Hybrid Flow Shop Scheduling with Maintenance Constraints: Complexity, Algorithms and Application.

by

Hamid ALLAOUI

Civil Enginner and DEA in Computer Science

Submitted to the
in partial fulfillment of the requirements for the degree of

Docteur de l'Université Pierre et Marie Curie and Docteur en Science de Gestion

at the

UNIVERSITÉ PIERRE ET MARIE CURIE AND FACULTÉS UNIVERSITAIRES CATHOLIQUES DE MONS

June 2004

Signature of Author .....
15 June 2004

Certified by.....
Abdelhakim ARTIBA
Professor
Thesis Supervisor

Certified by.....
Alain RIVIERE
Professor
Thesis Supervisor

Accepted by.....
Marcel GERARD
Professor

Hybrid Flow Shop Scheduling with Maintenance Constraints: Complexity, Algorithms and Application.

by

Hamid ALLAOUI

Submitted to the
on 15 June 2004, in partial fulfillment of the
requirements for the degree of
Docteur de l'Université Pierre et Marie Curie and Docteur en Science de Gestion

Abstract

The success of a product depends on the costs incurred through the entire processing. Scheduling plays a very important role infecting the completion time, the work in process, and resource utility. An efficient schedule can significantly reduce the total costs. Most of the literature on scheduling assumes that machines are always available. However, due to maintenance activities machines cannot operate continuously without some unavailability periods. In general, maintenance activities can be classified into two categories: preventive and corrective. This thesis deals with the scheduling hybrid flow shop with maintenance constraints. From a theoretical point of view three directions are investigated. The complexity of such problems are analyzed. We formulate dynamic programming and branch and bound algorithms to solve such a problem optimally. The performance of some heuristics are calculated to approximate the optimal solution. From a practical point of view we describe an approach based on integrating simulation and optimization in order to deal with additional constraints. We also propose a decision support system which gives the possibility to compare many solution methods.

Thesis Supervisor: Abdelhakim ARTIBA
Title: Professor

Thesis Supervisor: Alain RIVIERE
Title: Professor

Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 Industrial and Economic Context

The rapid evolution and high competitive nature of today's global markets confer to the function Operation-Production a role of first importance in the global competitiveness of companies. Nowadays manufacturers are facing an economy and an international trade where competition is based on products of high quality and increasing changes offered with lower prices while respecting the due dates of customers. In this context the reduction of costs and the improvement of the quality of the products and service became the principal concerns of the industrialists who seek to increase the performances of their companies.

In the field of the industrial production, the current tendencies indicate that the powerful manufacturing systems must be quickly adapted to the fluctuations of the market (random requests) and to the internal disturbances (breakdowns of the machines). The machines must be simultaneously able to manufacture several types of products in small quantities. In such a context, the optimal planning of the production, the reduction of the production cycle time and the control of these machines increasingly become the main objectives both for the investors and the producers. Under these conditions, the determination of the production rate, the maintenance policy and the scheduling rule

which minimize the operations costs of these systems is nowadays an important research problem in the optimization of production systems area.

1.2 Motivation

Production management concerns the management of operational processes in such a way that objectives with respect to flow time and due date reliability of the work orders and with respect to resource utilization are achieved. Maintenance management concerns the management of maintenance processes in such a way that objectives with respect to availability of technical systems are achieved under economic constraints.

Production and maintenance are two important functions in any industrial process and are interrelated problems. In the past, production and maintenance have been treated as two separate functions. Nowadays because of the interdependence between them, there is an increasing interest to develop optimization models that take into consideration the integration of the two functions.

One area where integration between maintenance and production functions can be advantageous is scheduling. Both production and maintenance scheduling must recognize the realities of the other's needs. A classic question is: under what circumstances does production schedule around maintenance versus maintenance scheduling around production? The integration of the two scheduling systems gives both functions the ability to more productively accomplish their scheduling, leading to greater plant utilization.

Most of the literature on scheduling assumes that machines are available at all times. However, due to maintenance activities machines cannot operate continuously without some kind of unavailability periods. In general, maintenance activities can be classified into two categories: preventive and corrective. For preventive maintenance the machine is checked, repaired and re-calibrated before failure. Thus the system can be kept in an appropriate condition. On the other hand, for corrective maintenance the machine is repaired following any breakdown.

In the case of preventive maintenance the question involved is whether the maintenance decision was done separately or jointly with the job scheduling decision. If the maintenance decision is made separately in advance, the non availability periods will then become a given parameter or a constraint while we plan the job scheduling. That is, while we are doing the job scheduling, the machine unavailability period is given. On the other hand, if the maintenance decision is made jointly with the job scheduling. In such a case, the machine non-availability interval is a decision variable. In the case of corrective maintenance the question involved for a partial failure is whether we should stop the machine now and repair it immediately or repair it later. If we do not repair it, the machine can still operate yet at a less-efficient speed. On the other hand, for a complete failure which can happen at any time the machine should be repaired to be back to normal speed.

1.3 Contributions of This Research

Scheduling under maintenance constraints became a main direction for integrating the two functions maintenance and production. It has attracted much attention recently. The non-availability consideration adds complexity to any scheduling problem. In this research we deal with the hybrid flow shop scheduling problem by supposing that the decision of preventive is already made and the corrective action is made immediately after the breakdown.

Three directions have always been very important in the investigation of scheduling problems. The first one is the investigation of computational complexity. The second one is the search for algorithms that give the optimal solution. If the time needed by these algorithms is exorbitant the last direction is imposed. It consists on the use of heuristics to give approximate solutions.

In this thesis we start to study the two machine flow shop scheduling problem to minimize the makespan under availability constraints. The two machine flow shop with-

out availability constraint is a polynomial problem. However if we consider only one non-availability period either on the first machine or on the second machine the problem becomes NP-hard. To solve this problem optimally we propose an improved dynamic programming model, which is independent of the processing times of the jobs, if the gap is on the first machine and an efficient combinatorial approach if the gap is on the second machine. We discuss the optimality conditions and we analyze the performance of Johnson's rule.

A branch and bound algorithm is proposed in this thesis to solve a two stage hybrid flow shop with only one machine on the first stage and m identical machines on the second stage under availability constraints to minimize the makespan which is NP-hard on the strong sense. The time required by this algorithm is still reasonable for a small size instances. However for a large size instances we give the performance of three heuristics.

To reduce the gap between the theory and the real life industry, we deal with a generic problem in term of constraints and objectives. Thus we deal with the k-stage hybrid flow shop with both preventive maintenance and breakdowns to minimize criteria based on flow time and due date. Furthermore cleaning, transportation and setup times are taken into consideration. To tackle this problem which presents the double complexity algorithmic and functional-structural, we propose an approach based on the idea of integrating simulation and optimization. Using this approach and applying some heuristics an experimental study reveals that the performance of heuristics applied to this problem is in dependency with the level of breakdown time. Having the goal to give the researchers and the industrialists a tool of comparison of different solution methods we propose and describe an architecture of a decision support system to schedule an hybrid flow shop under such constraints.

1.4 Outline of The Thesis

All the material in this thesis is devoted to machine scheduling problems. It is presented in nine chapters.

In **chapter two** basic aspects of scheduling problems classification used throughout the thesis including assumptions, optimality criteria, representation are introduced.

In **chapter three** first we introduce the different notions and definitions in the computational complexity field in order to make the difference between polynomial and NP-hard problems. After we discuss the well known the solution methods. They are presented in two categories namely exact methods and approximate methods. In the first category we list in detail the two methods dynamic programming and branch and bound because they are the most widely used methods to solve scheduling problems optimally. In the second category we give some useful definitions and we describe the two metaheuristiques widely used in the literature simulated annealing and tabu search.

Chapter four focuses on the problem of scheduling under maintenance constraints tackled throughout this research. First we describe the scheduling environment and then we specify the maintenance constraints and the criteria to optimize. We illustrate this problem by giving a semiconductor industrial application.

With respect to this problem we examine in **chapter five** the related state of the art through the literature survey of hybrid flow shop scheduling and scheduling with availability constraints.

Chapter six is concerned with the problem of scheduling two machine flow shop with the objective of minimizing the makespan, when it is known that there shall be only one interruption in machine availability either on the first machine or on the second machine due to scheduled maintenance (deterministic case). We solve the problem optimally and we focus on Johnson's rule optimality conditions and performances.

In **chapter seven** we investigate the two stage hybrid flow shop scheduling problem with only one machine on the first stage and m identical machines on the second stage to minimize the makespan. We assume that each machine is subject to at most one

unavailability period fixed in advance. We solve the problem optimally by a branch and bound method and we give the performance of three heuristics.

In **chapter eight** we describe the approach of integrating simulation and optimization to schedule k-stage hybrid flow shop problem with both preventive and corrective maintenance taking into consideration cleaning, transportation and setup times to minimize some criteria based on flow time and due date. We focus on the impact of breakdown times level on the heuristic performances applied to this problem. An architecture of a decision support system is described in this chapter to support the scheduling of hybrid flow shop under such constraints.

Chapter 2

SCHEDULING OVERVIEW AND CLASSIFICATION

2.1 Introduction

Scheduling involves the efficient and effective allocation of limited resources to various tasks to optimize a given measure of performance. It is a decision-making process that can have a profound effect on the success of an organization. Therefore, developments in scheduling theory are both important and essential [15].

In Parunak [19], scheduling problem was described, by asking **what** has to be done **where** and **when**. A task (**what**) occupies a dedicated resource exclusively (**where**) for some period of time (**when**). A set of tasks may form a complex relationship, in which some of the tasks have to use resources in specific orders, thus the temporal order of resource allocations may be restricted by these dependencies among the tasks. Any combination of **what**, **where** and **when** can be said to be a schedule.

A basic problem in scheduling theory is the **machine scheduling problem.** This occurs whenever **jobs**, each of which consists of a given sequence of **operations**, have to be scheduled on one or more **machines** during a given period of time, such that a given objective function is minimized. In the other words, the essence of scheduling problems

gives rise to (1) allocation decisions and (2) sequencing decisions [2]. A scheduling problem is often modelled as an optimization problem where there is an objective function to minimize under some assumptions or constraints to satisfy. Hence the optimal solution of a scheduling problem is to find a processing order or sequence of jobs on each machine so that some measure of performance achieves its optimal value.

In this chapter we deal with the definition and various formulation aspects of scheduling problems. We discuss in the section 2.1 problem representation and introduce notation to designate several concepts involving jobs, operations and machines, and several conditions such as restrictive assumptions or constraints on the jobs and machines are examined. We also focus in the section 2.2 on the optimality criteria used on the scheduling literature. Various job, machine and scheduling characteristics, are reflected by the classification with format $\alpha \mid \beta \mid \gamma$, are given in the section 2.3.

2.2 Assumptions (Restrictions)

We shall introduce a number of basic definitions which explain the structure of scheduling problems. Here, we list assumptions for machines, job and other aspects of our problem.

2.2.1 Assumptions about Machines

M1: the number of machines is known and fixed.

M2: all machines are available at the same instant and independent of each other.

M3: all machines remain available during an unlimited period of time.

M4: each machine can be in one of three states : waiting for the next job, operating on a job or having finished its last job.

M5: all machines are equally important.

M6: breakdown or preventive maintenance of any machine can occur during the planning period.

M7: any machine can process any job assigned to it.

M8: each machine can process at most one job at the same time.

M9: there is only one of each type of machine (no machine group).

2.2.2 Assumptions about Jobs

J1 The number of jobs is known and fixed.

J2 All jobs are available at the same time and are independent.
However we shall face some situations where each job i has a
non-negative integer release date r_i at which it becomes
available for processing.
Further situations arise when jobs are not independent,
i.e. precedence constraints among jobs exist.

J3 Each job can be in one of three states: waiting for the next
machine, being operated on by a machine, or having passed its last machine.

J4 Each job has the same degree of importance. In many cases this
assumption is dropped and to each job i is associated a non-negative integer w_i
related to the importance of that job.

J5 Each job is processed by all the machines assigned to it.

J6 Each job can be processed by only one machine at the same time.

J7 Any operation which has started is not interrupted by other operations and
is continued to its completion. This assumption is sometimes relaxed,
i.e, job splitting (preemption) is allowed.

2.2.3 Other Assumptions

- Each processing time is known and fixed. If not mentioned otherwise, set-up times, and transportation times of jobs between machines are assumed to be negligible.
- Any release date is known and fixed.
- All other quantities needed to define a particular problem are known and fixed, e.g. transportation times of the jobs between the machines.

2.3 Performance Measures (Optimality Criteria)

Before we can define performance measures in precise mathematical terms, we need some definition and notation. Let n denote the number of jobs. Also, we define for each job i $(i = 1, ..., n)$ from the set J of jobs:

r_i : a release date at which job i become available for processing.

$p_{i,j}$: a processing time of its j^{th} operation, $j = 1, ..., m_i$,
where m_i is the number of operations on job i.

d_i : a due date of job i, that is the time by which ideally we would like to have completed job i.

w_i : the weight of job i (relative importance).

We can compute for each job i $(i = 1, ..., n)$:

C_i : the completion time of job i on the last machine on which it requires processing.

$F_i = C_i - r_i$: the flow time of job i which is the sum of waiting and processing times.

$L_i = C_i - d_i$: the discrepancy (lateness) of job i.

$T_i = \max(0, L_i)$: the tardiness of job i.

$U_i = \left\{ \begin{array}{l} 1 \text{ if } L_i > 0 \\ 0 \text{ otherwise} \end{array} \right\}$: the tardiness indicator.

$E_i = \max(0, -L_i)$: the earliness of job i.

$V_i = \left\{ \begin{array}{l} 1 \text{ if } L_i < 0 \\ 0 \text{ otherwise} \end{array} \right\}$: the earliness indicator.

Scheduling may be attempted with respect of various objectives. Criteria can be based on completion times, on due dates, on inventory costs and utilizations, or on combination of previous parameters. The most treated criteria in scheduling area are:

- Makespan : $C_{\max} = \max_{i \in J} \{C_i\}$.
- The Mean Completion Time : $\overline{C} = \frac{\sum_{i \in J} C_i}{n}$

- The Total Completion Time : $\overline{C}_w = \frac{\sum_{i \in J} w_i C_i}{n}$
- Maximum Tardiness : $T_{\max} = \max_{i \in J} \{T_i\}$.
- The Number of Tardy jobs : $N_t = \sum_{i \in J} U_i$
- The Mean Tardiness : $\overline{T} = \frac{\sum_{i \in J} T_i}{N_t}$.
- The Total Weighted Tardiness : $\overline{T}_w = \sum_{i \in J} w_i T_i$
- Maximum Earliness : $E_{\max} = \max_{i \in J} \{E_i\}$.
- The Number of Tardy jobs : $N_e = \sum_{i \in J} V_i$
- The Mean Tardiness : $\overline{E} = \frac{\sum_{i \in J} E_i}{N_e}$.
- The Total Weighted Tardiness : $\overline{E}_w = \sum_{i \in J} w_i E_i$

Here we note that there is classification of performance measures into those are regular and those that are not. A regular measure f is simply one that is non-decreasing in the completion times. Thus, a regular measure f is a function of $C_1, ..., C_n$ such that:

$$f(C_1, ..., C_n) < f(C'_1, ..., C'_n)$$

implies that $C_i < C'_i$ for at least one job i.

2.4 Problem Representation

The first representation of scheduling problems was proposed by Conway, Maxwell and Miller [11], and was extended by Lenstra [18]. It consists of five parameters $A|B|C, D|E$, where A is the number of jobs, B is the number of resources, C describes the type of resource organization, D indicates the possible assumptions or restrictions on the organization, and E specifies the objective function to minimize. Another classification

was proposed by Gragam et al. [14] and extended by Lawler et al. [17]. It is based on three fields: $\alpha \mid \beta \mid \gamma$, where the field α is the machine environment, the field β describes the job characteristics and the field γ denotes the objective function.

