


Figure 5.4: Solution for CFMS problem HIL88 [71] from Advanced Hopfield Approach.

approach.

Without restricting time limit on the iterations of all three approaches, this decision proved vital when we analyzed the individual approaches in more detail. This is evident as the Advanced Hopfield managed to solve the large and complex FT06 and LA04 problems only after over 48 hours had elapsed. This could be influenced by the fact that both test problems had 4.4×10^38 and 3.9×10^71 possible feasible solutions. Surprisingly, the LRRNN approach is able to obtain the minimum WIP of value 7 for problem FT06 after over 37×10^6 iterations.

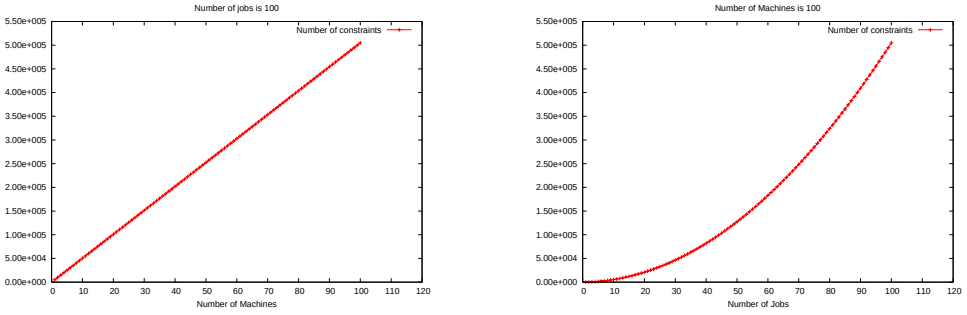
Figure 5.4 shows the schedule with minimum WIP of 5 obtained from the Advanced Hopfield approach, after 6530 iterations.

5.8 Discussion of Results

In this chapter, we have introduced several test problems from cyclic job shop scheduling problems modified from acyclic job shop. Based on these test problems, we utilized the Recurrent Neural Network (RNN) and Lagrangian Relaxation Neural Network (LRRNN) approaches in simulating and solving these problems. We then analyzed the results obtained based on the three criteria of accuracy of

results, speed of computations and performance between these two approaches.

We have concluded that the RNN and LRRNN approaches are able to solve cyclic job shop scheduling problems with up to 100 operations. There are however some concerns when classifying cyclic job shop with 100 operations as this may vary in terms of number of jobs or number of machines. As seen in Figure 5.5, for a fixed number of machines in a cyclic job shop, the number of constraints compares less than for a fixed number of jobs when number of machines increases. Hence could means less complexity for a small job large machine compared to large job, small machines type of cyclic job shop problems.



(a) For 100 Jobs (b) For 100 Machines

Figure 5.5: Graph showing number of constraints for a n jobs m machine job shop.

The RNN and LRRNN performed poorly for problems with more than 200 operations due to the complexity of constraints involved. The constraints in the problems are a combination of conjunctive constraints and disjunctive constraints. Overall, in solving the cyclic job shop scheduling problem, we have found that the RNN is faster in computing the optimum solution compared to the LRRNN approach. Although with this increased computational effort required, the LRRNN performed better for larger problems.

In the case of cyclic flexible manufacturing systems scheduling problems, we presented several test problems from FMS literature in order to analyze our three approaches of RNN, LRRNN and Advanced Hopfield. Based on a feasible initial schedule generated by our Modified Competitive Dispatch Rule Phase, and maximum iter-

ation restriction, our approaches were found to be effective in solving the problems. The Advanced Hopfield approach clearly outperform the RNN and LRRNN approaches. We have found that, without time restriction, the Advanced Hopfield approach has the potential to search a large part of the search space before stabilizing at a minimum energy level. There could be the factor of how modelling the cyclic schedule was done, that may hold an advantage over the LRRNN model used. However when comparing between recurrent networks, the use of changing weights in the form of Lagrange multiplier versus fixed weights in the case of RNN, was able to help LRRNN pursue a more optimum solution.

Part IV

Discussions

CHAPTER

SIX

Conclusions

6.1 Outcomes of The Thesis

In this concluding chapter, we will summarize the main points covered in this thesis, while analyzing the critical properties of all our approaches when solving the cyclic scheduling problems. We will also present suggestions for future research work.

In the first part of this thesis, we described the general scheduling problem. The scheduling problems depend on the type of manufacturing environment being considered. The different variations of these manufacturing environments hold their own advantages in dealing with the different products being manufactured and operations being processed. As such, each task or operation, or machines has their own characteristics and constraints.

We then reviewed the cyclic scheduling problems where we highlight the advantages of the cyclic scheduling against the non-cyclic scheduling approach. A review on the evolution of the cyclic scheduling problem as a whole was presented that included research literature that have worked on this type of scheduling problem. We also surveyed in terms of the types of problems in existence and their associated complexities, hence presenting a case for the reason our research is based on analyzing and solving the NP-Hard cyclic job shop and flexible manufacturing systems scheduling problem.

We then described in Chapter 3, the modelling techniques used in scheduling problems. The accurate modelling of the scheduling problem must involve constraints in the system and the objective for which the system is to be optimized for. We also summarized from complexity research literature that the scheduling problems

are generally NP-Hard problems and are difficult to solve in polynomial time with conventional optimization approaches.

In Section 3.5, we proposed the modelling of the cyclic job shop scheduling problem. Here the model showed how the cycle time can be calculated, with the inclusion of the constraints between operations of the job and constraints that the machines must adhere to. The cyclic job shop problem was modelled based on the assumption that generated schedule would be repeated into a cyclic schedule based on the optimum cycle time. Our research focused on minimizing the cycle time associated with the cyclic job shop (compared to the more common minimizing tardiness or maximizing the makespan) and finally presenting our modelling approach in a linear programming form.

Extending the cyclic scheduling further into the flexible manufacturing systems, we studied and modelled the scheduling problem in Section 3.6 that described the cyclic schedule based on the minimal cycle time constraint. The model included a function to accurately calculate the number of Work in progress (WIP) in the schedule and used this criteria as the objective function.

The common types of neural network and their use as an effective optimization approach suitable for solving the cyclic scheduling problem has been analyzed in Section 3.8. The evolution of neural networks that were successfully used by researchers in solving the scheduling problem fall into 3 main groups, Hopfield type networks, competitive networks and the backpropagation network.

In order to solve the cyclic scheduling problem using the models developed, we propose the Recurrent Neural Network (RNN) approach. This RNN approach finds the optimum solution by minimizing the energy state of the network and is based on the steepest descent method. We then extended this RNN technique into the Lagrangian Relaxation Neural Network (LRRNN) approach where the abilities of the recurrent neural network are enhanced with the Lagrange multipliers. One of the major characteristics of this LRRNN is the relaxation of the constraints of the cyclic scheduling problem using Lagrange Relaxation, which reduces the complexity of the problem.