Suppose that n jobs numbered $1, ..., n$ have to be processed on m machines M_i, so as to minimize some objective function (s). We represent a machine scheduling problem, which involve well-defined set of jobs, machines and objective function (s), by the three parameter notation $\alpha \mid \beta \mid \gamma$.

We shall now describe the first field $\alpha = \alpha_1\alpha_2$, the machine environment. If $\alpha_1 \in \{1, P, Q, R\}$, each job i consists of a single operation that can be processed on any machine M_j with speed S_{ij} ; the processing of job i on M_j requires a total time $\frac{P_i}{S_{ij}}$.

(P) Identical machines: All S_{ij} are equal. In this case we may assume that $S_{ij} = 1$ for all i and j.

(Q) Uniform machines: $S_{ij} = S_{ik}$ for all machines i, and all jobs j and k. In this case each machine i performs all the jobs at the same speed S_i.

(R) Unrelated machines: There is no particular relationship between the S_{ij} values.

Hence we have:

$\alpha_1 = 1$: a single machine problem

$\alpha_1 = P$: an m identical machine problem

$\alpha_1 = Q$: an m uniform machine problem

$\alpha_1 = R$: an m unrelated machine problem

If $\alpha_1 \in \{O, F, J\}$, each job i consists of a set of operations. Thus:

$\alpha_1 = O$: Open shop problem. There is no restriction on the order in which the operations are executed.

$\alpha_1 = F$: Flow shop problem. Each job has the same sequence of operations.

$\alpha_1 = J$: Job shop problem. Each job has specified sequence of operation which may differ from the sequence of operation for other jobs.

The parameter α_2 denotes whether m is a constant or variable; if α_2 is a positive integer then the number of machines is constant and equal to α_2. If α_2 is not specified,

then m is variable.

The second field $\beta = \beta_1, ..., \beta_8$ describes job and resource characteristics. Parameter $\beta_1 \in \{\varnothing, pmtn\}$ indicates the possibility of task preemption:

$\beta_1 = \varnothing$: no preemption is allowed.

$\beta_1 = pmtn$: preemption is allowed.

Parameter $\beta_2 \in \{\varnothing, res\}$ characterizes additional resources constraints:

$\beta_2 = \varnothing$: no additional resources constraints exist.

$\beta_2 = res$: there are specified resource constraints.

Parameter $\beta_3 \in \{\varnothing, prec, uan, tree, chains\}$ represents the precedence constraints:

$\beta_3 = \varnothing$: independent jobs.

$\beta_3 = prec$: general precedence constraints.

$\beta_3 = uan$: unconnected activity networks.

$\beta_3 = tree$: precedence constraints forming a tree.

$\beta_3 = chains$: precedence constraints forming a chains.

Parameter $\beta_4 \in \{\varnothing, r_i\}$ describes release dates:

$\beta_4 = \varnothing$: all release dates are equal to zero.

$\beta_4 = r_i$: release dates differ per job.

Parameter $\beta_5 \in \{\varnothing, p_i = p, \underline{p} \leq p_i \leq \overline{p}\}$ describes job processing times:

$\beta_5 = \varnothing$: jobs have arbitrary processing times.

$\beta_5 = (p_i = p)$: all jobs have the same processing time equal to p.

$\beta_5 = (\underline{p} \leq p_i \leq \overline{p})$: jobs have processing times between $\underline{p}$ and $\overline{p}$.

Parameter $\beta_6 \in \{\varnothing, d_i\}$ describes due dates:

$\beta_6 = \varnothing$: due dates are not considered on the system.

$\beta_6 = d_i$: due dates are imposed on the performance of a set of jobs.

Parameter $\beta_7 \in \{\varnothing, n_i \leq k\}$ describes the maximal number of operations per job in case of job shop system.

$\beta_7 = \varnothing$: the number of operations is arbitrary.

$\beta_7 = (n_i \leq k)$: the number of operations for each job is not greater then k.

Parameter $\beta_8 \in \{\varnothing, no - wait\}$ describes a no-wait property in the case of scheduling on dedicated machines.

$\beta_8 = \varnothing$: stocks of unlimited capacity are assumed.

$\beta_8 = no - wait$: stocks among machines are of zero capacity and a job after finishing its processing on one machines must immediately start on the consecutive machine.

The field γ denotes the objective function or the optimality criterion (performance measure), i.e. $\gamma \in \{C_{\max}, T_{\max}, \sum C_i, \sum T_i, \sum W_i C_i, \sum W_i T_i\}$.

To illustrate the $\alpha \mid \beta \mid \gamma$ representation we give some examples and in each example we will describe the problem.

Example 1 $P \mid prec; p_i = 1 \mid C_{\max}$ *is the problem of scheduling jobs with unit processing times and arbitrary precedence constraints on m identical parallel machines to minimize the makepsan.*

Example 2 $1|r_i; pmtn| \ \overline{C}$ *is the problem of scheduling a single machine with release dates to minimize the mean completion times.*

Example 3 $F2|no-wait|C_{\max}$ *is the two machine no-wait flow shop scheduling problem to minimize the makespan.*

Bibliography

[1] Artiba A. and Elmaghraby, S.E. , (1997). "The planning and scheduling of production systems ". Chapman & Hall.

[2] Baker, K.R., (1974). "Introduction to Sequencing and Scheduling". John Wiley & Sons.

[3] Baker, K.R. and Scudder, G.D., (1990). "Sequencing with earliness and tardiness penalties: A review". *Oper Res 38*, 22–36.

[4] Blazewicz, J., Ecker, K.H., Schmidt, G. and Weglarz, J., (1994). "Scheduling Computer and Manufacturing Processes". Springer.

[5] Blazewicz, J., Lenstra, J.K. and Rinnooy Kan, A.H.G., (1983). "Scheduling subject to resource constraints: Classification and complexity". *Disc App Math 5 1*, 11–24.

[6] Bowman, E. H., (1959). "The schedule sequence problem". *Oper Res, Vol. 7*, 621-24.

[7] Brucker, P., (1998). "Scheduling Algorithms". Third Edition, Springer.

[8] Brucker, P., Drexl, A., Möhring, R., Neumann, K. and Pesch, E., (1999). "Resource-constrained project scheduling: Notation, classification, models and methods". *Euro J Oper Res 112*, 3–41.

[9] Carlier, J. et Chrétienne, P., (1988). "Problèmes d'ordonnancement : modélisation, complexité, algorithmes". Masson.

[10] Chu, C. et Proth, J.M, (1996). "L'ordonnancement et ses applications". Masson.

[11] Conway, R.W., (1967). "Theory of Scheduling". Hardcover & Addison-Wesley.

[12] Edward G. Coffman, Jr. , Peter J. Denning, 1990. "Operating Systems Theory", Prentice Hall Professional Technical Reference.

[13] Gargeya, V.B. and Deane, R.H., (1996). "Scheduling research in multiple resource constrained job shops: A review and critique". *Int J Prod Res 34*, 2077–2097.

[14] Graham, R.L., Lawler, E.L, Lenstra, J.K. and Rinnooy Kan, A.H.G., (1979). "Optimization and approximation in deterministic sequencing and scheduling: A survey". *Ann. Disc Math. 5*, 287-326.

[15] Gupta, J.N.D., (2002). "Editorial Special issue: Scheduling theory and applications". *Prod plan & Cont, vol 13, no 2*, 103-104.

[16] Johnson, L.A. and Montgomery, D.C., (1964).

[17] Lawler, E. L., Lenstra,J. K., Rinnooy Kan,A. H. G. and Shmoys,D. B., (1989). "Sequencing and scheduling: algorithms and complexity". Report BS-R8909, Center for Mathematics and Computer Science, Amsterdam.

[18] Lenstra, J.K., (1979). "Sequencing by enumerative methods, Mathematisch Centrum". Amsterdam.

[19] Parunak, V.D, (1991). "Characterizing the Manufacturing Systems Scheduling System". *J of Man Sys, 10(3)*, 241-259.

[20] Pinedo, M., (2002). Scheduling : Theory, Algorithms and Systems. Second Edition, Prentice-Hall.

[21] Rinnooy Kan, A.H.G., (1976). "Machine scheduling problems. Classification, complexity and computations ". Nijhoff, The Hague.

Chapter 3

COMPUTATIONAL COMPLEXITY AND SOLUTION METHODS

3.1 Introduction

Recently, scheduling problems have received much intention. Two issues have always been widely investigated in this area of problems, namely:

- The computational complexity of scheduling problems.
- The search for algorithms that determine the optimal sequence efficiently, preferably in polynomial time.

Classifying scheduling problems according to their algorithmic complexity reveals that some problems are easier to solve than others. If a problem is NP-hard (see section 3.2) then it is unlikely that it can be solved by a polynomial algorithm. If the optimal solution is unattainable in reasonable time by exact methods (Dynamic Programming, Branch and Bound...) then it is reasonable to sacrifice optimality and settle for a "good" feasible solution that can be computed efficiently. Of course, we would like to sacrifice as

little optimality as possible, while gaining as much as possible in efficiency. Trading-off optimality in favour of tractability is the paradigm of approximation algorithms.

In the section 3.2 of this chapter the theory of computational complexity related to the scheduling area is discussed. We focus in the section 3.3 on the well-known methods that have been used to solve machine scheduling problems.

3.2 Theory of Computational Complexity

Complexity theory is based on a mathematical framework developed by logicians and computer scientists. This theory was developed to study the intrinsic difficulty of algorithmic problems and has proven very useful for combinatorial optimization. In complexity theory a distinction is made between optimization problem and decision problem. The question raised in a decision problem requires either a yes or a no answer. With every optimization problem one can associate a decision problem. If there exists a polynomial time algorithm for a decision problem it exists also for an optimization problem and vice versa [42]. The techniques of Cook [9] and Karp [27] have been applied extensively in this area to locate the limit between P the class of problems for which a polynomial bounded, good or efficient algorithm exists, and NP-complete, the class of problems for which the existence of such algorithm is very unlikely [6]. Problem reduction is a fundamental notion in complexity theory. To focus on this important field in the scheduling area we give some definitions ([12] and [42]):

Definition 4 *The class P contains all decision problems for which there exists a Turing machine algorithm that leads to the right yes-no answer in a number of steps bounded by a polynomial in the length of the encoding.*

Definition 5 *The class NP contains all decision problems for which the correct answer, given a proper clue, can be verified by a Turing machine in a number of steps bounded by a polynomial in the length of the encoding.*

The class P is, clearly, a subclass of the class NP. One of the most important issues open in mathematical logic and combinatorial optimization is the question whether or not $P = NP$.

Definition 6 *Problem π' is reduced to problem π ($\pi' \alpha\ \pi$) if , for any instance (particular case of a problem) of π' , an instance of π can be constructed in polynomial bounded time such that solving the instance of π will solve the instance of π' as well. The two problems π' and π are equivalent if $\pi' \alpha\ \pi$ and $\pi\ \alpha\ \pi'$.*

When $\pi' \alpha\ \pi$ and $\pi \in P$, this implies that $\pi' \in P$. Conversely, if $\pi' \alpha\ \pi$ and $\pi' \notin P$, then $\pi \notin P$.

Definition 7 *A problem π, either a decision problem or an optimization problem, is called NP-hard if the entire class of NP problems polynomially reduces to π.*

If for example a problem can be solved in polynomial time as a function of the size of the problem in unary encoding but not binary encoding we say that the problem is **NP-hard** in the **ordinary sense** and the algorithms for this kind of problems are called **pseudo-polynomial**. However if polynomial time algorithm does not exist under either unary or binary encoding we say that the problem is NP-hard in the **strong sense**.

We illustrate all these concepts we give some scheduling example:

Example 8 *Some Polynomially solvable problems*

$1|r_i; p_i = p| \sum w_i U_i$ $O(n^7)$ *Baptiste [5]*

$1|prec; r_i|C_{\max}$ $O(n^2)$ *Lawler [31]*

Example 9 *Some NP-hard on the ordinary sense problems*

$1|\ |\sum T_i$ *(Lawler [32]);* $P2|\ |C_{\max}$ *(Karp [27]);*

Example 10 *Some NP-hard on the strong sense problems*

$F3|\ |C_{\max}$ *(Garey et al. [15])* ; $F2|prec|C_{\max}$ *(Monma [38])*

An optimization problem is specified by a set $\Im$ of inputs (or instances), by a set $SOL(I)$ of feasible solutions for every input $I \in \Im$, and by an objective function c that specifies for every feasible solution σ in $SOL(I)$ an objective value or cost $c(\sigma)$. Suppose that we deal only with a minimization problem where the optimal solution is a feasible solution with minimum possible cost. We denote the optimal objective value for instance I by $OPT(I)$. By $|I|$ we denote the size of an instance I, i.e., the number of bits used in writing down I in some fixed encoding. Now assume that we are dealing with an $NP-hard$ optimization problem where it is difficult to find an exact optimal solution within polynomial time in $|I|$. At the expense of reducing the quality of the solution, we can often get considerable speed-up in the time complexity. This motivates the following definition ([45], [12], [44] and [23]).

Definition 11 ***Approximation algorithms***

X be a minimization problem. Let $\epsilon > 0$, and set $\rho = 1 + \epsilon$. An algorithm A is called a $\rho -$ approximation algorithm for problem X, if for all instances I of X it delivers a feasible solution with objective value $A(I)$ such that

$$|A(I) - OPT(I)| \leq \epsilon \times OPT(I)$$

In this case, the value ρ is called the performance guarantee or the worst case ratio of the approximation algorithm A. The class APX consists of all minimization problems that have a polynomial time approximation algorithm with some finite worst case ratio.

Definition 12 ***Approximation schemes***

X be a minimization problem.

An approximation scheme for problem X is a family of $(1+\epsilon)-$approximation algorithms A_ϵ for problem X over all $0 < \epsilon < 1$.

A polynomial time approximation scheme ($PTAS$) for problem X is an approximation scheme whose time complexity is polynomial in the input size.

A fully polynomial time approximation scheme ($FPTAS$) for problem X is an approxi-

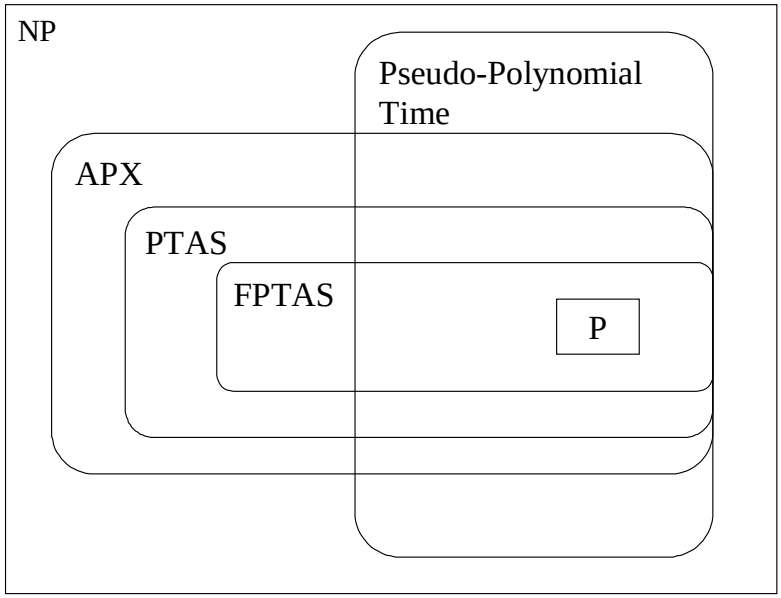

Figure 3-1: Relationships between some of the complexity classes.

mation scheme whose time complexity is polynomial in the input size and also polynomial in $\frac{1}{\epsilon}$.

Figure3-1 illustrates the relationships between the classes NP, APX, P, the class of problems that are pseudo-polynomially solvable, and the classes of problems that have a PTAS and FPTAS.

3.3 Methods of Solutions

Since a machine scheduling problem is a combinatorial optimization problem, the objective in this kind of problems is to find an optimal schedule from a finite number of

feasible schedules. To find one with the smallest value of the performance measure, we can enumerate all the schedules. For example for one machine non-preemptive scheduling problems, there exists $n!$ possible solutions. Hence for the corresponding m-machines problem, there are $(n!)^m$ possible solutions. This number is very large even for relatively small values of m and n. It is clear that in this case the enumerations of all possible solutions is still not suitable even for problems of small size. In this section we present two exact methods: Branch and Bound and Dynamic Programming based on the idea of intelligently enumerating of all feasible solutions. We also focus on approximation methods called also heuristics which give an approximate solution.

3.3.1 Dynamic Programming Method

Dynamic Programming is one of the more widely used techniques (Linear Programming, Branch and Bound...) for dealing with combinatorial optimization problems. It is an optimization approach that transforms a complex problem into a sequence of simpler problems; its essential characteristic is the multi-stage nature of the optimization procedure. It can be applied to problems solvable as well as problems not solvable in polynomial time. Dynamic Programming is an approach developed to solve sequential, or multi-stage, decision problems; hence, the name "dynamic" programming. Dynamic Programming, like Branch and Bound approach, is a way of decomposing certain hard to solve problems into equivalent formats that are more amenable to solution. Basically, what Dynamic Programming approach does is that it solves a multi-variable problem by solving a series of single variable problems. This is achieved by tandem projection onto the space of each of the variables. In other words, we project first onto a subset of the variables, then onto a subset of these, and so on. The essence of dynamic programming is Richard Bellman's Principle of Optimality [4]. This principle, even without rigorously defining the terms, is intuitive:

Lemma 13 *An optimal policy has the property that whatever the initial state and the initial decisions are, the remaining decisions must constitute an optimal policy with regard*

to the state resulting from the first decision.

Hence it determines the optimal solution for each subproblem and its contribution to the objective function. At each iteration, it determines the optimal solution for a subproblem, which is larger than all previously solved subproblems. It finds a solution for the current subproblem by utilizing all the information obtained before in the solutions of all the previous subproblems. Dynamic Programming is characterized by three types of equations, namely:

- Initial conditions.
- A recursive relation.
- An optimal value function.

3.3.2 Branch and Bound Method

Branch and Bound method is enumeration technique used to combinatorial optimization problems. It was developed and used by Eastman [11] for the traveling salesman problem and by Land and Doig [30] for integer programming. It was first applied to scheduling problems by Lomnicki (*Lomnicki*) and Ignall and Scharge [26]. In this method widely used for scheduling problems we try to construct a proof that a solution is optimal, based on successive partitioning of the solution space. The branch in Branch and Bound refers to this portioning process; the bound refers to lower bounds that are used to construct a proof of optimality without exhaustive search [41].

We calculate a lower bound (LB) on the value of each solution in a subset. Hence we say that a node is active if the associated lower bound is the smallest lower bound. If the associated lower bound of the node is larger than the upper bound (UB), this subset is ignored. This upper bound UB is usually defined as the minimum of the values of all feasible solutions currently found. Otherwise if no feasible solution is known, UB is initially taken to be infinity until the first feasible solution is found. At each node from

which to branch according to some search strategy the sequence is evaluated and if its value is less than the current UB, this UB is reset to take that value. The procedure is repeated until all subsets have been considered (i.e., lower bounds of all nodes in the search tree are greater than the UB); a feasible solution with this UB is an optimal solution [43].

The quality of the bound is the most important factor for the efficiency of the Branch and Bound algorithm. One often has a choice between bounds that are relatively tight but require relatively large computation time and bounds that are not so tight but can be computed fast. The most bounding methods used in the literature are as follows:

- Relaxation of constraints.
- Lagrangian relaxation of constraints.
- Dynamic programming state-space relaxation.
- Relaxation of objective.

The branching procedure describes the method used to partition a set of solutions, which is represented by a node, into possible subsets. Each subset is represented by a child of the original node. The most branching methods used in the literature are as follows:

- Forwards branching : the jobs are sequenced one by one from the beginning.
- Backwards branching : the jobs are sequenced one by one from the end.
- A job can be sequenced either at the beginning or at the end according to an heuristic method.
- A job can be sequenced first, last, directly before or directly after an other job according to an heuristic method.

The search strategy describes the method of choosing a node of the search tree to explore. There are three most used techniques to choose this node:

- Branching from the node with smallest lower bound.
- Branching from the recently created node.
- Branching from a node with smallest lower bound amongst the recently created nodes.

One can improve the computational time of the Branch and Bound algorithm by adding dominance rules to specify whether a node can be eliminated before computing its lower bound.

3.3.3 Heuristics and Approximation Algorithms

Approximation algorithms called also Heuristics have been developed in response to the impossibility of solving a great variety of important optimization problems. Too frequently, when attempting to get a solution for a problem, one is confronted with the fact that the problem is NP-hard. For years NP-hard problems were treated with exact methods like Dynamic Programming and Integer Programming tools or "heuristics". The first kind of tools are forms of implicit enumeration algorithms that combine efficient derivations of lower and upper bounds with a hopeful search for an optimal solution. The amount of time required to solve even a typical moderate-size optimization problem is exorbitant. The point is that exact methods provide no guarantee. It is impossible to tell if 5 additional minutes of running time would get you a significantly better solution, or if 5 more days of running time would yield no improvement at all [23]. Approximation algorithms address both the issue of guarantee and of making good feasible solutions available.

One method of studying the performance or the quality of heuristics is to examine its worst-case performance (see Section 3.2). This amount serves to establish the maximum

relative deviation between the optimal and near optimal solutions. The first important results on the worst-case performance of heuristics applied to scheduling problems were given by Garey, Graham and Johnson [16].

Example 14 *For the problem of minimizing makespan in m parallel machines (Pm|* $|C_{\max}$*), we have* $R_{LPT} = \frac{C_{\max}^{LPT}}{C_{\max}^{*}} \leq \frac{4}{3} - \frac{1}{3m}$*, and this bound is tight. LPT is the longest processing time rule (Graham [21]).*

Local search techniques are widely used to obtain approximate solutions to combinatorial optimization problems in general, and the scheduling problems in particular. Two important categories of local search methods are neighborhood search and genetic algorithms. As follows two neighborhood search algorithms: simulated annealing and tabu search are widely used for scheduling problems:

Simulated Annealing (SA)

Simulated Annealing was proposed to solve combinatorial optimization problems by Kirkpatrick, Gelatt and Vecchi [28]. SA is an iterative optimization technique that frees from local optima with following method: a new solution that is accepted with a probability called the acceptance function. This function is computed from the evolution of the optimized criteria and a control parameter called the temperature. This short description of SA shows that the algorithm is fully defined with the following elements:

- the representation of the solution and the evaluation function. These two elements are generally strongly connected to the problem but they have an effect on the global behavior of the algorithm;
- the definition of the neighborhood, i.e the method used to choose a new solution from the current solution;
- the outline of the acceptance function. This outline depends on the expression used to compute the probability and it depends on the temperature from an iteration to the next.

Tabu Search (TS)

Tabu Search was originally proposed by Glover [19]. TS is an iterative improvement approach designed for getting a global optimum solution for combinatorial optimization problems. The idea of TS can be briefly described as follows. Starting from an initial solution, TS iteratively moves from the current solution s to its best neighboring solution s^* , even if s^* is worse than the current solution s, until a superimposed stopping criterion becomes true. In order to avoid cycling to some extent, moves which would bring us back to a recently visited solution should be forbidden or declared tabu for a certain number of iterations. This is accomplished by keeping the attributes of the forbidden moves in a list, called a tabu list. The size of the tabu list must be large enough to prevent cycling, but small enough not to forbid too many moves. Additionally, an aspiration criterion is defined to deal with the case in which an interesting move (such as a move that leads to a new best solution) is tabu. If a current tabu move satisfies the aspiration criterion, its tabu status is canceled and it becomes an allowable move.

Bibliography

[1] Aarts, E. and Lenstra, J.K., (1998). "Local search in combinatorial optimization". John Wiley & Sons.

[2] Albers, S. and Brucker, P., (1993). "The complexity of one-machine batching problems". *Dis App Math 47*, 87-107.

[3] **Allaoui, H.**, (2002). "Scheduling hybrid flowshops with availability constraint : Dynamic Programming and Heuristics". DEA at Mons Hainault University, Belgium.

[4] Bellman, OR., Esogbue, A.O. and Nabeshima, I., (1982). "Mathematical aspects of scheduling and applications". Pergamon Press.

[5] Baptiste, P., (1999). "Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times". *Journal of Scheduling 2*, 245-252.

[6] Blazewicz, J., Lenstra, J.K. and Rinnooy Kan, A.H.G., (1983). "Scheduling subject to resource constraints: classification and complexity". *Disc App Math 5*, 11-24.

[7] Brucker, P., Jurisch, B. and Kraemer, A., (1997). "Complexity of scheduling problems with multi-purpose machines". *Ann Oper Res 70*, 57-73.

[8] Brucker, P., Jurisch, B. and Sievers, B., (1994)."A branch and bound algorithm for the job-shop scheduling problem".*Disc App Math 49,* 1-3.

[9] Cook, S.A., (1971). "The complexity of theorem-proving procedures". *Proc of 3rd ann ACM on Theory of computing* , 151-158.

[10] Du, J. and Leung, J.Y.-T. (1989). "Complexity of scheduling parallel task systems". *SIAM J Disc Math 2*, 473-487.

[11] Eastman, W.L., (1958). "Linear Programming with Pattern Constraints". Report BL 20, The Computation Laboratory, Harvard University, Cambridge.

[12] Garey, M.R., and Johnson, D.S. (1979). "Computers and intractability, a guide to the theory of NP-Completeness". Freeman.

[13] Garey, M. R. and Johnson, D. S.(1978). " Strong NP-Completeness Results: Motivation, Examples, and Implications". *JACM 25 n.3*, 499-508.

[14] Garey, M. R. and Johnson, D. S., (1979). "Computers and Intractability: A Guide to the Theory of NP-Completeness". W. H. Freeman & Co., New York.

[15] Garey, M. R. and Johnson, D. S. and Sethi, R., (1976). "The complexity of flowshop and jobshop scheduling". *Math Ope Res 1*, 117-129.

[16] Garey, M. R., Graham, R. L. and Johnson, D. S., (1974). "Some NP-complete geometric problems ". *Proc of 8th ann ACM on Theory of computing.* 47 - 63.

[17] Gen., M. and Cheng, R., (1997). "Genetic algorithms & engineering design". John Wiley & sons.

[18] Gill, P.E., Murray, W., and Wright, M.H., (1981). "Practical Optimization". Academic Press.

[19] Glover, F. and Laguna, M., (1998). "Tabu Search". Kluwer Academic Publishers.

[20] Gonzalez, T. and Sahni, S. (1978). "Flow shop and job shop schedules: complexity and approximation". *Oper Res 26*, 36-52.

[21] Graham, R.L., Lawler, E.L, Lenstra, J.K. and Rinnooy Kan, A.H.G. (19979). "Optimization and approximation in deterministic sequencing and scheduling: A survey". *Ann. Disc Math. 5*, 287-326.

[22] Hall, L.A., (1995). "Approximability of flow shop scheduling". *Proc. 36th IEEE Symp. on Foundations of Computer Science*, 82–91.

[23] Hochbaum, D.S., (1997). "Approximation Algorithms for NP-Hard Problems". PWS Publishing Company .

[24] Hoogeveen, J.A., Van de Velde, S.L. and Veltman., V. (1994). "Complexity of scheduling multiprocessor tasks with prespecified processor allocations", *Disc App Math 55*, 259-272.

[25] Ibaraki, T. and Nakamura, Y., (1987). "A dynamic programming method for single machine scheduling", *Eur J Oper Res 76, 1, 6,* 72-82.

[26] Ignall, E. and Schrage, L. (1965). "Application of the Branch and Bound Technique to some Flow-Shop Scheduling Problems". *Oper Res 13*, 400-412.

[27] Karp, R.M., (1972). "Reducibility among combinatorial problems". *In R.E. Miller, J.W. Thatcher (eds.): Complexity of Computer Computations, Plenum Press*, 85-103.

[28] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P., (1983). "Optimization by simulated annealing". *Science 220*, 671-680.

[29] Laarhoven, V. and Aarts, E.H.L, (1987). "Simulated Annealing : Theory and Applications" . Kluwer Academic Publishers.

[30] Land, A.H. and Doig, A.G., (1960). "An automatic method for solving discrete programming problems". *Econometrica 28*, 497–520.

[31] Lawler, E.L., (1973). "Optimal sequencing of a single machine subject to precedence constraints". *Management Science 19*, 544-546.

[32] Lawler, E.L., (1977). "A 'pseudopolynomial' algorithm for sequencing jobs to minimize total tardiness". *Ann Disc Math 1*, 331-342.

[33] Lee, C.Y. and Uzsoy, R., (1992). "A new dynamic programming algorithm for the parallel machines total weighted completion time problem". *Oper Research Lett 11* , 73–75.

[34] Lenstra, J.K., Rinnooy Kan, A.H.G., and Brucker, P, (1977). "Complexity of machine scheduling problems". *Ann. Discrete Math. 1*, 343-362.

[35] Lenstra, J.K. and Rinnooy Kan, A.H.G., (1979). "Computational complexity of discrete optimization problems". *Ann Disc Math 4*, 121-140.

[36] Lomnicki, Z. A. (1965). "A Branch-and-Bound Algorithm for the Exact Solution of the Three-Machine Scheduling Problem". *Oper Res Quar 16(1)*, 89-100.

[37] Luus, R., (2000). "Iterative Dynamic Programming". Chapman & Hall.

[38] Monma, C.L., (1989). "On the complexity of scheduling with batch setup times". *Operations Research 30*, 116-124.

[39] Nemhauser G.L. and Wolsey, L.A., (1988). "Integer and combinatorial optimization". John Wiley & Sons.

[40] Osman, I.H., (1996). "Meta-Heuristics: Theory & Applications". Kluwer Academic Publishers.

[41] Papadimitriou, C.H. and Steiglitz, K., (1982). "Combinatorial Optimization : Algorithms and Complexity". Prentice Hall.

[42] Pinedo, M., (2002). Scheduling : Theory, Algorithms and Systems. Second Edition, Prentice-Hall.

[43] Potts, C. (1985). "Analysis of A Linear Programming Heuristic for Scheduling Unrelated Parallel Machines". *Disc App Math 10*, 155-164.

[44] Schuurman, P. and Woeginger, G.J., (2000). "A polynomial time approximation scheme for the two-stage multiprocessor flow shop problem". *Theo Comp Science 237*, 105-122.

[45] Shmoys, D.B., Stein, C. and Wein, J., (1994). "Improved approximation algorithms for shop scheduling problems". *SIAM J. Comput. 23,* 617–632.

[46] Ullman, J. D. (1975). "NP-complete scheduling problems". *J Comp and Syst Sciences 10*, 384-393.

[47] Ventura, J. A. and Weng, M.X, (1995). "An improved dynamic programming algorithm for the single-machine mean absolute deviation problem with a restrictive common due date". *Oper Res Lett, 17.3*, 149-152.

[48] Yagiura, M. and Ibaraki, T., (1996). "The use of dynamic programming in genetic algorithms for permutation problems". *Eur J Oper Res, 92.2*, 387-401.

[49] Wolsey, L.A., (1998). "Integer programming". Wiley New York.

Chapter 4

PROBLEM STATEMENT

4.1 Introduction

A common manufacturing environment in many industries (such as the glass, steel paper, textile and semiconductor) is an hybrid flow shop. This scheduling system is a mixture of the serial shop organization and the parallel shop organization. In this thesis we deal with hybrid flow shop scheduling problems assuming that machines may become unavailable during certain periods. Indeed most of the literature that deal with scheduling problems assume that the machines are always available during the scheduling period. However in most real life industrial settings a machine can be unavailable for maintenance activities, such as unforeseen breakdowns (stochastic unavailability) or scheduled preventive maintenance where the periods of unavailability (called also gaps) are known in advance (deterministic unavailability). Under such circumstances, special consideration is needed in order to obtain optimal solutions.