We then extended our study of the cyclic job shop problem into the

linear constraints version of the problem. Based on the research found from *Munier (1996)* [124], *Hanen (2008)* [67], we presented a Delinearization algorithm that is able to transform the linear precedence constraints in the cyclic job shop into their equivalent uniform form. Due to a lack of test problems specific in this case, we managed to evaluate this approach on several test problems from *Dupas (2001)* [40]. We show proof that our approach of delinerization and solving the problem using our LRRNN approach is as competitive as the Genetic Algorithm, Constraint Satisfaction Problem (CSP) and Constraint Optimization Problem (COP) approaches.

In order to improve our approach in solving the cyclic flexible manufacturing systems scheduling problem, we developed the Advanced Hopfield Network approach. This particular approach was mapped from the model that represented the cyclic schedule in a 2D matrix and using the Schedule Generating Algorithm to generate the cyclic FMS schedule. Similar to the RNN and LRRNN approaches, this Advanced Hopfield network approach iteratively finds the optimum solution by minimizing the energy function of the network.

Finally from the experimental simulation results, our RNN, LRRNN and Advanced Hopfield approaches were evaluated against benchmark problems that showed the potential and the effectiveness of our approaches, not only in terms of the ability to optimize to a minimum state but also within a competitive computational time period.

The critical features of our three approaches can be summarized as follows:

1. The optimum schedules were obtained from the minimum energy state of the recurrent neural network,
2. The modelling approach developed of the cyclic scheduling problem effectively included cost function and the penalty factor effect on the problem.
3. Our approaches considered constraints violations of generated solutions to compensate for un-beneficial direction of the search path,
4. Constantly adjusting generation on schedule thus reflecting a cyclic generation ability based on cyclic constraint adjustments,

5. Always ensuring that feasibility of the final optimum solution obtained through the application of the postprocessing phase,
6. Placements of operations within generated schedules is progressive (far from complete randomness) and allows for interaction of all operations placement has on effect on the overall schedule rather than individual effect,
7. The characteristic of recovering the schedule from being captured in local minima state. This is possible through a sensitivity measure of the system state (e.g. energy value stuck in range on steady value) that will allow for activation of perturbation algorithm), transforming the energy state of the system and kick starting the search path,
8. The cost function modelled included a learning ability of the system as the system goes fourth in the search path. This allows for more accurate adjustments to solutions formulations/searches,
9. The Modified/Competitive Dispatch Rule Phase (CDRP or MCDRP) will encapsulate and reduce search space that is unbeneficial prior to employment of the recurrent neural network approaches, thus reducing computational time associated with unrealistic search paths.

Our main contributions through this research work can be summarized as follows:

- Analyzed the overall cyclic scheduling problem.
- Effectively modelled the cyclic job shop scheduling problem.
- Presented the recurrent neural network and Lagrangian relaxation neural network as viable, effective approaches in solving the cyclic job shop scheduling problem due to the lack of neural networks research as an approach in solving the cyclic scheduling problem.
- Effectively modelled the cyclic flexible manufacturing system scheduling problems.
- Proposed the advanced Hopfield network, combined with a modified competitive dispatch rule phase (MCDRP) in effectively solving the cyclic flexible manufacturing system scheduling problem.

- Modelled, delinearized with the Delinearization Algorithm and presented a LRRNN approach to the cyclic job shop problem with linear precedence constraints problem.

This research has focused on the cyclic scheduling problem itself. We focused on a modelling approach that can accurately depict the problem and allow that model to be solved by the recurrent neural network approaches.

6.2 Suggestion for Future Research

As this thesis has shown from the experimental results on cyclic job shop and cyclic flexible manufacturing system scheduling problems that our neural network techniques are effective in solving these cyclic problems, an extension of this research would involve researching to solve other types of cyclic scheduling problems that may include robotics problems, hoist problems, cyclic flow shop problem or cyclic open shop problem. This will greatly enhance the application of our RNN, LRRNN and Advanced Hopfield further into the field of cyclic scheduling problems.

Another possible extension of our neural network approaches is research into the sensitivity of our neural network approach to stochasticity in cyclic scheduling problems. Based on machine related issues (e.g. machines with bottleneck, linear, maintenance related issues, non-uniform speed) or load variation on production lines, the modelling of these issues would allow for our approach to move a step closer to modelling real-life cyclic scheduling problems.

One areas of future research that could improve the computational time of our neural network approaches would be the inclusion of other optimization approaches. These optimization approaches that could include tabu search, swarm optimization, fuzzy logic, petri net or genetic algorithm, could be allowed to work in parallel with the neural network approaches without jeopardizing the efficiency to obtain optimal solutions. Hence these hybrid approaches will be able to combine the strengths from each of the infused optimization approach in tackling the combinatorial problem. One other method to reduce the computational time in solving the scheduling problems could be by using the parallelism approach of agents. These

agents would be be able to represent the dynamics of the neural networks.

As our RNN, LRRNN and Advanced Hopfield approaches were developed in a JAVA program, the advancements in the form of hardware, web based online scheduling and real-time cyclic scheduling approaches would be an interesting aspect to look into. These integrations may also allow for user-modifiable characteristics and expert inputs into neural network models.

BIBLIOGRAPHY

[1] I.N.K. Abadi, N.G. Hall, and C. Sriskandarajah. Minimizing cycle time in a blocking flowshop. *Operations Research*, 48(1):177–180, 2000.

[2] J Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3):391–401, 1988.

[3] S.V.B. Aiyer, M. Niranjan, and F. Fallside. A theoretical investigation into the performance of the Hopfield model. *Neural Networks, IEEE Transactions on*, 1(2):204–215, June 1990.

[4] T. Çakar and I. Cil. Artificial neural networks for design of manufacturing systems and selection of priority rules. *International Journal of Computer Integrated Manufacturing*, 17(3):p195 – 211, April 2004.

[5] D.E. Akyol and G.M. Bayhan. A coupled gradient network approach for the multi-machine earliness and tardiness scheduling problem. In *Computational Science and Its Applications - ICCSA 2005*, volume 3483/2005 of *Lecture Notes in Computer Science*, pages 596–605. Springer Berlin / Heidelberg, May 2005.

[6] D.E. Akyol and G.M. Bayhan. Multi-machine earliness and tardiness scheduling problem: an interconnected neural network approach. *The International Journal of Advanced Manufacturing Technology*, 2007.

[7] S. Albers and G. Schmidt. Scheduling with unexpected machine breakdowns. *Discrete Applied Mathematics*, 110(2-3):85–99, 2001.

[8] A. Allahverdi and T. Aldowaisan. No-wait flowshops with bicriteria of makespan and maximum lateness. *European Journal of Operational Research*, 152(1):132–147, January 2004.

[9] A. Allahverdi, J.N.D. Gupta, and T. Aldowaisan. A review of scheduling research involving setup considerations. *Omega*, 27(2):219–239, April 1999.

[10] D. Applegate and W. Cook. A computational study of the job-shop scheduling problem. *ORSA Journal on Computing*, 3(2):149–156, 1991.

[11] I. Arizono, A. Yamamoto, and H. Ohta. Scheduling for minimizing total actual flow time by neural networks. *International Journal of Production Research*, 30(2):503–511, 1992.

[12] E. Balas. Machine sequencing via disjunctive graphs: An implicit enumeration algorithm. *Operations Research*, 17(6):941–957, November - December 1969.

[13] P. Baptiste and C.L. Pape. Disjunctive constraints for manufacturing scheduling: Principles and extensions. *International Journal of Computer Integrated Manufacturing*, 9(4):306–310, July 1996.

[14] C. Bierwirth and D.C. Mattfeld. Production scheduling and rescheduling with genetic algorithms. *Evolutionary Computation*, 7(1):1–17, 1999.

[15] O. Bilkay, O. Anlagan, and S.E. Kilic. Job shop scheduling using fuzzy logic. *The International Journal of Advanced Manufacturing Technology*, 23(7):606–619, 2004.

[16] J. Blazewicz, W. Domschke, and E. Pesch. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93(1):1–33, August 1996.

[17] J. Blazewicz and D. Kobler. Review of properties of different precedence graphs for scheduling problems. *European Journal of Operational Research*, 142(3):435–443, 2002.

[18] F. Boussemart, G. Cavory, and C. Lecoutre. Solving the cyclic job shop scheduling problem with linear precedence constraints using CP techniques. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 7, 6-9 October 2002.

[19] P. Brucker and T. Kampmeyer. Tabu search algorithms for cyclic machine scheduling problems. *Journal of Scheduling*, 8(4):303–322, 2005.

[20] P. Brucker and T. Kampmeyer. *Operations Research Proceedings 2005*, chapter Cyclic Scheduling Problems with Linear Precedences and Resource Constraints, pages 667–672. Operations Research Proceedings. Springer Berlin Heidelberg, 2006.

[21] P. Brucker, Y.N. Sotskov, and F. Werner. Complexity of shop-scheduling problems with fixed number of jobs: a survey. *Mathematical Methods of Operations Research*, 65(3):461–481, June 2007.

[22] Peter Brucker. *Scheduling Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[23] Peter Brucker and Sigrid Knust. Complexity results for scheduling problems. http://www.mathematik.uni-osnabrueck.de/research/OR/class/.

[24] J. Campos, G. Chiola, J.M. Colom, and M. Silva. Properties and performance bounds for timed marked graphs. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on [see also Circuits and Systems I: Regular Papers, IEEE Transactions on]*, 39(5):386–401, May 1992.

[25] S. Cavalieri. On the use of an enhanced Hopfield neural model to solve FMS performance optimization problem. *Applied Intelligence*, 8(2):123–138, 1998.

[26] S. Cavalieri, S. Terzi, and M. Macchi. A benchmarking service for the evaluation and comparison of scheduling techniques. *Computers in Industry*, 58(7):656–666, September 2007.

[27] G. Cavory, R. Dupas, and G. Goncalves. A genetic approach to solving the problem of cyclic job shop scheduling with linear constraints. *European Journal of Operational Research*, 161(1):73–85, 2005.

[28] I. Chaieb, 0. Korbaa, and H. Camus. Short term planning of cyclic production in FMSs. In *29th International Conference on Computers and Industrial Engineering, Montreal*, 2001.

[29] P. C. Chang. A branch and bound approach for single machine scheduling with earliness and tardiness penalties. *Computers & Mathematics with Applications*, 37(10):133–144, May 1999.

[30] M. Chen and Y. Dong. Applications of neural networks to solving SMT scheduling problems a case study. *International Journal of Production Research*, 37:4007–4020, November 1999.

[31] R.M. Chen and Y.M. Huang. Competitive neural network to solve scheduling problems. *Neurocomputing*, 37(1):177–196, 2001.

[32] P. Chretienne. The basic cyclic scheduling problem with deadlines. *Discrete Applied Mathematics*, 30(2-3):109–123, 1991.

[33] P. Chrétienne, E.G. Coffman, J.K. Lenstra, and Z. Liu. *Scheduling theory and its applications*. John Wiley & Sons New York, 1995.

[34] G. Chryssolouris, M. Lee, and M. Domroese. The use of neural networks in determining operational policies for manufacturing systems. *Journal of Manufacturing Systems*, 10(2):166–175, 1991.

[35] A. Cichocki, R. Unbehauen, K. Weinzierl, and R. Holzel. A new neural network for solving linear programming problems. *European Journal of Operational Research*, 93(2):244–256, September 1996.

[36] R.W. Conway, W.L. Maxwell, and L.W. Miller. *Theory of Scheduling*. Addison-Wesley, Reading, MA, 1967.

[37] C. Darken and J Moody. Note on learning rate schedules for stochastic optimization. In *NIPS-3: Proceedings of the 1990 conference on Advances in neural information processing systems 3*,

pages 832–838, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[38] M.W. Dawande, N.H. Geismar, and S.P. Sethi. Dominance of cyclic solutions and challenges in the scheduling of robotic cells. *SIAM Review*, 47(4):709–721, 2005.

[39] D. Draper, A.K. Jónsson, D.P. Clements, and D. Joslin. Cyclic scheduling. In *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI '99)*, pages 1016–1021, San Francisco, CA, USA, 1999.

[40] R. Dupas. Benchmarks of cyclic job shop with linear constraints. http://www.iut-bethune.univ-artois.fr/dupas/benchCyclique/indexBench.html.

[41] L. Fang and T. Li. Design of competition-based neural networks for combinatorial optimization. *International Journal of Neural Systems*, 1(3):221–235, 1990.

[42] L. Fang, P.B. Luh, and H. Chan. Improving the Lagrangian relaxation approach for large job-shop scheduling. *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, 1:552–557, 2000.

[43] S. Feng, L. Li, L. Cen, and J. Huang. Using mlp networks to design a production scheduling system. *Computers and Operations Research*, 30(6):821–832, 2003.

[44] A. Fink and S. Vosz. Solving the continuous flow-shop scheduling problem by metaheuristics. *European Journal of Operational Research*, 151(2):400–414, December 2003.

[45] H. Fisher and G. L. Thompson. Probabilistic learning combinations of local job-shop scheduling rules. In J.F. Muth and G.L. Thompson, editors, *Industrial Scheduling*, pages 225–251. Prentice Hall, 1963.