This chapter is organized as follows. A description of the hybrid flow shop environment studied in this thesis is given in the section 4.1. The section 4.2 focuses on maintenance constraints. A practical industrial case is illustrated in the section 4.3.

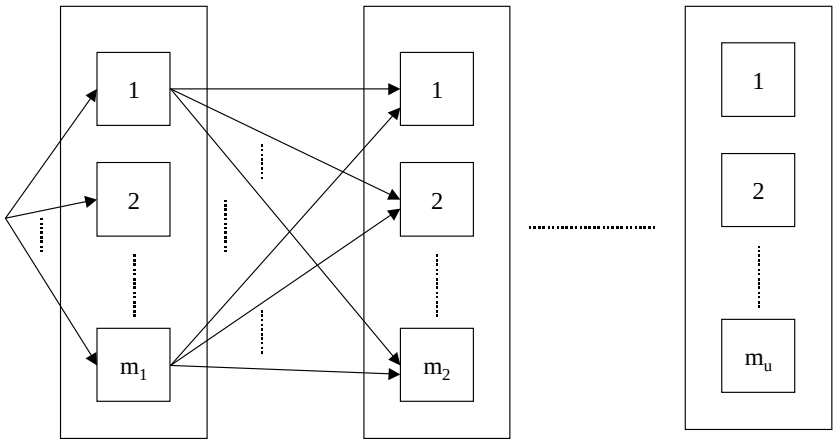

Figure 4-1: Hybrid Flow Shop

4.2 Hybrid Flow Shop

A hybrid flow shop (HFS) called also flexible flow shop, is a multi-stage production process which consists of a set of u stages with each stage k $(1 \leq k \leq u)$ having m_k parallel machines with the property that all jobs have to be processed through all the stages on the same order: there are n jobs J_i $(1 \leq i \leq n)$, each job consisting of a chain of u operations $O_{i1}, ..., O_{iu}$ that have to be executed on this order. In the classical flow shop each stage contains only one machine. The design of a such environment is described in the Figure4-1.

At any time every job can be processed by at most one machine and every machine can process at most one job. In scheduling hybrid flow shop organizations treated in this

thesis, basic assumptions are as follows:

- All N jobs to schedule are independent and available for processing at time $t = 0$ (i.e $\forall i : r_i = 0$).
- Machines of a same stage can be identical, uniform or unrelated.
- The processing time $p_{i,j,k}$ of each job i on the machine j of stage k is known and fixed.
- Each job may have its own due date, d_i, which represents the date the job is promised to the customer.
- Preemption and splitting of any particular job is not allowed: a job, once started on a machine, continues in processing until it is completed.
- Jobs are allowed to wait between two stages, and the storage is unlimited.
- Setup, cleaning and transportation times are not negligible and not included in processing times.

Four kinds of decision-making goals are prevalent in scheduling and considered in this thesis:

- Efficient utilization of resources: maximum completion time (or makespan) $C_{\max}$.
- Rapid response to demands, or minimization of work in process: mean completion time $\overline{C}$, mean flow time $\overline{F}$, or mean waiting time $\overline{W}$.
- Close conformance to prescribed deadlines: mean tardiness $\overline{T}$, maximum tardiness $T_{\max}$, and the number of tardy jobs N_t.
- Just in time based management: mean earliness $\overline{E}$, maximum earliness $E_{\max}$, and the number of early jobs N_e.

4.3 Maintenance Constraints

Most of the literature on scheduling assumes that machines are available at all times. However, due to various reasons, machines may not be always available in many realistic situations. Therefore, a more realistic scheduling model should take into account the following associated machine maintenance activities:

- Preventive maintenance: where maintenance is performed on a scheduled basis with scheduled intervals often based on manufacturers' recommendations and past experience with the equipment. This may involve replacement or repair, or both.
- Breakdown maintenance: Replacement or repair is performed only at the time of failure. This may be the appropriate strategy in some cases, such as when the hazard rate is constant and/or when the failure has no serious cost or safety consequence or it is low on the priority list.

The problem of integrating production and preventive maintenance has been generally approached in the literature in two different ways. Some authors approached this problem by determining the optimal preventive maintenance schedule in the production system and others by taking maintenance as a constraint to the production system ([6], [7] and [8]). Thus the question involved here is whether the preventive scheduling decision was done separately or jointly with the job scheduling decision. In this thesis and for the preventive maintenance we deal only with the first approach (i.e we ensure that the preventive maintenance for each machine is done in the specified window). However for the breakdown maintenance the question does not arise in this case since the machine breakdown can happen at any time.

In this thesis, we focus on problems in which job preemption is not allowed. There are three types of machine unavailability ([23], [25] and [26]) discussed in the literature:

- *Resumable* : A machine is called resumable if a job that cannot be finished before a down period of a machine can be continued without any penalty after the machine becomes available again.

- $Non - Resumable$: A machine is called non-resumable if the job that cannot be completed before a period of machine non-availability must be totally restarted rather than continuing after the machine is brought back on line.

- $Semi - Resumable$: A machine is called semi-resumable if the non-finished job before a period of machine non-availability must be partially restarted. There are two-types of semi-resumability: In $Type - I$, in addition to processing the non-finished part, the machine needs to process extra work that is proportional to the finished part of that job. In $Type - II$, if a job is not processed to completion before the machine is stopped for maintenance, an additional setup is necessary when the processing is resumed. Figure4-2 depicts the two types. In the upper figure, z is the finished part of job I before the non-availability interval, d and $0 \leq \alpha \leq 1$. In the lower figure, q is the setup time of job i.

4.4 Notations

In order to be able to refer the problem under study, we use a notation similar to the three field notation $\alpha \mid \beta \mid \gamma$ presented in the section 2.3 and modified by Kubiak [22] and Lee [23] taking into account the availability constraints:

- $F2(P), a_{kj,l} \mid r \mid Cmax$: Minimizing the makespan hybrid flowshop problem with a *resumable* availability constraint and an arbitrary number of gaps (l) on each machine (j) on each stage (k);

- $F2(P), a_{kj,l} \mid nr \mid Cmax$: Minimizing the makespan in the two-machine flowshop problem with a *non-resumable* availability constraint and an arbitrary number of gaps (l) on each machine (j) on each stage (k);

- $F2(P), a_{kj,l} \mid sr \mid Cmax$: Minimizing the makespan in the two-machine flowshop problem with a *semi-resumable* availability constraint and an arbitrary number of gaps (l) on each machine (j) on each stage (k);

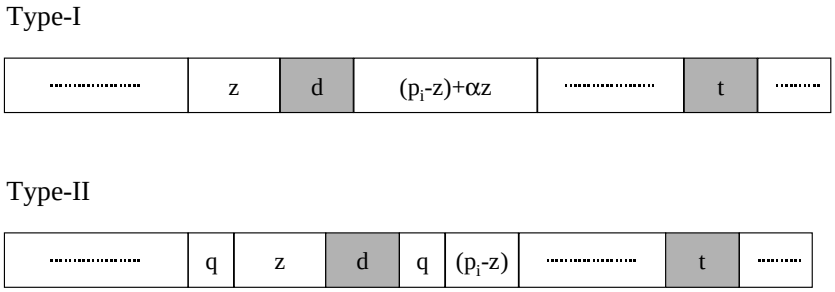

Figure 4-2: The semi-resumable case.

For example $F2(P), h_{11,l}|\ (m_1 = 1, m_2 = 2), nr\ |Cmax$ represents the problem of minimizing makespan in the two stages hybrid flow shop problem with one machine on the first stage and two machines on the second stage under a *non-resumable* availability constraint and with an arbitrary number of gaps on the first stage and no gaps on the second stage.

For the classical two machine flow shop problem it is more easy to use the Lee's notation [23].

- $F2|r - a|Cmax$: Minimizing the makespan in the two machine flow shop with a *resumable* availability constraints.
- $F2|nr - a(M_j)|Cmax$: Minimizing the makespan in the two machine flow shop with a *non-resumable* availability constraints on the machine M_j.

4.5 Industrial Context

One of the motivations of this research is the industrial relevance of the hybrid flow shop environment. HFS is often found in many manufacturing systems. namely: the chemistry industry [4], textile industry [11], the pharmaceutical industry [5], glass manufacturing [27] and printed circuit board [21]. In this thesis we are interested by the production of Integrated Circuits (IC) in the semiconductor industry.

ICs are constructed on a single-crystal silicon "wafer". This is known as the "substrate". Photolithography is used to mark different areas of the substrate to be doped or to have polysilicon or aluminum tracks sputtered on them. Each device is tested, before packaging. The wafer is then diced into small rectangles called die. The die is then connected into a package using gold or aluminum wires which are welded to "pads", usually found around the edge of the die. After packaging, the devices go through final test on very expensive automated testers, which account for over 25 percent of the cost of fabrication. In the most advanced processes, the wafers exceed 30 centimeters in diameter. Generally, the whole assembly process can be classified into two parts: front

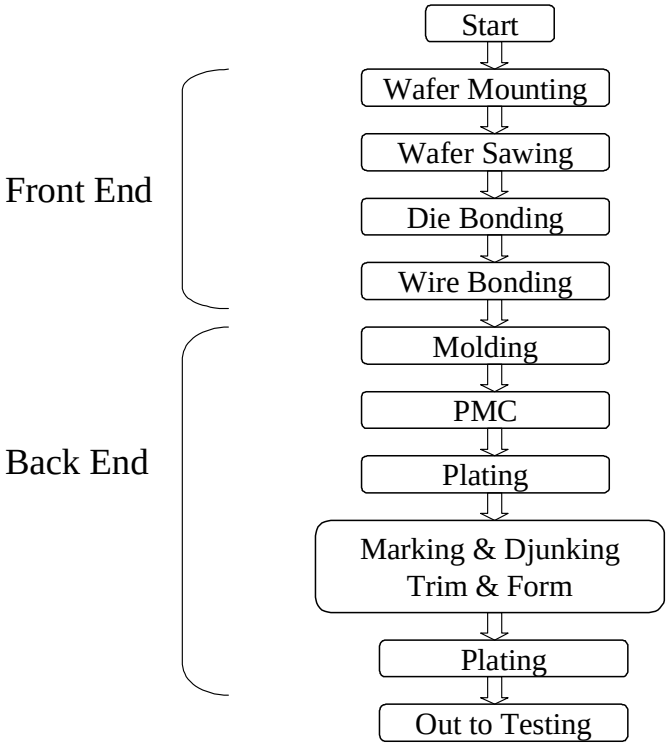

Figure 4-3: IC Assembly Process Flow.

end and back end (Figure4-3). The front end includes the wafer mounting, wafer sawing, die bounding processes anf the back end includes the molding, postmold curing, plating, marking, dejunking and trimming & forming processes ([13], [16] and [17]).

IC packaging scheduling environment is a hybrid flow shop (Figure4-4 and Figure4-5). Multiple parallel machines are involved at each stage of production flow. The number of machines on each stage can change between that stage and its adjacent stage from 1 to m, m to 1 or m_1 to m_2. The machines may be either identical or non-identical. Among all the stages the wire bonding and molding are the most important stages.

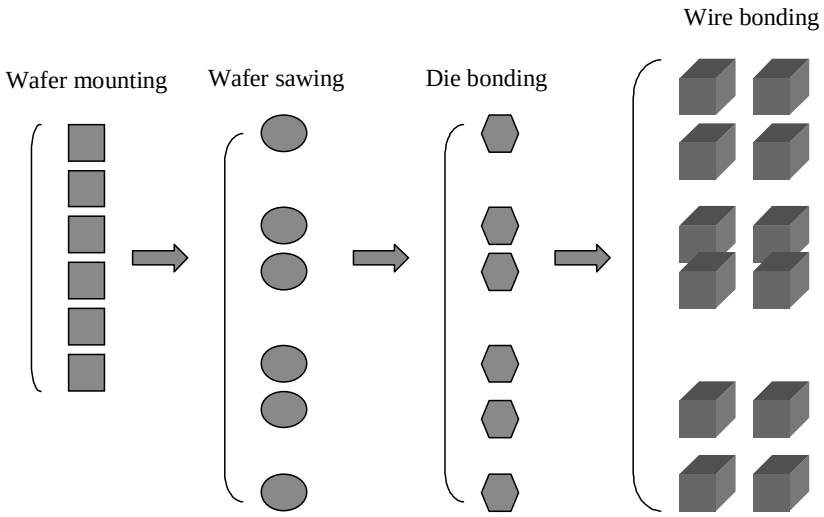

Figure 4-4: Front end assembly machines

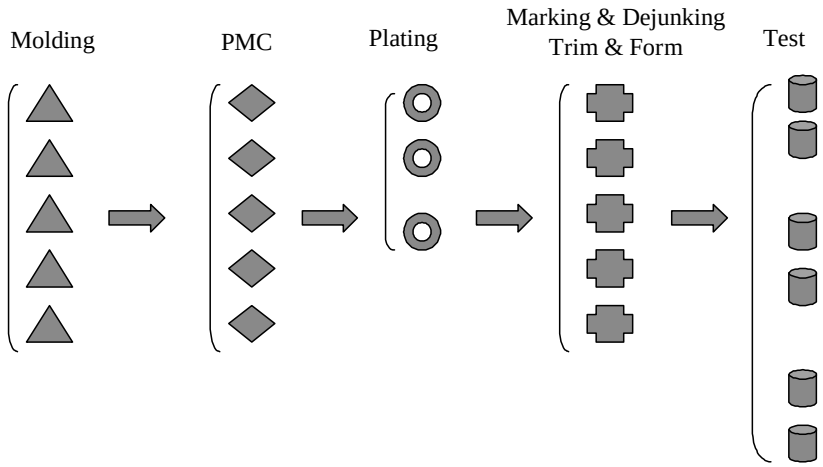

Figure 4-5: Back end assembly machines.

Bibliography

[1] Adler, L., Fraiman, N., Kobacker, E., Pinedo, Plotnicoff, M. J.C. and Wu, T.P., (1993). "BPSS: a scheduling support system for the packaging industry". *Oper. Res. 41* ,641–648.

[2] Allaoui, H., Artiba, A. and Riane, F., (2002). "Scheduling jobs on a single machine subject to breakdowns". *In 12th MINI EURO conference proceedings.* Brussels, Belgium.

[3] **Allaoui, H.**, Artiba, A. and Riane, F., (2003). "Scheduling of two machine flow shop with availability constraints". *In 18th ORBEL conference proceedings.* Brussels, Belgium.

[4] Artiba, A. and Riane, F., (1998). "An application of a planning and scheduling multi-model approach in the chemical industry". *Comp in Industry 36, Issue 3-1,* 209-229

[5] Artiba, A., and Tahon, C.,(1992). "Production planning knowledge-based system for pharmaceutical manufacturing lines". *Euro J Oper Res 61, Issues 1-2, 25* , 18-29 .

[6] Ben Daya, M., (1999). "Integrated Production, Maintenance, and Quality Model for Imperfect Processes". *IIE Transactions, 31* , 491-501.

[7] Ben Daya, M. and Hariga, M., (1998). "A Maintenance Inspection Model: Optimal and Heuristic Solutions", *Inter J Quality and Reliability Management, 5* , 481-488.

[8] Ben Daya, M. and Makhdoom, M., (1998). "Integrated Production and Quality Model Under Various Preventive Maintenance Policies". *J. Oper Res Soc, 49*, 840-853.

[9] Bitran, G.R. and Tirupati, D., (1988). "Planning and Scheduling for Epitaxial Wafer Production Facilities". *Oper Res 36, No. 1*, 34-49.

[10] Bitran, G.R. and Tirupati, D., (1988). "Development and Implementation of a Scheduling System for a Wafer Fabrication Facility". *Oper Res 36, No. 3*, 377-395.

[11] Botta, G.V. "Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness". *Inter J Prod Economics 64 (1-3)*, 101-111.

[12] Chevalier, G., Barrier, J. and Richard, P., (1996). "Production planning in the glass industry". *Revue VERRE 25*, 27–29.