[46] M. L. Fisher. The Lagrangian Relaxation method for solving integer programming problems. *Management Science*, 50(12 Supplement):1861–1871, 2004.

[47] Y.P.S. Foo and T. Takefuji. Stochastic neural networks for solving job-shop scheduling. I. Problem representation. In *IEEE 2nd International Conference on Neural Networks, 24-27 July*, volume 2, pages 275–282, 1988.

[48] Y.P.S. Foo and T. Takefuji. Stochastic neural networks for solving job-shop scheduling. II. Architecture and simulations. In *IEEE 2nd International Conference on Neural Networks, 24-27 July*, volume 2, pages 283–290, 1988.

[49] Y.P.S. Foo and T. Takefuji. Integer linear programming neural networks for job-shop scheduling. *Neural Networks, 1988., IEEE International Conference on*, 2:341–348, 24-27 Jul 1988.

[50] S. French. *Sequencing and Scheduling - An Introduction to the Mathematics of the Job-Shop*. Ellis Horwood, John-Wiley & Sons, New York, 1982.

[51] H.L. Gantt. *Organizing for Work*. Harcourt, Brace and Howe, New York., 1919.

[52] M.P. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[53] M.R. Garey, D.S. Johnson, and R. Sethi. The complexity of &ow shop and job shop scheduling. *Mathematics of Operations Research*, 1(2):117–129, May 1976.

[54] F. Glover. Tabu search– part i. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[55] F. Glover. Tabu search– part ii. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[56] J.F. Gonçalves, J.J. de Magalhães Mendes, and M.G.C. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167(1):77–95, 2005.

[57] T. Gonzalez and S. Sahni. Flowshop and jobshop schedules: Complexity and approximation. *Operations Research*, 26(1):36–52, January-February 1978.

[58] M. Gourgand, N. Grangeon, and S. Norre. A contribution to the stochastic flow shop scheduling problem. *European Journal of Operational Research*, 151(2):415–433, December 2003.

[59] R. Graham, E. Lawlerm, E. Lenstra, and K. Rinnooy. Optimization and approximation in determenistic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[60] H. Gultekin, M.S. Akturk, and O.E. Karasan. Cyclic scheduling of a 2-machine robotic cell with tooling constraints. *European Journal of Operational Research*, 174(2):777–796, 2006.

[61] L.G. Hall, T.E. Lee, and M.E. Posner. Periodic shop scheduling problems. Technical Report 97–25, Dept of Industrial Engineering, KAIST, Taejeon, Korea., 1997.

[62] N.G. Hall, T.-L. Lee, and M.E. Posner. The complexity of cyclic shop scheduling problems. *Journal of Scheduling*, 5(4):307–327, 2002.

[63] R.W. Hall. Cyclic scheduling for improvement. *International journal of production research*, 26(3):457–472, March 1987.

[64] C. Hanen. Study of a NP-hard cyclic scheduling problem: The recurrent job-shop. *European Journal of Operational Research*, 72(1):82–101, January 1994.

[65] C. Hanen and A. Munier. Cyclic scheduling on parallel processors: an overview. In P. Chrétienne, E. G. Coffman, J. K. Lenstra, and Z. Liu, editors, *Scheduling Theory and its Applications*, pages 193–226. John Wiley & Sons, 1995.

[66] C. Hanen and A. Munier. A study of the cyclic scheduling problem on parallel processors. *Discrete Applied Mathematics*, 57(2-3):167–192, 1995.

[67] C. Hanen and A. Munier-Kordon. Periodic schedules for linear precedence constraints. *Discrete Applied Mathematics*, In Press, Corrected Proof, Available online 19 May 2008.

[68] Z. Hanzalek. Petri net based cyclic scheduling. In *3rd IEEE European Workshop on Computer Intensive Methods in Control and Data Processing, UTIA CAV & University of Reading, Prague 1998.*, 1998.

[69] J.W. Herrmann. Cyclic production systems: Scheduling and performance tradeoffs. In *12th Annual Industrial Engineering Research Conference (IERC 2003) Portland, OR, May 18-20, 2003*, May 2003.

[70] H. P. Hillion, J.-P. Proth, and X.-L. Xie. A heuristic algorithm for the periodic scheduling and sequencing job-shop problem. *Decision and Control, 1987. 26th IEEE Conference on*, 26:612–617, December 1987.

[71] H.P. Hillion and J.M. Proth. Analyse de fabrications non linéaires et répétitives à l'aide des graphes d'evénements temporisés. *RAIRO*, 22(2), September 1988.

[72] H.P. Hillion and J.M. Proth. Performance evaluation of job-shop systems using timed event-graphs. *IEEE Transactions on Automatic Control*, 34(1):3–9, 1989.

[73] K.L. Hitz. Scheduling of flexible flow shops. Technical report, MIT, Cambridge, MA,, 1979.

[74] J. A. Hoogeveen and S. L. van de Velde. Stronger lagrangian bounds by use of slack variables: Applications to machine scheduling problems. *Mathematical Programming*, 70:173–190, 1995.

[75] J.J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the United States of America*, 81(10):3088–3092, May 1984.

[76] J.J. Hopfield and D.W. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:1–25, 1985.

[77] T. Hsu, R. Dupas, and G. Goncalves. A genetic algorithm to solving the problem of flexible manufacturing system cyclic scheduling. In *IEEE International Conference on Systems, Man and Cybernetics*, October 2002.

[78] T. Hsu, O. Korbaa, R. Dupas, and G. Goncalves. Cyclic scheduling for F.M.S.: Modelling and evolutionary solving approach. *European Journal of Operational Research*, 191(2):464–484, 1 December 2008.

[79] S.H. Huang and H.-C. Zhang. Neural networks in manufacturing: A survey. *Fifteenth IEEE/CHMT International on Elec-*

tronic Manufacturing Technology Symposium, 1993,, pages 177–190, October 1993.

[80] Y.M. Huang and R.M. Chen. Scheduling multiprocessor job with resource and timing constraints using neural networks. *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, 29(4):490–502, 1999.

[81] J. Hurink, B. Jurisch, and M. Thole. Tabu search for the job-shop scheduling problem with multi-purpose machines. *OR Spectrum*, 15(4):205–215, 1994.

[82] J.R. Jackson. An extension of johnson's results on job lot scheduling. *Naval Research Logistics Quarterly*, 3:201–203, 1956.

[83] A.K. Jain and H.A. Elmaraghy. Production scheduling/rescheduling in flexible manufacturing. *International Journal of Production Research*, 35(1):281–309, January 1997.

[84] A.S. Jain and S. Meeran. Job shop scheduling using neural networks. *International Journal of Production Research*, 36(5):1249–1272, 1998.

[85] A. Jones and L. C. Rabelo. Survey of job shop scheduling techniques. NISTIR, National Institute of Standards and Technology, Gaithersburg, MD, 1998.

[86] B. Jurisch. *Scheduling jobs in shops with multi-purpose machines*. PhD thesis, Department of Mathematics/Informatics, Universitat Osnabrtick, 1992.

[87] B. Jurisch. Lower bounds for the job-shop scheduling problem on multi-purpose machines. *Discrete Applied Mathematics*, 58(2):145–156, 1995.

[88] J. Kamburowski. More on three-machine no-idle flow shops. *Computers and Industrial Engineering*, 46(3):461–466, June 2004.

[89] T. Kampmeyer. *Cyclic Scheduling Problems*. PhD thesis, University of Osnabruck, Germany, 2006.

[90] S. Karabati and P. Kouvelis. Cyclic scheduling in flow lines: Modeling observations, effective heuristics and a cycle time minimization procedure. *Naval Research Logistics*, 43(2):211–231, December 1998.

[91] S. Karabati and B. Tan. Stochastic cyclic scheduling problem in synchronous assembly and production lines. *The Journal of the Operational Research Society*, 49(11):1173–1187, November 1998.

[92] V. Kats and E. Levner. Cyclic scheduling in a robotic production line. *Journal of Scheduling*, 5(1):23–41, January 2002.

[93] T. Kohonen. The self-organizing map. *Neurocomputing*, 21(1–3):1–6, November 1998.

[94] 0. Korbaa. *Commande cyclique des systèmes flexibles de production manufacturière à l'aide des réseaux de Petri: de la planification*

à l'ordonnancement des régimes transitoires. Thèse de doctorat, Universit*é* de Lille1., 1998.

[95] O. Korbaa and H. Camus. Cyclic productions planning. *Computational Engineering in Systems Applications, IMACS Multiconference on*, pages 1110–1117, 2006.

[96] O. Korbaa, H. Camus, and J.-C. Gentina. Heuristic for the resolution of the general FMS cyclic scheduling problem. In *IEEE International Conference on Systems, Man, and Cybernetics, 1997. 'Computational Cybernetics and Simulation'.*, volume 3, pages 2903–2908, 1997.

[97] O. Korbaa, H. Camus, and J.-C. Gentina. Transient state study for cyclic schedules: bounds and optimization. In *IEEE International Symposium on Assembly and Task Planning (ISATP 97)*, pages 188–193, Marina del Rey, CA, USA, August 1997.

[98] O. Korbaa, H. Camus, and J.-C. Gentina. A new cyclic scheduling algorithm for flexible manufacturing systems. *International Journal of Flexible Manufacturing Systems*, 14:173–187, April 2002.

[99] M. Kubale and A. Nadolski. Chromatic scheduling in a cyclic open shop. *European Journal of Operational Research*, 164(3):585–591, August 2005.

[100] T. Ladhari and M. Haouari. A computational study of the permutation flow shop problem based on a tight lower bound. *Computer and Operations Research*, 32(7):1831–1847, July 2005.

[101] E.L. Lawler. Optimal sequencing of a single machine subject to precedence constraints. *Management Science*, 19(5):544–546, January 1973.

[102] S. Lawrence. *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement)*. Graduate School of Industrial Administration, Carnegie-Mellon University, 1984.

[103] J.-K. Lee. Benchmarking study of the cyclic scheduling analysis methods in FMS. *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, 3:6 pp., 6-9 Oct. 2002.

[104] J.-K. Lee and O. Korbaa. Scheduling analysis of FMS: an unfolding timed petri nets approach. *Mathematics and Computers in Simulation*, 70(5):419–432, 2006.

[105] J.-K. Lee, O. Korbaa, and J.-C. Gentina. Modeling and analysis of cycle schedule using petri nets unfolding. In *IEEE International Conference on Systems, Man, and Cybernetics*, 2001.

[106] T.-E. Lee and M.E. Posner. Performance measures and schedules in periodic job shops. *Operations Research*, 45(1):72–91, 1997.

[107] T.E. Lee and J.W. Seo. Stochastic cyclic flow lines: nonblocking, Markovian models. *Journal of the Operational Research Society*, 49(5):537–548, May 1998.

[108] J.K. Lenstra and A.H.G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1):22–35, January-February 1978.

[109] J.K. Lenstra and A.H.G. Rinnooy Kan. Computational complexity of discrete optimization problem. *Annals of Discrete Mathematics*, 4:121–140, 1979.

[110] J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.

[111] J.Y.-T. Leung, O. Vornberger, and J.D. Witthoff. On some variants of the bandwidth minimization problem. *SIAM Journal on Computing*, 13(3):650–667, 1984.

[112] G. Liansheng, S. Gang, and W. Shuchun. Intelligent scheduling model and algorithm for manufacturing. *Production Planning and Control*, 11:234–243(10), 1 April 2000.

[113] R. Linn and W. Zhang. Hybrid flow shop scheduling: A survey. *Computers & Industrial Engineering*, 37(1–2):57–61, October 1999.

[114] P.B. Luh and D.J. Hoitomt. Scheduling of manufacturing systems using the Lagrangian relaxation technique. *IEEE Transactions on Automatic Control*, 38(6):1066–1080, July 1993.

[115] P.B. Luh, X. Zhao, Y. Wang, and L.S. Thakur. Lagrangian relaxation neural networks for job shop scheduling. *Robotics and Automation, IEEE Transactions on*, 16(1):78–88, 2000.

[116] R. Maheswaran, S.G. Ponnambalam, D.N. Samuel, and A.S. Ramkumar. Hopfield neural network approach for single machine scheduling problem. *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, 2:850–854, 2004.

[117] O. Marchetti and A. Munier-Kordon. A polynomial algorithm for a bi-criteria cyclic scheduling problem. In *The 25th Workshop of the UK PLANNING AND SCHEDULING Special Interest Group, (PlanSIG 2006), 14th - 15th December, 2006, ENGLAND*, December 2006.

[118] D.C. Mattfeld and C. Bierwirth. An efficient genetic algorithm for job shop scheduling with tardiness objectives. *European Journal of Operational Research*, 155(3):616–630, 2004.

[119] G. McMahon and M. Florian. On scheduling with ready times and due dates to minimize maximum lateness. *Operations Research*, 23(3):475–482, May–June 1975.

[120] P.R. McMullen. A Kohonen self-organizing map approach to addressing a multiple objective, mixed-model jit sequencing problem. *International Journal of Production Economics*, 72(1):59–71, 2001.

[121] K. Metaxiotis and J. Psarras. Neural networks in production scheduling: Intelligent solutions and future promises. *Applied Artificial Intelligence*, 17(4):361–373, 2003.