[13] Connors, D., Feigin, G. and Yao, D., (1994). "Scheduling Semiconductor Lines Using A Fluid Network Model". *IEEE Trans Rob Auto 10, No. 2*, 88- 98.

[14] Dedopoulos, I.T. and Shah, N., (1995). "Optimal short-term scheduling of maintenance and production for multipurpose plants". *Ind Eng Chem Res 34* , 192–201.

[15] Dramaix, F., Artiba, A., **Allaoui., H.**, (2002). "Ordonnancement de type flowshop hybride : utilisation d'algorithmes stochastiques". Mémoire d'ingénieur de gestion. Catholic University of Mons, Belgium.

[16] Duenyas, I., Fowler, J.W and Schruben, L. W., (1994). "Planning and Scheduling in Japanese Semiconductor Manufacturing". *J Man Syst 13-5.*

[17] Fargher, H. E., Kilgore, M. A., Kline, P. J. and Smith, R. A., (1994). "Planner And Scheduler For Semiconductor Manufacturing". *IEEE Trans Semi Man, 7-2*, 117-126.

[18] Fortemps, Ph., Ost, Ch., Pirlot, M., Teghem, J. and Tuyttens, D., (1996). "Using metaheuristics for solving a production scheduling problem in a chemical firm. A case study". *Inter J Prod Econ 46-47*, 13-26.

[19] Gertsbakh, I.B., (1977). "In Models of Preventive Maintenance". Elsevier.

[20] Gupta, J.N.D., (1988). "Two stage hybrid flowshop scheduling problem". *J. Oper. Res. Soc. 39,* 359–364.

[21] Jin, Z.H., Ohno, K., Ito, T. and Elmaghraby, S.E., (2002). "Scheduling hybrid flowshops in printed circuit board assembly lines,". *POM 11*, 216-230.

[22] Kubiak, W., Blazewicz,.J., .Formanowicz, P., and Schmidt, G. (2002). "Two-machine flowshop with limited machine availability", *Euro.J.Of.Oper.Res. 136*, 528-540.

[23] Lee, C.-Y., Lei, L. and Pinedo, M., (1997). "Current trends in deterministic scheduling". *Ann of Oper Res 70*, 1–41.

[24] Lee, C-Y, Cheng, T. C. E. and Lin, B. M., (1993). "Minimizing the Makespan in the 3-Machine Assembly-Type Flowshop Scheduling Problem". *Management Science, 39-5*, 616-625.

[25] Lee, C.-Y.,.(1997). "Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint", *Oper.Res.Lett. 20*, 129-139.

[26] Lee, C.-Y. (1996). "Machine scheduling with an availability constraint", *J.Global Optimization.9,* 363-382.

[27] Paul, R.J., (1979). "A production scheduling in the glass container industry". *Oper. Res. 22*, 290–302.

[28] Pistikopoulos, E.N., Vassiliadis, C.G., Arvela, J. and Papageorgiou, L.G., (2001). "Interactions of maintenance and production planning for multipurpose process plants a system effectiveness approach". *Ind Eng Chem Res 40*, 3195–3207.

[29] Tan, J.S. and Kramer, M.A., (1997). "A general framework for preventive maintenance optimization in chemical process operations". *Comp Chem Eng 21-12*, 1451–1469.

[30] Tsubone, H. Ohba, M., Takamuki, H. and Miyake, Y., (1993). "Production scheduling system in the hybrid flow shop". *Omega 21-2*, 205.

Chapter 5

LITERATURE REVIEW

5.1 Introduction

The hybrid flow shop is a non-trivial problem. Most of the works explore three different issues: computational complexity, modeling criteria and constraints and solution methods. In terms of complexity hybrid flow shop scheduling problems can be roughly grouped into three categories: (1) Two stage hybrid flow shop, (2) Three stage hybrid flow shop and (3) k-stage hybrid flow shop. In the theoretical research most of the literature on HFS scheduling deal with single criterion problems. Among which, the minimization of makespan is the most widely used criterion. Apart from the makespan, other objectives in HFS include the minimization of the maximum tardiness, the total flow time, the sum of completion time. It is apparent that there is a gap between the literature of scheduling and the real life industry. The hybrid flow shop scheduling problems generally are considered with a simplistic way. For example researchers start to give more attention for maintenance or availability constraints in a single machine, flow shop and parallel machines problems but not yet in the hybrid flow shop problems with more than one stage and more than one machine in each stage. In terms of solution methods to HFS scheduling problems, heuristics and Branch and Bound are the two mostly used. The HFS scheduling problems will be reviewed in section 5.2 from the point of view of com-

plexity, exact methods and heuristics. The section 5.3 will focus on the scheduling with availability constraints.

5.2 The Hybrid Flow Shop

5.2.1 Complexity

The most of the work in the hybrid flow shop scheduling literature in term of the complexity is down on the single and two stage hybrid flow shop. While interesting results have been obtained for these cases, there has been less work on the k-stage ($k \geq 3$). For the single stage hybrid flow shop which is the parallel machine problem, Karp [59] has proved that the problem with only two machines to minimize the makepsan without preemption $P2|\ |C_{\max}$ is NP-hard in the ordinary sense. The only variant of the hybrid flow shop problem solved in polynomial time is the classical two machine flow shop $F2\,|\,.|\,C_{\max}$ solved by Johnson [58]. Garey et al [37] have shown that the problem $F3\,|\,.|\,C_{\max}$ is NP-hard in the strong sense. It has been shown by Hoogveen [53] that the problem $F2(P)\,|\,.|\,C_{\max}$ is NP-hard in the strong sense even if there is only one machine on the first stage and two machines on the second stage for both cases preemption and non-preemption.

5.2.2 Exact Methods

The Branch and Bound methods have been widely used in hybrid flow shop for finding an optimal solution. The B&B algorithm of Salvador [102], and Rajendran and Chaudhuri [95] can be used to solve two-stage HFS in minimizing makespan with two parallel identical machines on the first stage and only one machine the second stage. Brah and Hunsucker (Brah) give B&B algorithm to solve k-stage ($k \geq 3$) HFS problems. This algorithm will be modified after by Portman et al. [93]. The B&B approaches have not been tried in real world application because when the machine environment is complicated, it

is difficult to apply a B&B algorithm to obtain optimal solution since its computational time is moderate only for a small size instances.

5.2.3 Two Stage HFS

We will review works in the two stage hybrid flow shop literature according six categories. The first category involves single machine on the first stage and parallel identical machines on the second stage ($m_1 = 1, m_{2(I)} = m > 1$). Several studies have been reported for the second category with parallel identical machines on the first stage and one machine on the second stage ($m_{1(I)} = m > 1, m_2 = 1$). The third category consists on one machine on the first stage and non-identical machines on the second stage ($m_1 = 1, m_2 = m > 1$). The forth category involves parallel non-identical machines on the first stage and only one machine on the second stage ($m_1 = m > 1, m_2 = 1$). In the fifth category we find identical machines on both stages ($m_{1(I)} > 1, m_{2(I)} > 1$). In the last category the problem studied is to deal with non-identical machines on both stages ($m_1 > 1, m_2 > 1$).

In the first category Sriskandarajah and Sethi [106] deal with the problem of minimizing the makespan. They show that an arbitrary sequence has a worst case performance ($wcpb$) ratio of $3 - \frac{1}{m}$. They develop a heuristic based on Johnson's rule and show that its $wcpb$ is equal to $3 - \frac{3}{m} + \frac{1}{m^2}$. Gupta et al. [46] develop two heuristics to find an acceptable solution to minimize the makespan. In the second category Chen [48] considered Gupta's heuristics for the two stage hybrid flow shop with a single machine at the second stage and determines its $wcpb$ to be of $3 - \frac{2}{m}$. Gupta [45] develops an efficient heuristic algorithm for finding an approximate solution. He proposes using Johnson's rule to first sequence the jobs using only one machine at each stage, then assigning jobs to the machines at the first stage in order to minimize the additional idle time incurred on the second stage. His computational experiments are limited to only two machines at the first stage. Chen [29] classifies the heuristics for this problem into three classes and perform some empirical comparisons among selected heuristics in three classes.

In the third category Narasimhan et al [83] develop a heuristic solution procedure

known as the Cumulative Minimum Deviation (CMD) rule. It is proved to be better than SPT and LPT in decreasing the machine idle time and in-process waiting on the second stage. In the forth category Elmaghraby and Soewandy [34] propose a new heuristic based on viewing the second stage as sequencing jobs with 'ready times' that are secured from optimizing processing the jobs optimally on the first stage. Gupta [45] proposes four heuristics. The two first heuristics H_1 and H_2 have a *wcpb* of 2, and the two last heuristics have a *wcpb* of $2 - \frac{1}{m}$.

Among works on the fifth category we can cite the work of Sriskandarajah and Sethi [106]. They consider the problem when $m_1 = m_2$.They show that Johnson's sequence has a *wcpb* of $3 - \frac{1}{m}$ and their heuristic has an error bound between $\frac{7}{3} - \frac{2}{3m}$ and $3 - \frac{1}{m}$. The contributions of Langston [62] may be summarized into three . First he shows that if the sequence is randomly generated, then the *wcpb* is bounded by $3 - \frac{1}{m}$. Second if the jobs are sequenced according to the SPT rule with respect to $p_{i,1}$, then the *wcpb* is bounded by $\frac{5}{2}$. Third he demonstrates that the *wcpb* will be improved to 2 if the sequence is arranged according to the SPT rule with respect to $p_{i,2}$. Lee and Vairaktarakis [65] develop a heuristic H that has an error bound of $2 - \frac{1}{\max(m_1,m_2)}$. This bound extends a recent bound for the case $(m_1 = 1, m_2 = m > 1)$ and significantly improves all other results for some special cases of the two stage hybrid flow shop.

The general case of the last category involving many machines either different or identical at both stages has been addressed by Narasimhan and Mangiameli [84] and Paul [90]. Paul examines the production scheduling problems in glass container industry in which there are four stages and several unrelated machines in each stage. The author chooses the first two stages to develop the scheduling and formulates it as a job shop problem. The objectives are to minimize the number of tardy jobs and the average tardiness. To find a solution, a simulation method is adopted to test dispatching rules. Narasimhan and Mangiameli [84] study a case where there are several identical machines in the first stage and uniform machines in the second.

5.2.4 Three Stage HFS

Adler et al [1] consider the problem of the three-stage problem with parallel non-identical machines at stage 1, 2 and 3 ($m_1 > 1, m_2 > 1, m_3 > 1$) in Bagpak Production Scheduling System (BPSS) which is a scheduling support system for plants that produce multiply paper bags. The production process consists of three stages, but not all orders have to go through all three stages. They try to solve the compromise among three objectives to minimize: (i) the sum of tardiness; (ii) the sum of setup times; and (iii) the work-in-process inventory. Riane et al. [98] focuses on a three stage hybrid flow shop to minimize the makespan with one machine in each stage 1 and 3, and two machines in the second stage. The two machines on the second stage are dedicated (i.e. It is known beforehand on which of the two machines each job will be processed). They propose two heuristics, one based on the dynamic programming and the second one on the branch and bound algorithm. They evaluate the performance of these heuristics by an experimental study. Jin et al [57] treat a three stage HFS for the production of printed circuit board. They propose a global procedure that utilizes genetic algorithm and three subproblems to minimize the makepsan. The performance of the procedure is evaluated via an experimentation study.

5.2.5 k-Stage HFS

The k-stage problem with parallel identical machines at each stage ($m_{1(I)} > 1, m_{2(I)} > 1, ..., m_{k(I)} > 1$) is studied by Wittrock ([112] and [113]). He develops heuristics to minimize makespan & amount of buffer space used and to minimize makespan & queuing in the buffers respectively. On the other hand Zhou et al. consider the k-stage problem with parallel non-identical machines at each stage ($m_1 > 1, m_2 > l, ..., m_k > 1$). They propose a concept of synthetic knowledge and a heuristic node-path intelligent search method. The objective they use is to minimize the sum of all machines' changeover number. The most general heuristic for the k-stage HFS problem with parallel identical machines is provided by Lee and Vairaktarakis [65]. They show that the *wcpb* r of their

heuristic is:

$$r \leq k - \frac{1}{\max\{m_1, m_2\}} - \frac{1}{\max\{m_3, m_4\}} - ... - \frac{1}{\max\{m_{k-1}, m_k\}}.$$

For some special cases of the k-stage HFS problem, this bound improves significantly all the previous results.

5.2.6 Approximation Scheme

Schuurman and Woeginger [105], Hochbaum and Shmoys [52], Hall [51] and Williamson et al [114] investigate the new scheduling field called Approximation Scheme to determine the approximability behavior of the problem $F(P) \mid \mid Cmax$. Their known results on the approximability of the hybrid flow shop problem are summarized in this table [105].

		Number of machines per stage		
		=1	Constant	Arbitrary
k stages	=1	Trivial	FPTAS	PTAS
	=2	Polynomial	PTAS	PTAS
	Constant ≥ 3	PTAS	PTAS	Open
	Arbitrary	∄PTAS	∄PTAS	∄PTAS

The single-stage flow shop is the ordinary multiprocessor scheduling problem $P \mid \mid Cmax$. In case the number of machines is constant, Sahni [101] proves that the problem possesses a pseudopolynomial solution algorithm that can be used to construct an FPTAS. In case the number of machines is part of the input, the problem is strongly NP-hard, and hence the PTAS of Hochbaum and Shmoys [52] is the best-possible approximation result unless P=NP. The two-stage flow shop can be solved in polynomial time if $m_1 = m_2 = 1$ [58]. For the cases where the number k of stages and the number of machines per stage all are constants, Hall [51] constructs an PTAS. Schuurman and Woeginger [105] construct an PTAS for the problem of two stages and an arbitrary number of machines per stage. Determining the approximability behavior of the k-stage HFS $F(P) \mid \mid Cmax$ where $k > 3$ is a constant and where the number of machines is part

of the input, is still an open [51] question in the area of hybrid flow shop. Finally, for the case where the number of stages is part of the input, Williamson et al. [114] prove that the existence of a polynomial time approximation algorithm with worst-case ratio less than $\frac{5}{4}$ would imply $P = NP$.

5.3 Scheduling with Availability Constraints

Generally the limited availability of machines are due to the maintenance constraints (preventive and breakdowns). It may also result from other reasons. An example is the case of preschedules which exist mainly because most of the real world resource planning problems are dynamic. Thus The machines unavailability can happen when machines continue to process unfinished jobs that were scheduled in the previous planning period. Hence, the machine is not available at the beginning of the planning period. In this section we review all works related to this area in general situations and according two categories of problems: the first one contains deterministic problems which involve among other applications the preventive maintenance and the second one contains stochastic problems taking on consideration breakdowns.

The machine scheduling with availability constraints is an important topic in scheduling (see, for example, Blazewicz [13] and Pinedo [92]) and has attracted much attention recently (see the surveys by Lee et al. [68] and Schmidt [103]). The non-availability consideration adds complexity to any scheduling problem.

5.3.1 Deterministic Case

Fixed Non-Availability Intervals

Single Machine Problems

The Resumable Case Lee [66] deals with a single machine problem with different performance measures assuming that there is only one non-availability interval. He shows

that an arbitrary sequence is optimal for $1|r-a|C_{\max}$, the SPT sequence is optimal for $1|r-a|\sum C_i$, and the EDD sequence solves $1|r-a|C_{\max}$ optimally. That is, the algorithms are the same as those for the classical problems without availability constraints. For $1|r-a|\sum U_i$, the Moore-Hodgson Algorithm with slight modification is still optimal. However, for $1|r-a|\sum w_iC_i$, the WSPT algorithm is not optimal unless there is one job, J_i, with a completion time equal to the starting time of the non-availability interval ($C_i = s$) in the WSPT solution. He demonstrates that this problem is NP-hard even if $w_i = p_i$ for all i, and $\frac{F_w(WSPT)}{F_w^*}$ can be arbitrarily large even if $w_i = p_i$ for all i, where $F_w(WSPT)$ is the weighted total completion time obtained by applying WSPT to the problem and F_w^* is the optimal weighted total completion time. He develops a heuristic approach to solve the problem with an error-bound analysis and proposes a Dynamic Programming to solve the problem optimally and hence it was deemed NP-hard in the ordinary sense.