[122] M. Middendorf and V.G. Timkovsky. On scheduling cycle shops: classification, complexity and approximation. *Journal of Scheduling*, 5(2):135–169, March 2002.

[123] S.A. Munawar, M. Bhushan, R.D. Gudi, and A.M. Belliappa. Cyclic scheduling of continuous multiproduct plants in a hybrid flowshop facility. *Industrial And Engineering Chemistry Research*, 42(23):5861–5882, 2003.

[124] A. Munier. The basic cyclic scheduling problem with linear precedence constraints. *Discrete Applied Mathematics*, 64(3):219–238, February 1996.

[125] J. F. Muth and G. L. Thompson. *Industrial Scheduling*. Prentice-Hall, Inc., Englewood Cliffs, N. J., 1963.

[126] M. Nakamura, K. Hachiman, H. Tohme, T. Okazaki, and S. Tamaki. Evolutionary computing of petri net structure for cyclic job shop scheduling. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E89-A(11):3235–3243, November 2006.

[127] M. Nakamura, H. Tome, K. Hachiman, B.M. Ombuki, and K. Onaga. Cyclic job-shop-scheduling based on evolutionary petri nets. In *26th Annual Conference of the IEEE on Industrial Electronics Society (IECON 2000)*, 2000.

[128] E. Nowicki and C. Smutnicki. An advanced Tabu Search algorithm for the job shop problem. *Journal of Scheduling*, 8(2):145–159, 2005.

[129] H. Ohl, H. Camus, E. Castelain, and J.-C. Gentina. A heuristic algorithm for the computation of cyclic schedules andthe necessary WIP to obtain optimal cycle time. In *Fourth International Conference on Computer Integrated Manufacturing and Automation Technology, New York, USA*, October 1994.

[130] S.S. Panwalkar and W. Iskander. A survey of scheduling rules. *Operations Research*, 25(1):45–61, January - February 1977.

[131] B.J. Park, H.R. Choi, and H.S. Kim. A hybrid genetic algorithm for the job shop scheduling problems. *Computers & Industrial Engineering*, 45(4):597–613, 2003.

[132] M. Pinedo. *Scheduling Theory, Algorithms, and Systems*. Prentice Hall, second edition, 2002.

[133] J.-P. Proth. Scheduling: New trends in industrial environment. *Annual Reviews in Control*, 31(1):157–166, 2007.

[134] U.S. Rao and P.L. Jackson. Estimating performance measures in stochastic cyclic schedules. *IIE Transactions*, 28:929–939, 1996.

[135] Raúl Rojas. *Neural Networks: a systematic introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[136] R. Roundy. Cyclic schedules for job shops with identical jobs. *Mathematics of Operational Research*, 17(4):842–865, November 1992.

[137] I. Sabuncuoglu. Scheduling with neural networks: a review of the literature and new research directtions. *Production Planning and Control*, 9(1):2–12, 1998.

[138] I. Sabuncuoglu and B. Gurgun. A neural network model for scheduling problems. *European Journal of Operational Research*, 93(2):288–299, September 1996.

[139] I. Sabuncuoglu and S. Touhami. Simulation metamodelling with neural networks: an experimental investigation. *International Journal of Production Research*, 40(11):2483–2505, July 2002.

[140] T. Satake, K. Morikawa, and N. Nakamura. Neural network approach for minimizing the makespan of the general makespan. *International Journal of Production Economics*, 33(1-3):67–74, January 1994.

[141] J.-W. Seo and T.-T. Lee. Steady-state analysis and scheduling of cyclic job shops with overtaking. *International Journal of Flexible Manufacturing Systems*, 14(4):291–318, 2002.

[142] P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal Discrete Mathematics*, 2(4):550–581, 1989.

[143] J.S. Smith, B.A. Peters, and A. Srinivasan. Job shop scheduling considering material handling. *International Journal of Production Research*, 37(7):1541–1560, 1999.

[144] Y.N. Sotskov and N.V. Shakhlevich. NP-hardness of shop-scheduling problems with three jobs. *Discrete Applied Mathematics*, 59(3):237–266, 1995.

[145] R.H. Storer, S.W. Wu, and R. Vaccari. New search spaces for sequencing problems with application to job shop scheduling. *Management Science*, 38(10):1495–1509, 1992.

[146] L. Tang, H. Xuan, and J. Liu. A new Lagrangian relaxation algorithm for hybrid flowshop scheduling to minimize total weighted completion time. *Computers and Operations Research*, 33(11):3344–3359, November 2006.

[147] V. T'kindt, F. Della Croce, and C. Esswein. Revisiting branch and bound search strategies for machine scheduling problems. *Journal of Scheduling*, 7(6):429–440, 2004.

[148] A. Toguyeni and O. Korbaa. Indirect monitoring of the failures of a flexible manufacturing system under cyclic scheduling. *Robotics and Computer-Integrated Manufacturing*, 21(1):1–10, February 2005.

[149] B. Trouillet, A. Benasser, and J.-C. Gentina. Transformation of the cyclic scheduling problem of a large class of FMS into

the search of an optimized initial marking of a linearizable weighted t-system. In *Sixth International Workshop on Discrete Event Systems (WODES'02)*, page 83, Washington, DC, USA, 2002. IEEE Computer Society.

[150] B. Trouillet, O. Korbaa, and J.-C. Gentina. Formal approach of FMS cyclic scheduling. *Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(1):126–137, January 2007.

[151] C. Valentin. Modeling and analysis methods for a class of hybrid dynamic systems. In *Proccedings of the Symposium ADPM'94, Bruxelles, Belgique*, November 1994.

[152] W.-L. Wang, X.-L. Xu, and Q.-D. Wu. Hopfield neural networks approach for job shop scheduling problems. *Intelligent Control. 2003 IEEE International Symposium on*, pages 935–940, 2003.

[153] V.C.S. Wiers. A review of the applicability of OR and AI scheduling techniques in practice. *Omega, International Journal Management Science*, 25(2):145–153, 1997.

[154] T.M. Willems and J.E. Rooda. Neural networks for job-shop scheduling. *Control Engineering Practice*, 2(1):31–39, 1994.

[155] T. Witkowski, P. Antczak, and G. Strojny. The implementation of neural networks for the optimization of the production scheduling. *Proceedings of IEEE International Joint Conference on Neural Networks, 2004.*, 3, 2004.

[156] A. Yahyaoui and F. Fnaiech. Recent trends in intelligent job shop scheduling. In *First IEEE International Conference on E-Learning in Industrial Electronics*, pages 191–195, December 2006.

[157] T. Yamada and R. Nakano. Genetic algorithms for job-shop scheduling problems. *Proceedings of Modern Heuristic for Decision Support*, pages 67–81, 1997.

[158] S. Yang and D. Wang. Constraint satisfaction adaptive neural network and heuristics combined approaches for generalized job-shop scheduling. *Neural Networks, IEEE Transactions on*, 11(2):474–486, 2000.

[159] C.Y. Zhang, P.G. Li, Z.L. Guan, and Y.Q. Rao. A Tabu Search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers and Operations Research*, 34(11):3229–3242, 2007.

[160] H. Zhang and S.C. Graves. Cyclic scheduling in a stochastic environment. *Operations Research*, 45(6):894–903, November-December 1997.

[161] Y.I. Zhang, P.B. Luh, K. Yoneda, T. Kano, and Y. Kyoya. Mixed-model assembly line scheduling using the Lagrangian relaxation technique. *IIE Transactions*, 32(2):125–134, 2000.

[162] D.N. Zhou, V. Cherkassky, T.R. Baldwin, and D.E. Olson. A neural network approach to job-shop scheduling. *Neural Networks, IEEE Transactions on*, 2(1):175–179, January 1991.

APPENDIX

A

Appendix

Example of for evaluating the benchmarks for the cyclic flexible manufacturing systems problems comprises of:

Benchmark [HIL87] This *Hillion et. al. (1987)* [70] benchmarks consist of 4 Jobs×4 Machines. Jobs (A, B, C, D) require (4, 4, 3, 4) operations each.

A description of the benchmark is as follows:

- Job A: (A,1),M1,1; (A,2),M2,4; (A,3),M3,3; (A,4),M4,3.
- Job B: (B,1),M4,1; (B,2),M2,2; (B,3),M1,3; (B,4),M3,1.
- Job C: (C,1),M1,2; (C,2),M3,1; (C,3),M4,1.
- Job D: (D,1),M3,3; (D,2),M1,2; (D,3),M2,1; (D,4),M4,2.

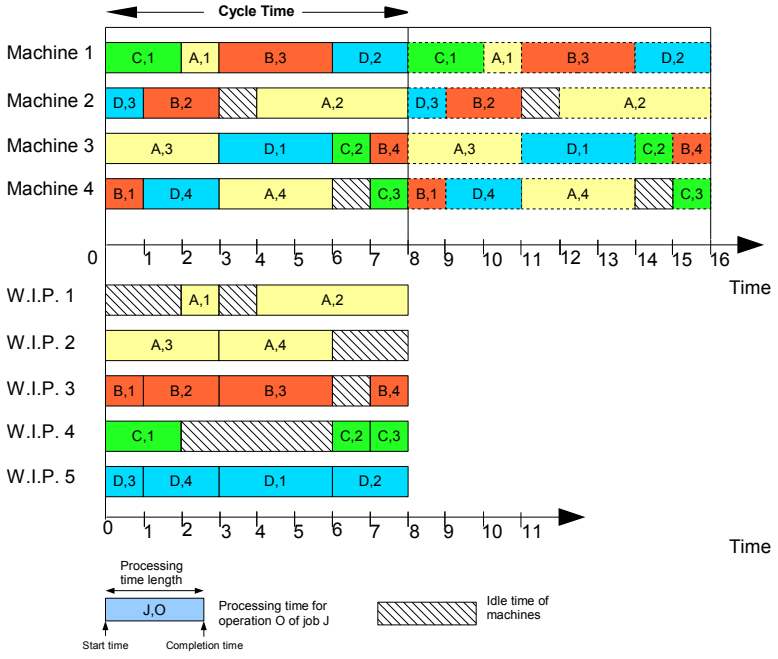


Figure A.1: Solutions for HIL87 Benchmark.

The dual gantt chart in Figure A.1 shows the solution found for the benchmark.

Benchmark [HIL88] This 4 Jobs×3 Machines problem comes from *Hillion and Proth (1988)* [71] where Jobs (A, B, C, D) require (3, 2, 2, 2) operations each.

A description of the benchmark is as follows:

- Job A: (A,1),M1,1; (A,2),M2,3; (A,3),M3,3.
- Job B: (B,1),M3,1; (B,2),M2,2.
- Job C: (C,1),M1,2; (C,2),M3,1.
- Job D: (D,1),M1,2; (D,2),M3,1.

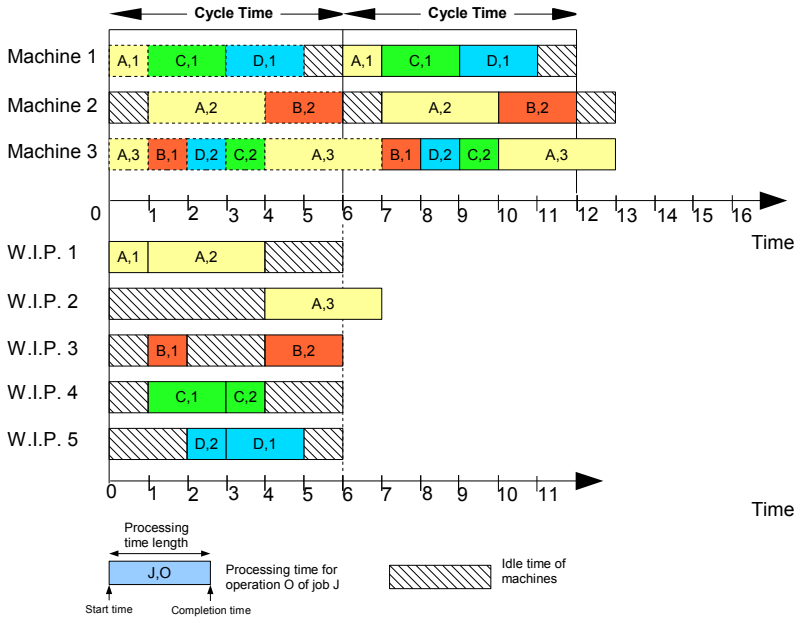


Figure A.2: Solutions for HIL88 Benchmark.

The dual gantt chart in Figure A.2 shows the solution found for the benchmark.

Benchmark [VAL94] *Valentin (1994)* [151] generated a 5 Jobs×3 Machines cyclic FMS problem. The Jobs (A, B, C, D, E) require (3, 3, 3, 2, 2) operations each.

A description of the benchmark is as follows:

- Job A: (A,1),M1,2; (A,2),M2,3; (A,3),M3,2.
- Job B: (B,1),M1,2; (B,2),M2,3; (B,3),M3,2.
- Job C: (C,1),M1,2; (C,2),M2,3; (C,3),M3,2.
- Job D: (D,1),M2,1; (D,2),M1,2.
- Job E: (E,1),M2,1; (E,2),M1,2.

The dual gantt chart in figure A.3 shows the solution found for the benchmark.

Benchmark [OHL95] The *Ohl et. al. (1995)* [129] problem consists of 2 Jobs×6 Machines. Jobs (A, B) have (6, 4) operations each.