The Non-Resumable Case For the non-resumable case, Adiri et al [2] and Lee and Liman [64] show that $1|nr-a|\sum C_i$ is NP-hard. Lee and Liman prove that applying SPT to $1|nr-a|\sum C_i$ will have $\frac{F_{SPT}}{F^*} \leq \frac{9}{7}$, where F_{SPT} is the total completion time by applying SPT to the problem and F^* is the optimal total completion time. It is also shown by Lee [66] that the problem $1|nr-a|C_{\max}$ is NP-hard. He establishes that the problem becomes NP-hard in the strong sense if there are multiple non-availability intervals as the problem can be transformed from the Three-Partition problem. Since $1|nr-a|C_{\max}$ is NP-hard it implies that $1|nr-a|L_{\max}$ and $1|nr-a|\sum U_i$ are also NP-hard. It is shown that if we apply the Moore-Hodgson Algorithm to solve $1|nr-a|\sum U_i$ then the maximum deviation of the number of tardy from optimal is one. The error of applying EDD to solve $1|nr-a|L_{\max}$ is at most $p_{\max}$. It is clear that $1|nr-a|\sum w_iC_i$ is NP-hard and $\frac{F_w(WSPT)}{F_w^*}$ can be arbitrarily large even if $w_i = p_i$ for all i. Leon and Wu [77] study one machine scheduling with multiple non-availability intervals with ready-time constraints on the jobs to minimize the maximum lateness. A branch and bound algorithm is proposed to

solve the problem optimally. Saddfi et al. [100] give a $\frac{20}{17}-$approximation algorithm for the problem $1|nr-a|\sum C_i$.

Parallel Machine Problems

The Resumable case The classical parallel machine scheduling problem without an availability constraint, $Pm|\ |C_{\max}$, is NP-hard. Hence $Pm|r-a|C_{\max}$ and $Pm|nr-a|C_{\max}$ are both NP-hard. The problem in which each machine has at most one non-availability interval is considered by Lee [63]. He shows that if $s_j > 0$ for all j then $\frac{C_{LPT}}{c^*}$ can be arbitrarily large even for the two machine problem. This implies that no polynomial approximation scheme exists unless $P = NP$. Lee [63] applies the following two versions of LPT to the problem $Pm|r-a|C_{\max}$:

- LPT1: Assigning a job to the least loaded machine.
- LPT2: Assigning a job on the top of the list to a machine such that the finishing time of that job is minimized.

He shows that $\frac{C_{LPT1}}{c^*}$ can be arbitrarily large even if there two machines. However, $\frac{C_{LPT2}}{c^*} \leq \frac{3}{2} - \frac{1}{2m}$, if we assume that there is at least one machine always available. A pseudo-polynomial dynamic programming algorithm is provided to solve $P2|r-a|\sum w_iC_i$ optimally since it is a NP-hard in the ordinary sense.

The Non-Resumable Case Lee (Lee) studies the problem $Pm|nr-a|C_{\max}$ when some machines may not be available at time zero. He shows that the performance *wcpb* of the LPT heuristic is of $\frac{3}{2} - \frac{1}{2m}$. Using the modified heuristic MLPT, he shows that the performance is bounded by $\frac{4}{3}$. Lee and Liman [64] study the problem $P2|nr-a|\sum C_i$ with $s_1 = s_2 = \infty$ (one machine is always available while the other one is not available from s_2 to infinity). They show that the problem is NP-hard. They also provide an SPT-based heuristic with $\frac{F_H}{F^*} \leq \frac{3}{2}$. Mosheviov [82] deals with the problem $Pm|nr-a|\sum C_i$ assuming that M_j is available only in the interval $[x_j, y_j]$ and shows that SPT is asymptotically

optimal as the number of jobs approaches infinity. For $Pm|nr-a|C_{\max}$, Lee [63] shows that $\frac{C_{LS}}{C^*} \leq m$ under the assumption that at least one machine is always available ($s_j = \infty$ for some j), where LS denotes the List Scheduling, which is defined as "Given an arbitrary order of jobs, assign the job to the machine such that the job can be finished as early as possible. He also shows that the *wcpb* of LPT is of $\frac{m+1}{2}$.

Flow Shops Problems

The Resumable Case Lee (Lee) shows that both $F2|r-a(M_1)|C_{\max}$ and $F2|r-a(M_2)|C_{\max}$ are NP-hard, where $r-a(M_1)$ and $r-a(M_2)$ indicate that there is only one non-availability interval located in machine 1 and machine 2, respectively. If $s_1 = s_2 = 0$ then Johnson's algorithm is optimal for $F2|r-a|C_{\max}$. He also shows that for $F2|r-a(M_1)|C_{\max}$, the *wcpb* of Johnson's heuristic is equal to 2. He provides two heuristics called H1 and H2 with respectively $\frac{C_{H1}}{c^*} \leq \frac{3}{2}$ and $\frac{C_{H2}}{c^*} \leq \frac{4}{3}$. He notes that the problem is irreversible, an important characteristics that is distinct from the classical flow shop problem. Cheng and Wang [30] provide an algorithm for $F2|r-a(M_1)|C_{\max}$ with an improved performance, $\frac{C_H}{c^*} \leq \frac{4}{3}$. Kubiak et al. [61] show that no polynomial-time algorithm with a fixed worst case performance ratio exists unless $P = NP$. They construct a branch and bound algorithm to solve optimally the problem with multiple non-availability intervals. Blazewicz et al. [16] use local search based heuristic algorithms for the problem $F2|r-a(M_1)|C_{\max}$. Braun et al. [19] derive sufficient conditions for the optimality of Johnson's permutation in the case of one or more non-availability intervals. They show that usually Johnson's permutation remains optimal in the case of non-availability intervals.

The Non-Resumable and Semi-Resumable Cases Lee [71] studies $F2|sr1-a|C_{\max}$ where an availability constraint is imposed only on one machine as well as being imposed on both machines. The problem is clearly NP-hard because a special case, $1|nr-a|C_{\max}$, is already NP-hard. Furthermore, the problem with multiple non-

availability intervals strongly NP-hard as its special case, the one machine problem, is already strongly NP-hard [66]. Lee [71] provides a pseudo-polynomial dynamic programming algorithm to solve the problem $F2|sr1-a(M_1)|C_{\max}$ optimally in which the semi-resumable availability constraint is imposed on machine 1. He shows that $wcpb$ of Johnson's algorithm is equal to 2. Thus, both the resumable and non-resumable cases have the same error bound and is independent of the value of α. On the other hand, if we apply Johnson's algorithm to the problem $F2|sr1-a(M_2)|C_{\max}$ then $\frac{C_{JA}}{C^*} \leq \max\{\frac{3}{2}, 1+\alpha\}$ and the bound is tight. He develops an improved heuristic with $wcpb \leq \frac{3}{2}$. It is also shown by Lee (Lee) that for the problem, where an availability constraint is imposed on both machines, $F2|sr1-a(M_1, M_2)|C_{\max}$, is NP-hard even if $s_1 = s_2 = s$, and $t_1 = t_2 = t$. For such a special case, $s_1 = s_2 = s$, and $t_1 = t_2 = t$ we have $\frac{C_{JA}}{C^*} \leq 1 + \alpha$.

Non-Availability Intervals as Decision Variables

Single Machine Problems If the starting of the non-availability interval is a decision variable, i.e., it is decided jointly with the job scheduling, Qi et al. [94] show that the problem is NP-hard in the strong sense if there are multiple such intervals. They also provide a branch and bound algorithm for solving the problem in the non-resumable case. Graves and Lee [50] deal with the problem in the Type-II semi-resumable case. They assume that the time period between two consecutive maintenance periods cannot exceed a fixed number, say T. That is, the maintenance must be performed within a fixed period, T, and the time for the maintenance is a decision variable. Graves and Lee prove that there exists an optimal solution such that no more than $2m-2$ maintenance periods are required in any $1|sr2-a|$ problems with regular performance measures if $\sum_{i=1}^{n}(q_i + p_i) \leq mT$, for some positive integer m, where q_i is the setup time of job i. Hence, for any problem with $\sum_{i=1}^{n}(q_i + p_i) \leq 2T$, there are at most two maintenance periods required. For the two problems $1|sr2-a|\sum w_iC_i$ and $1|sr2-a|L_{\max}$ they studies two scenarios concerning the planning horizon:

- When the planning horizon is not short in relation to T : they show that both of problems are NP-hard and they provide a pseudo-polynomial time dynamic programming algorithms.

- When the planning horizon is short in relation to T : they show that the problem of minimizing the total weighted completion times is NP-hard, while the SPT gives the optimal solution to minimize the total completion time and EDD guaranties the optimal solution for minimizing the maximum lateness.

Parallel Machine Problems Lee and Chen [72] study the problem of m parallel machines. Their objective is to schedule jobs and maintenance activities so that the total weighted completion time of jobs is minimized. Two cases of problems are studied:

- $Pm|nr-a, ind|\sum w_iC_i$: there are sufficient resources such that different machines can be maintained simultaneously if necessary. Hence, the machines are treated independently.

- $Pm|nr-a, ind|\sum w_iC_i$: only one machine can be maintained at any given time. Hence machines are dependant.

They first show that, even when all the jobs have the same weight, both cases of the problem are NP-hard. They provide a pseudo-polynomial algorithm to solve both problems when the number of machines is fixed. Since the complexity is very high (even though it is pseudo-polynomial), they propose branch and bound algorithms based on the *column generation* approach for solving both cases of the problem. The algorithms are capable of solving medium-size problems to optimality within a reasonable computational time. Their approach can also solve the general problem when at most k machines can be maintained simultaneously ($k \leq m$, and m is the total number of machines).

5.3.2 Stochastic Case

Single Machine Problems

Glazebrook ([39], [40] and [41]) investigates a single-machine problem in which the machine breakdowns and formulated it as a cost-discounted Markov decision process. Li and Cao [76] study the problem of scheduling a set of stochastic jobs on a single machine which is subject to several types of breakdowns according to different probabilities. After the machine fails, jobs' processing time, uptime and repair time of the machine may be different from the original period. They find the optimal nonpreemptive policies that minimize the following objective functions: (1) the weighted sum of the completion times, (2) the weighted number of late jobs having constant due dates, (3) the weighted number of late jobs having random due dates. These results partly generalize the conclusions obtained by Pinedo and Rammouz [91].

Lee and Leon [74] deal with the problem of scheduling jobs and a rate-modifying activity on a single machine. A rate-modifying activity is an activity that changes the production rate of the equipment under consideration. Hence the processing times of jobs vary depending on whether the job is scheduled before or after the rate-modifying activity. The decisions under consideration are when to schedule the rate-modifying activity and the sequence of jobs to optimize some performance measure. In this paer they develop polynomial algorithms for solving problems of minimizing makespan, and total completion time respectively. they also develop pseudo-polynomial algorithms for solving problems of total weighted completion time under the agreeable ratio assumption. they prove that the problem of minimizing maximum lateness is NP-hard and also provide a pseudo-polynomial time algorithm to solve it optimally.

Parallel Machine Problems

Albers and Schmidt [3] investigate an online version of a basic scheduling problem where a set of jobs has to be scheduled on a number of identical machines so as to minimize the

makespan such that new machines may be added as well. They first show that no online algorithm can construct optimal schedules. They also show that no online algorithm can achieve a bounded competitive ratio if there may be time intervals where no machine is available. Then they present an online algorithm that constructs schedules with an optimal makespan if a lookahead of one is given, i.e., the algorithm always knows the next point in time when the set of available machines changes. Finally, they give an online algorithm without lookahead that constructs schedules with a nearly optimal makespan of $C_{\max}^{OPT} + \varepsilon$ for any ϵ, if at any time at least one machine is available.

Cai and Zhou [22] deal with a stochastic scheduling on parallel machines subject to random breakdowns to minimize expected costs for earliness and tardy Jobs. The stochastic breakdowns are characterized by a Poisson process. The deadline is an exponentially distributed random variable. They derive static and dynamic optimal policies for some special cases of the problem.

Flow Shop Problems

Allahverdi and Mittenhal [6] address the problem of minimizing makespan in a two-machine flowshop when the machines are subject to random breakdowns. They first show that it is sufficient to consider the same sequence of the jobs on each machine. After providing an elimination criterion for minimizing makespan with probability 1, they show that under appropriate conditions Johnson's algorithm stochastically minimizes makespan.

Allahverdi and Mittenhal [5] consider a two-machine fowshop scheduling problem, where machines suffer random breakdowns and processing times are constant, with respect to both makespan and maximum lateness objective functions. They provide an elimination criterion in a two machine flow shop when both machines are subject to random breakdowns. They show that the LPT and SPT orders are optimal with respect to both criteria in a two machine flow shop when the first or the second machine, respectively, suffers stochastic breakdowns.

Bibliography

[1] Adler, L.B., (1993). "BPSS: A scheduling support system for the packaging industry". *J. Oper. Res. 41*, 641–648.

[2] Adiri, I., Bruno, J., Frostig, E., and Rinnooy Kan, A.H.G., (1989). "Single machine flow-time scheduling with a single breakdown". Acta Informatica 26, 679-696.

[3] Albers, S. and Schmidt, G., (2001). "Scheduling with unexpected machine breakdowns". *Disc Appl Math 110*, 85-99.

[4] Allahverdi, A., Mittenhal, J., (1994). "Scheduling on M parallel machines subject to random breakdowns to minimize expected mean flow time. *Nav. Res. Log. Quart.41.* 677-682.

[5] Allahverdi, A., (1996). "Two-machine proportionate flowshop scheduling with breakdowns to minimize maximum lateness". *Comp& Oper Res 23-10*, 909-916

[6] Allahverdi, A. and Mittenthal, J., (1998). "Dual criteria scheduling on a two-machine flowshop subject to random breakdowns". *Inter Trans Oper Res* 5-4, 317-324

[7] Arthanary, L.S. and Ramamurthy, K.G. (1971). "An extension of two machines sequencing problem". *J.Oper.Res.41,* 641-648.

[8] Artiba, A., (2003). "Performance evaluation, planning and control, of productive systems". *Inter J Prod Econ* 85-1, 1-2 .

[9] Baker, K.R., (1974). "Introduction to Sequencing and Scheduling". John Wiley & Sons.

[10] Birge, J.R. and Glazebrook, K.D.(1988). "Assessing the effects of machine breakdowns in stochastic scheduling". *Oper Res Lett* 7-6, 267-271.

[11] Birge, J.R, Frenk, J.B.G., Mittenhall, J., and Rinnooy Kan, A.H.G., (1990), "Single-machine scheduling subject to stochastic breakdowns", *Nav. Res. Log. Quart.37.* 661-677.

[12] Błażewicz, J.K. Lenstra, A.H.G. Rinnooy Kan. "Scheduling subject to resource constraints: classification and complexity". *Disc App Math 5,* 11-24.

[13] Błażewicz, J., Finke, G., Haupt, R. and Schmidt, G., (1988). "New trends in scheduling theory". *Euro J.of Oper Res 37*, 303-317.

[14] Błażewicz, J., Kubiak, W. and Szwarcfiter, J., (1991). "Scheduling unit-time tasks on flow-shops under resource constraints". *Ann Oper Res 16,* 255-266.

[15] Błażewicz, J., Dror, M., Pawlak, G. and Stecke, K., (1994). "A note on flexible flowshop scheduling with two-stages". *Foun of Comp Dec Sciences 19*, 159-172.

[16] Blazewicz, J., Breit, J., Formanowicz, P., Kubiak, W., and Schmidt, G. (2001). "Heuristic algorithms for two-machine flowshop with limited machine availability", *Omega 29,* 599-608.