A description of the benchmark is as follows:

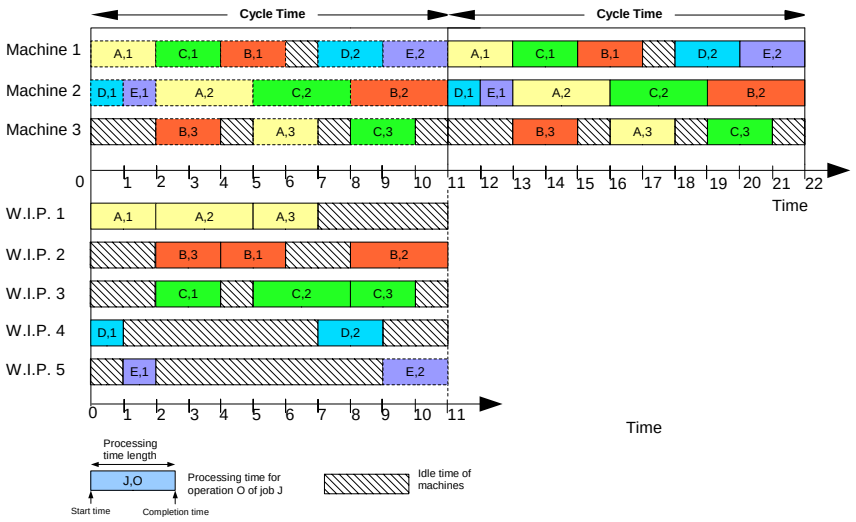


Figure A.3: Solutions for VAL94 Benchmark.

- Job A: (A,1),M1,18; (A,2),M3,14; (A,3),M2,12; (A,4),M3,14; (A,5),M2,10; (A,6),M4,14.
- Job B: (B,1),M5,5; (B,2),M6,4; (B,3),M5,5; (B,4),M6,4.

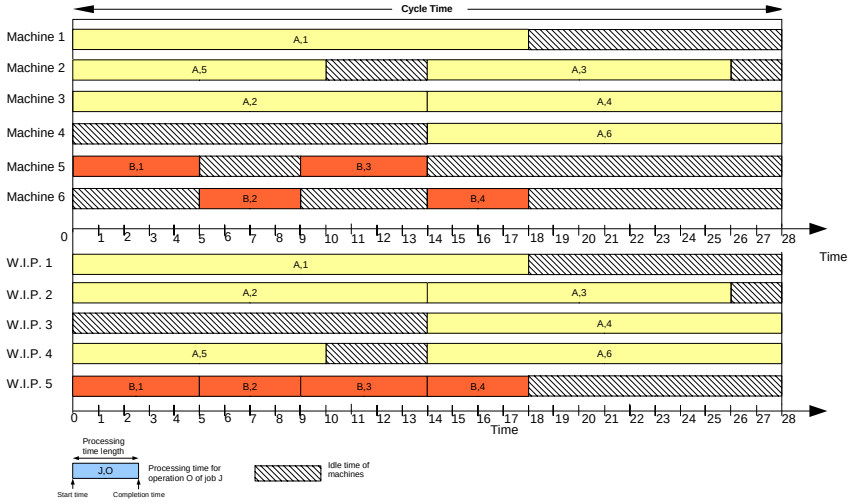


Figure A.4: Solutions for OHL95 Benchmark.

The dual gantt chart in Figure A.4 shows the solution found for the benchmark.

Résumé

Un problème d'ordonnancement consiste à exécuter sur un horizon de temps donné un ensemble de taches au moyen de ressources en nombre limité. On rencontre ce problème dans divers domaines, comme l'industrie de production, dans les systèmes de transport ou encore dans les ordinateurs avec l'allocation des tâches. Dans cette thèse, nous nous concentrons a la fois sur le problème d'ordonnance et le problème d'ordonnancement cyclique, en insistant sur ce dernier. La NP complexité du problème d'ordonnancement des tâches a motivé notre travail de recherche et nous a mené à développer une approche efficace utilisant les réseaux de neurones pour le résoudre. Cette thèse se concentre particulièrement sur le problème du Job Shop Cyclique et sur son utilisation dans le cadre des ateliers flexibles. Pour cela, nous avons développe des modèles pour résoudre le problème du temps cyclique minimum et du travail en cours. Néanmoins, dans cette thèse, nous proposons 3 variations autour des réseaux de neurones : un réseau de neurones récurrents (RNN), une relaxation Lagrangienne pour un réseau de neurones récurrents (LRRNN) et un réseau Hopfield avancé. Plusieurs algorithmes sont combinés avec ces réseaux de neurones pour assurer que les solutions générées sont toutes possibles et pour réduire l'effort de recherche des solutions optimales. A travers des tests comparatifs et expérimentaux, nous sommes capable de démontrer la conformité et l'applicabilité des approches utilisant des réseaux RNN, LRRNN et Advanced Hopfield comme des alternatives attrayantes par rapport à d'autres approches heuristiques traditionnelles pour les problèmes d'ordonnancement cyclique.
Mots clés : Ordonnancement, Ordonnancement cyclique, Réseau neuronal, Atelier flexible, Job shop, Modélisation, Contrainte Précédence, Réseau Hopfield.

Abstract

Scheduling deals with the allocation of tasks requiring processing to limited resources over time in areas including product manufacturing, computer processing and transportation. In this thesis we review the properties of both the general scheduling and cyclic scheduling problems with a focus on the cyclic version of the problem. The NP-Hard complexity of the cyclic scheduling problem has motivated this research work in developing an efficient neural network approach to solving this problem. This thesis focuses specifically on the cyclic job shop and cyclic flexible manufacturing system problems. Hence, models that will solve the minimum cycle time or work in progress of the problems are developed. Here, we develop and study three variations of the recurrent neural network approach. These are the Recurrent Neural Network (RNN), the Lagrangian Relaxation Recurrent Neural Network (LRRNN) and the Advanced Hopfield network approaches. Several algorithms are combined with these neural networks to ensure that feasible solutions are generated and to reduce the search effort for the optimum solutions. We also extend the review to include the cyclic job shop problem with linear precedence constraints. A delinearization algorithm is developed to solve this problem; an approach based on transforming the linear constraints of the problem into its uniform constraints as proven in existing cyclic scheduling literature. Through computational and comparative testing, we are able to demonstrate the suitability and applicability of the RNN, LRRNN and Advanced Hopfield network approaches as attractive alternatives to traditional heuristics in solving these cyclic scheduling problems.
Keywords : Scheduling, Cyclic Scheduling, Neural Network, Flexible Manufacturing System, Job Shop, Modeling, Precedence Constraint, Hopfield Network.