[17] Błażewicz, J., Pawlak, G. and Walter, B., (2002). "Scheduling production tasks in a two stage FMS". *Inter J of Prod Res 40*, 4341-4352.

[18] Brah, S.A. and Hunsucker, J.L. (1991), "Branch and bound algorithm for the flow shop with multiple processors", *Euro.J.Of.Oper.Res. 51,* 88-99.

[19] Braun, O, Lai, T.C., Schmidt, G. and Sotskov, Y.N., (2002). "Stability of Johnson's schedule with respect to limited machine availability". *Int J Prod Res 40*, 4381-4400.

[20] Breit, J., Schmidt, G., and Strusevich, V.A. (2001). "Two-machine openshop with an availability constraint", *Oper.Res.Lett. 29,* 65-77.

[21] Brucker, P., (1998). "Scheduling Algorithms". Third Edition, Springer.

[22] Cai, X and Zhou, S., (1999). "Stochastic Scheduling on Parallel Machines Subject to Random Breakdowns to Minimize Expected Costs for Earliness and Tardy Jobs". *Oper Resch* 47, 422-437.

[23] Carlier J., Néron E., (2000). "An exact method for solving the multiprocessor flow-shop". *RAIRO Oper Res 34*, 1-25.

[24] Chen, B., (1991). "Parametric bounds for LPT scheduling on uniform processors". *Acta Math App Sinica 7*, 67–73.

[25] Chen, B., (1993). "A note on LPT scheduling". *Oper Res Lett 14*, 139-142.

[26] Chen, B., (1994). "Scheduling multiprocessor flow shops". Advances in Optimization and Approximation (D.-Z. Du and J. Sun, Eds.), Kluwer 1-8.

[27] Chen, B., (1995). "Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage". *J of Oper Res Soc 46* , 234–244.

[28] Chen, B., Glass, C.A., Potts, C.N. and Strusevich, V.A., (1996). "A new heuristic for three-machine flow shop scheduling". *Oper Res 44*, 891-898.

[29] Chen, B., Potts, C.N. and Woeginger, G.J., (1998). "A review of machine scheduling: Complexity, algorithms and approximability". Handbook of Combinatorial Optimization (Volume 3) (Editors: D.-Z. Du and P. Pardalos), Kluwer Academic Publishers. 21-169.

[30] Cheng, T.C.E. and Wang, G., (2000). "An improved heuristic for two-machine flow-shop scheduling with an availability constraint", *Oper.Res.Lett. 26,* 223-229.

[31] Cheng, T.C.E. and Liu, Z., (2003). "Approximability of two-machine no-wait flowshop scheduling with availability constraints", *Oper Res Lett 31-4*, 319-322.

[32] Dudek, R.A.,(1992). "The lessons of flowshop scheduling research". *J. Oper. Res. 40*, 7–13.

[33] Elmaghraby, S.E. and Karnoub, R.E., (1997). "Production control in hybrid flowshops: an example from textile manufacturing". In: A. Artiba and S.E. Elmaghraby, Editors, The Planning and Scheduling of Production Systems, Chapman & Hall, UK Chap. 6.

[34] Elmaghrabi, S.E. and Soewandi, H., (2001). "Sequencing jobs on two-stage hybrid flowshop with identical machines to minimize makespan,". *IIE Trans.33*, 985-993.

[35] Elmaghrabi, S.E. and Soewandi, H., (2003). "Sequencing on two-stage flowshops with uniform machines to minimize makespan,". *IIE Trans. 35(5)*, 467-478.

[36] Frostig, E., (1991). "A note on stochastic scheduling on a single machine subject to breakdown - the preemptive repeat model". *Probab. Eng. Inform. Sci. 5*, 349–354.

[37] Garey, M.R. and Johnson, D.S. and Sethi, R., (1976). "The complexity of flow shop and job shop scheduling". *Math. Oper. Res. 1*, 117–129.

[38] Garey, M.R. and Johnson, D.S., (1979). "Computers and Intractability: A Guide to the Theory of NP-Completeness". , Freeman, New York .

[39] Glazebrook, K.D., (1984). "Scheduling stochastic jobs on a single machine subject to breakdowns". *Nav. Res. Log. Quart.31.* 251-264.

[40] Glazebrook, K.D., (1987). "Evaluating the effects of machine breakdowns in stochastic scheduling problems". *Naval Res. Logist. Quart.* 34, 319–335.

[41] Glazebrook, K.D., (1991). "On non-preemptive policies for stochastic single machine scheduling with breakdown". *Probab. Eng. Inform. Sci. 5*, 77–87.

[42] Grangeon, N., (2001). "Métaheuristiques et modèles d'évaluation pour le problème du flow shop hybride hiérarchisé: Contexte déterministe et contexte stochastique", Thesis at Blaise Pascal University, France.

[43] Graves, S.C., (1981). "A review of production scheduling". *J. Oper. Res.* 29, 646–675.

[44] Guinet, A., "A computational study of heuristics for two-stage flexible flowshops". *J. Oper. Res.* 34, 1399–1415.

[45] Gupta, J.N.D., (1988). "Two-stage, hybrid flowshop scheduling problem". *J. Oper. Res. Soc.* 38, 359–364.

[46] Gupta, J.N.D. and Tunc, E.A., (1991). "Schedules for a two stage hybrid flowshop with parallel machines at the second stage". *Int. J. Prod. Res. 29*, 1489–1502.

[47] Gupta, J.N.D. and Tunc, E.A., (1994). "Scheduling a two-stage hybrid flowshop with separable setup and removal times". *Eur. J. Oper. Res. 77*, 415–428.

[48] Gupta, J.N.D., Hariri, A.M.A. and Potts, C.N., (1997). "Scheduling a two-stage hybrid flow shop with parallel machines at the first stage". *J. Ann. Oper. Res. 69*, 171–191.

[49] Graham, R.L., (1969). "Bounds on multiprocessing timing anomalies". *SIAM J. Appl. Math. 17*, 416–429.

[50] Graves, G. H., and Lee, C.-Y., (1999). "Scheduling Maintenance and Semiresumable Jobs on a Single Machine". *Nav Res Logis 46*, 845-863.

[51] Hall, L.A., (1995). "Approximability of flow shop scheduling". *Proc. 36th IEEE Symp. on Foundations of Computer Science*, 82–91.

[52] Hochbaum, D.S. and Shmoys, D.B., (1987). "Using dual approximation algorithms for scheduling problems: theoretical and practical results". *J. ACM 34*, 144–162.

[53] Hoogeveen, J.A., Lenstra, J.K. and Veltman, B., (1996). "Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard". *Euro J. Oper. Res. 89*, 172–175.

[54] Ignall, E. and Schrage, L., (1965). "Application of the branch and bound technique to some flow-shop scheduling problems". *Oper Res 13*, 400–412.

[55] Janiak, A., (1988). "General flow-shop scheduling with resource constraints". *Inter J of Prod Res 26* , 1089–1103.

[56] Janiak, A. and Portmann, M.C., (1997). "Genetic algorithm for the permutation flow-shop scheduling problem with linear models of operations". *Annals of Oper Res 83*, 95–114.

[57] Jin, Z.H., Ohno, K., Ito, T. and Elmaghraby, S.E., (2002). "Scheduling hybrid flowshops in printed circuit board assembly lines,". *POM 11*, 216-230.

[58] Johnson, S.M. (1954). "Optimal two- and three-stage production schedules with setup times included," *Nav. Res. Log. Quart. 1,* 61-68.

[59] Karp, R.M., (1972). "Reducibility among combinatorial problems". *In R.E. Miller, J.W. Thatcher (eds.): Complexity of Computer Computations, Plenum Press*, 85-103.

[60] Kochhar, S. and Morris, R.J.T., (1987). "Heuristic methods for flexible flow line scheduling". *J. Manuf. Syst. 6*, 299–314.

[61] Kubiak, W., Blazewicz,.J., .Formanowicz, P., and Schmidt, G. (2002). "Two-machine flowshop with limited machine availability", *Euro.J.Of.Oper.Res. 136*, 528-540.

[62] Langston, M.A. (1987). "Interstage transportation planning in the deterministic flow-shop environment", *Oper.Res.35(4),* 556-564.

[63] Lee, C.-Y., (1991). "Parallel machine scheduling with nonsimultaneous machine available time", *Discrete Applied Mathematics, 30,* 53-61.

[64] Lee, C.-Y., and S. D. Liman, (1992). "Single Machine Flow-Time Scheduling With Scheduled Maintenance,". *Acta Informatica, 29*, 375-382.

[65] C.-Y. Lee and G.L. Vairaktarakis, (1994). "Minimizing makespan in hybrid flowshops". *Oper.Res.Lett. 16* . 149–158.

[66] Lee, C.-Y. (1996). "Machine scheduling with an availability constraint". *J.Global Optimization.9,* 363-382.

[67] Lee, C.-Y.,.(1997). "Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint". *Oper.Res.Lett. 20*, 129-139.

[68] Lee, C.-Y., Lei, L., and Pinedo, M. (1997). "Curent trends in deterministic scheduling". *Ann.Of.Oper.Res. 70,* 1-41.

[69] Lee, C.-Y., (1997). "Minimizing the Makespan in the Two-Machine Flowshop Scheduling Problem with An Availability Constraint,". *Oper Res Lett, 20,* 129-139.

[70] Lee, C.-Y., and Vairaktarakis, G. , (1998). "Performance Comparison of Some Classes of Flexible Flowshops and Job Shops,". *Inter J of Flex Manuf Syst, Special Issue on Manufacturing Flexibility, 10*, 379-405.

[71] Lee, C.-Y., (1999). "Two-Machine Flowshop Scheduling with Availability Constraints,". *Euro J of Ope Res, 114-2*, 198-207.

[72] Lee, C.-Y., Chen, Z.-L. (2000). "Scheduling Jobs and Maintenance Activities on Parallel Machines". *Nav. Res. Log. 47,* 61-67.

[73] Lee, C.-Y., and Lin, C.-S., (2001). "Single-Machine Scheduling with Maintenance and Repair Rate-Modifying Activities". *Eur J of Oper Res 135*, 491-513.

[74] Lee, C.-Y., and Leon, J., (2001). "Machine Scheduling with A Rate-Modifying Activity". *Eur J of Oper Res 128*, 119-128.

[75] Lenstra, H.W., (1983). "Integer programming with a fixed number of variables". *Math. Oper. Res. 8*, 538-548.

[76] Li, W. and Cao, J., (1995). "Stochastic scheduling on a single machine subject to multiple breakdowns according to different probabilities*1". *Oper Res Lett 18-2*, 81-91.

[77] Leon, V.J. and Wu, S.D., (1992). "On scheduling with ready-times, due-dates and vacations". *Nav Res Log 39*, 53-65.

[78] Linn, R. and Zhang, W., (1999). "Hybrid flow shop scheduling: a survey". *Comp & Ind Engineering.* 37, 57-61.

[79] Liu, Z. and Sanlaville, E. (1995). "Preemptive scheduling with variable profile, precedence constraints and due dates", *Euro.J.Of.Oper.Res. 114,* 420-429.

[80] MacCarthy, B.L. and Liu, J., (1993). "Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling". *Inter J of Prod Res 31*, 59–79.

[81] Mittenthall, J., and Ragavachari, M. (1993). "Stochastic flowshops", *Oper resch.30,* 148-162.

[82] Mosheiov, G., (1994). "Minimizing the sum of job completion times on capacitated parallel machines". *Math Comp Model 20*, 91-99.

[83] Narasimhan, S.L. and Panwalkar, S.S., (1984). "Scheduling in a two-stage manufacturing process". *Int. J. Prod. Res. 22*, 555–564.

[84] Narasimhan, S. and Mangiameli, P., (1987). "A comparison of sequencing rules for a two-stage hybrid flow shop". *Decis. Sci. 18*, 250–265.

[85] Nawaz, M., Enscore, E.E. and Ham, I., (1983). "A heuristic algorithm for the m-machine, n-job flow shop sequencing problem". *OMEGA 11*, 91–95.

[86] Néron E., Baptiste Ph., Gupta J.N.D, (2001). "Solving hybrid flow shop problem using energetic reasoning and global operations". *Omega 29*, 501-511.

[87] Ohno, K., Jin, Z.H. and Elmaghraby, S.E., (2002). "Scheduling hybrid flowshops in printed circuit board assembly lines", *Prod and oper Mana, 11,2,* 216-230.

[88] Oguz, C., (1997). "Two-stage flowshop scheduling with a common second-stage machine". *Comput. Oper. Res. 24*, 1169–1174.

[89] Panwalkar, S.S. and Iskander, W., (1977). "A survey of scheduling rules". *Oper Res 25*, 45–61.

[90] Paul, R.J., (1979). "A production scheduling in the glass container industry". *Oper. Res. 22*, 290–302.

[91] Pinedo, M. and Rammouz, E., (1988). "Single Machine Scheduling with Breakdown and Repair". *Prob Engin Infor Sci*, 41-49.

[92] Pinedo, M., (2002). Scheduling : Theory, Algorithms and Systems. Second Edition, Prentice-Hall.

[93] Portmann, M. -C. , Vignier, A., Dardilhac, D. and Dezalay, D., (1998). "Branch and bound crossed with GA to solve hybrid flowshops". *Eur. J. Oper. Res. 107,* 389-400.

[94] Qi, X., Chen, T. and Tu, F., (1999). "Scheduling the maintenance on a single machine". *J Oper Res Soc 50*, 1071-1078.

[95] Rajendran, C. and Chaudhuri, D., (1992). "Scheduling in n-job, m-stage flowshop with parallel processors to minimize makespan". *Int. J. Prod. Econ. 27*, 137–143.

[96] Rajendran, C. and Chaudhuri, D., (1992). "A multi-stage parallel-processor flowshop problem with minimum flowtime". *Eur. J. Oper. Res. 57*, 111–122.

[97] Rao, T.B.K., (1970). "Sequencing in the order A,B with multiplicity of machines for a single operation", *Oper.Res7,* 135-144.

[98] Riane, F., Artiba, A. and Elmaghraby, S.E., (1998). "A hybrid three-stage flowshop problem: Efficient heuristics to minimize makespan". *Euro J Oper Res 109-2*, 321-329.

[99] Riane, F., (1998). "Scheduling hybrid flow shop: algorithms and application", Thesis at Catholic University of Mons, Belgium.

[100] Saddfi, C., Blazewicz, J., Formanowicz, P., Penz, B. and Rapine, C., (2001). "A better approximation algorithm for the single machine total completion time scheduling problem with availability constraints". International Conference of Industrial Engineering and Production Management - IEPM, Quebec, Canada.

[101] Sahni, S., (1976). "Algorithms for scheduling independent tasks". *J. ACM 23.* 116–127.

[102] Salvador, M.S., (1973). "A solution to a special case of flow shop scheduling problems". In Symposium of the Theory of Scheduling and Applications, Elmaghraby, ed, 83-91.

[103] Schmidt, G. (2000). "Scheduling with limited machine availability", *Euro.J.Of.Oper.Res.121,* 1-15.

[104] Shmoys, D.B., Stein, C. and Wein, J., (1994). "Improved approximation algorithms for shop scheduling problems". *SIAM J. Comput. 23*, 617–632.

[105] Schuurman, P.and Woeginger, G.J., (2000). "A polynomial time approximation scheme for the two-stage multiprocessor flow shop problem", *Theo Comp Science, 237,* 105-122.

[106] Sriskandarajah, C. and Sethi, S.P., (1989). "Scheduling algorithms for flexible flow shops: worst and average case performance". *Euro.J.Of.Oper.Res. 43,* pp. 143–160.

[107] Sriskandarajah, C. and Sethi, S.P., (1989). "Scheduling algorithms for flexible flow shops: worst and average case performance". *Eur J. Oper. Res. 43*, 143–160.

[108] Uzsoy, R., Lee, C.Y. and Martin, V.L.A, (1994). "A review of production planning and scheduling models in the semiconductor industry. II. Shop-floor control". *J. IIE Trans. 26*, 44–55.

[109] Vignier, A., (1996). "Resolution of some 2-stage hybrid flowshop scheduling problems". *In: IEEE Inter Conf on Systems, Man and Cybernetics. Information Intelligence and Systems 4*, 2934–2941.

[110] Vignier, A., Commandeur, P. and Proust, V., (1997). "New lower bound for the hybrid flowshop scheduling problem". *In: IEEE 6th Inter Conf on Emerging Technologies and Factory Automation Proceedings*, 446–451.

[111] Vignier, A., Billaut, J.C and Proust, C., (1999). "Les problèmes d'Ordonnancement de type flowshop hybride : Etat de l'art". *RAIRO Recherche Opérationnelle / Operations Research 33-2*, 117-183.

[112] Wittrock, R.J., (1985). "Scheduling algorithms for flexible flow lines". *IBM J. Res. & Dev. 29*, 401–412.

[113] Wittrock, R.J., (1988). " An adaptable scheduling algorithm for flexible flow lines". *J. Oper. Res.* 36, 445-53.

[114] Williamson, D.P., Hall, L.A., Hoogeveen, J.A., Hurkens, C.A.J., Lenstra, J.K., Sevastianov, S.V. and Shmoys, D.B., (1997). "Short shop schedules". *Oper. Res. 45*, 288–294.

[115] Zhou, J.R., Huang, J.R. and Jiang, W.S., (1996). "Optimization production scheduling of multi-stage interrelated discrete system via synthetic knowledge". *In: Proceedings of the American Control Conference*, 724–728.

Chapter 6

SCHEDULING TWO MACHINE FLOW SHOP WITH AVAILABILITY CONSTRAINTS

6.1 Introduction

This chapter is concerned with the problem of scheduling n immediately available jobs in a flow shop composed of two machines in series with the objective of minimizing the makespan, when it is known that there shall be only one interruption in machine availability either on the first machine or on the second machine due to scheduled maintenance (deterministic case). For convenience we call the period of unavailability a gap.

The two machine flow shop (without a gap) with the objective of minimizing the makespan is perhaps the first "multi machine" scenario ever treated by researchers in the field. Its optimal solution is due to Johnson [7]. Since that date it has witnessed several extensions and variations; see Lee [11] and [12] for a comprehensive review of earlier contributions. Lee proves that the problem with one gap on only one machine is NP hard in the ordinary sense.

The processing times of job i are given by $p_{i,1}$ and $p_{i,2}$ on the first machine and

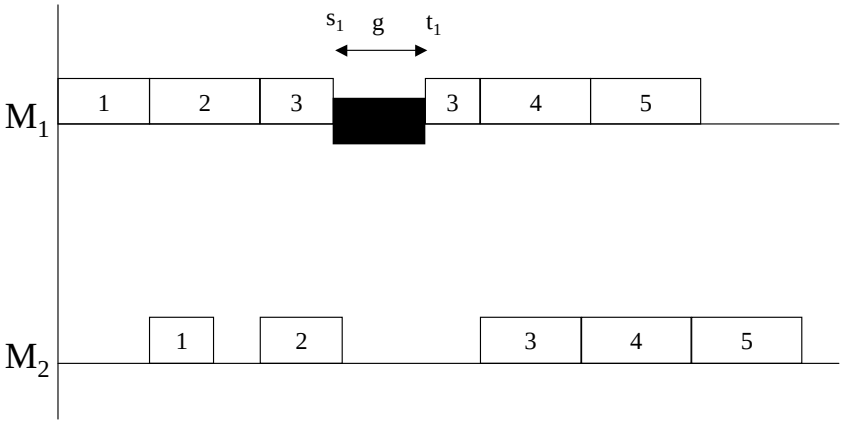

Figure 6-1: The gap on the first machine in resumable case.

the second machines respectively. We refer to the interruption as the "gap" in machine availability. The start time of the gap on machine j is denoted by s_j, its duration by g, and its termination by $t_j = s_j + g$; ($j = 1, 2$).We assume that machine j, $j = 1, 2$, is unavailable during the period from s_j to t_j $(0 \leq s_j \leq t_j)$. We assume that one machine is always available. Hence we deal with two cases of unavailability. In the first case we take only one gap in the first machine (section 6.3) and in the second case we take only one gap in the second machine (section 6.4). We also consider two types of processing : resumable and non-resumable (see in Figure6-1 an example if the gap is on the first machine in resumable case).

The contributions of this chapter are three. We prescribe the problem instances in which Johnson's rule gives the optimum solution. This is most valuable since Johnson's

procedure is computationally most efficient. We propose an improved dynamic programming model, which is independent of the processing times of the jobs, if the gap is on the first machine and we propose an efficient combinatorial approach if the gap is on the second machine. This reduces the computational burden of the search for optimality drastically. Finally we give some worst-case error bound of Johnson's rule on both cases of resumable and non-resumable.

6.2 Notations

For simplicity, we refer to *Johnson's rule* for optimizing the makespan in the absence of the gap by JR, and to the order of jobs (the sequence) resulting from applying the rule by JO. The presence of the gap may render the JO non-optimal. Still, we shall adopt the JR as our heuristic so that in the sequel mention of "the heuristic H" refers to the schedule following the JR.

Johnson's rule (JR) : Divide the n-job set into two disjoint subsets, S_1 and S_2, where $S_1 = \{J_i : p_{i,1} \leq p_{i,2}\}$ and $S_2 = \{J_i : p_{i,1} \succ p_{i,2}\}$. Order the jobs in S_1 in the nondecreasing order of $p_{i,1}$ and those jobs in S_2 in the nonincreasing order of $p_{i,2}$. Sequence jobs in S_1first, followed by S_2.

We consider n jobs to be processed in two-machine flow shop and we adopt the following notations.

A :	the set of jobs processed after the gap. A^H denotes the jobs in A when following heuristic H.
B :	the set of jobs processed before the gap. B^H denotes the jobs in B when following heuristic H.
H :	refers to Johnson's Rule (JR).
$\emptyset$:	indicates the absence of the gap.
G :	indicates the *presence* of a gap.
$C^H_{\max}(\emptyset)$:	the makespan following H when there is *no gap*, it is optimal $C^H_{\max}(\emptyset) = C^*_{\max}(\emptyset)$.
$C^H_{\max}(G)$:	the makespan following H when *there is a gap*, it may not be optimal.
$C^*_{\max}(G)$:	the *optimal* makespan when *there is a gap.*
$C_{i,j}(S_k)$:	the completion time of job i on machine j, $i \in S_k$.
$N_k(B)$:	the subset of k jobs that are to be scheduled before the gap; with $N_k(A) = N - N_k(B)$ denoting the subset of $n-k$ jobs that are to be scheduled after the gap.
$p_{i,j}$:	the processing time of job i on machine j, $j = 1, 2$.
s_j :	the start time of the gap on machine j, $j = 1, 2$.
r :	the resumable case.
nr :	the nonresumable case.
t_j :	the end of the gap, $t_j = s_j + g$, $j = 1, 2$.
S_1	$\{J_i : p_{i,1} \leq p_{i,2}\}$. In JO the jobs in this set are ordered in nondecreasing order of $p_{i,1}$
S_2	$\{J_i : p_{i,1} \succ p_{i,2}\}$. In JO the jobs in this set are ordered in nonincreasing order of $p_{i,2}$

In order to be able to refer the problem under study, we use the notation proposed by Lee [9] :

$F2|r-a(m_j)|Cmax$: Minimizing the makespan in the two machine flow shop problem

with a resumable availability constraint on machine j.

$F2|nr-a(m_j)|Cmax$: Minimizing the makespan in the two machine flow shop problem with a non-resumable availability constraint on machine j.

Lee proves in [9] an important lemma which gives a characteristic of the optimal solution either when the gap is on the first machine or on the second machine.

Lemma 15 *(Lee's [9] Lemma 2, page 132.) There exists an optimal sequence such that the jobs in the set* B *and the jobs in the set* A *are sequenced in* JO.

Proof. This assertion is proved by switching the order of two jobs in either B or A. Such a switch can only increase the makespan by the optimality of the JO of either sets. ■

6.3 The Gap on The First Machine

6.3.1 Optimality Conditions For The Johnson Order Optimality (F2|r,nr-a(M1)|C_{max})

In the two machine flow shop with availability constraints problem there are two kinds of idle time. The first one is due to the sequence of jobs denoted by I_{seq}, and the second one is due to the gap denoted by I_{gap}. The difference between the two kinds of idle time is illustrated in the two figures: Figure 6-2 and Figure6-3.

It is shown that Johnson's algorithm minimizes the sum $\sum I_{\text{seq}}$ (see [7]). Hence its optimality for the problem of two machine flow shop with a gap depends if it minimizes the second kind of idle time I_{gap} or not.

Let J^A_{JO} denote the first job completed after the gap on the first machine when the jobs are sequenced in JO.

Proposition 16 *If the completion time of the set B^H on the second machine is larger than $t_1 + p_{J^A_{JO},1}$ then* JO *is optimal.*

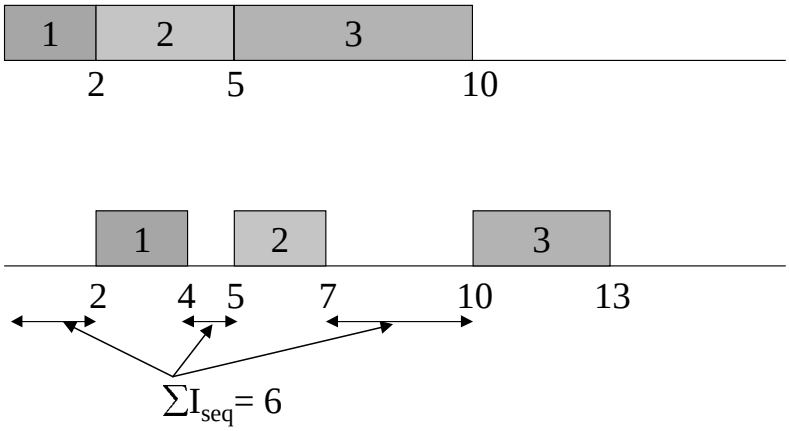

Figure 6-2: The idle time due to the sequence when there is no gap.

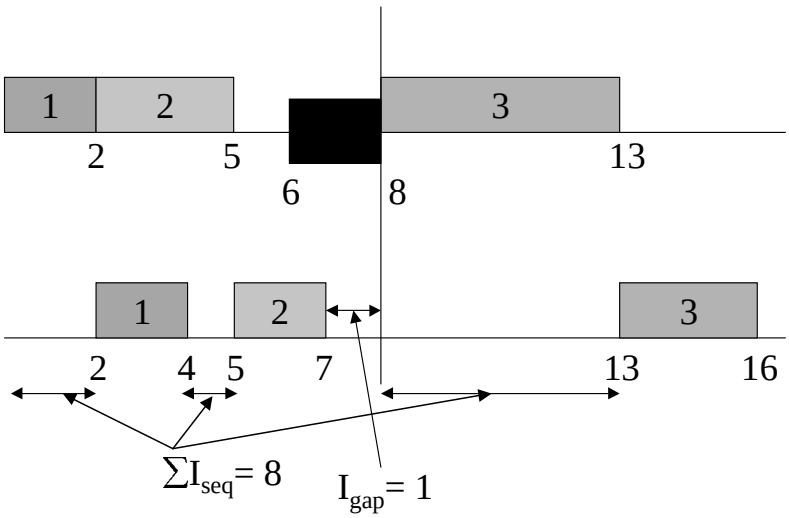

Figure 6-3: The two kinds of idle time when there is a gap on the first machine.

Proof. *The makespan under any sequence σ is given by*

$$C_{\max}(\sigma) = \sum_i p_{i,2} + \sum I_{\text{seq}} + I_{gap} \tag{6.1}$$

in which $\sum I_{\text{seq}}$ is the sum of idle time on the second machine due to the sequence of jobs, and I_{gap} the idle time due to the gap. Therefore in order to minimize the makespan one should find the sequence that minimizes the two of kinds of idle time. Now because of the completion time of the set B^H is larger than $t_1 + p_{J^A_{JO}}$,the sum $I_{gap} = 0$. Since JO *minimizes the sum of idle time $\sum I_{\text{seq}}$ then it is the optimal sequence.* ■

Proposition 17 *If the* JO *results in a set B^H that terminates on the first machine exactly at time s_1 then* JO *is optimal.*

Proof. The Proposition is proved by contradiction using a pair-wise interchange argument. Suppose any job j of A^H is inserted in B^H replacing job i that is currently in B^H, which is placed in A^H and results in a smaller makespan. Since jobs i and j were originally in JO then, by 15, job j must be placed last in B and job i must be placed first in A; see Fig.6-4. For feasibility we must have

$$p_{i,1} \geq p_{j,1} \tag{6.2}$$

Since the original sequence was according to JO, it must be true that

$$\text{(1) if} \quad i \in S_1 \text{ and } \quad j \in S_1 \Rightarrow p_{i,1} < p_{j,1}.$$

which contradicts (6.2) and job j will not fit before s_1, which violates its inclusion in the set B. Otherwise,

$$\text{(2) if} \quad i \in S_1 \text{ and } \quad j \in S_2,$$

then we must consider two eventualities. (a) $p_{i,1} < p_{,j,1}$, in which case job j will again contradict (6.2) and job j will not fit before s_1; and (b) $p_{i,1} > p_{j,1}$, in which case job

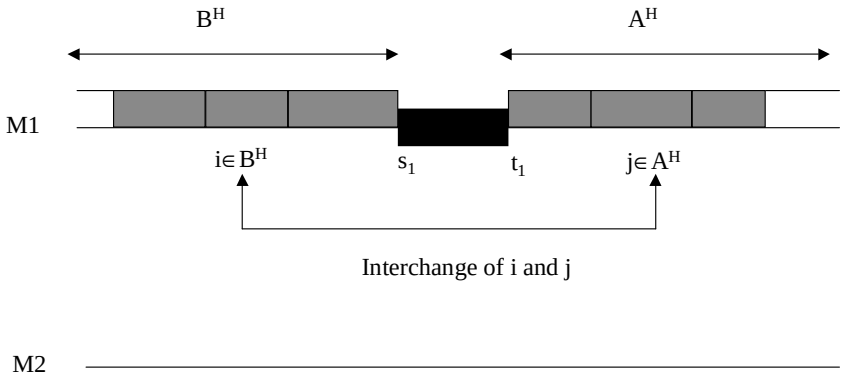

Figure 6-4: The permutation of jobs i and j.

j will fit before s_1 leaving idle time before s_1. Job i replacing j will take longer on the first machine, and $p_{i,1} > p_{j,1} \geq p_{j,2}$ (the last inequality by virtue of $j \in S_2$), whence the completion of job i on the second machine can only be delayed, which can only delay the makespan. Finally,

$$(3) \text{ if } \quad i \in S_2 \text{ and } \quad j \in S_2, \quad \text{with } \quad p_{i,2} \geq p_{j,2} \quad (\text{by } JO),$$

in which case again job i replacing j will take longer on both the first machine (by (6.2)) and on the second machine, which would delay its completion and thus delay the makespan. Hence if we permute the two jobs we can only delay the completion time of the set A and consequently increase the the makespan, which contradicts its improved performance. ■

The problem arises when $\sum_{i \in B^H} p_{i,1} < s_1$ because then one is never sure if another arrangement would not yield a better makespan.

6.3.2 A Dynamic Programming Model (F2|nr-a(M1)|Cmax)

It is easy to determine the maximal number of jobs that "fit" in the interval $[0, s_1]$: Simply order the jobs in non-decreasing order of $p_{i,1}$ and select the first n_1 jobs for which $\sum_{i=1}^{n_1} p_{i,1} \leq s_1$ but $\sum_{i=1}^{n_1+1} p_{i,1} > s_1$. Then we know that no more than n_1 jobs can be processed on the first machine before the gap. The issue now becomes: which jobs? This may be answered in one of two ways. (It is this determination that gives this problem its NP-hardness character.) One may solve a knapsack problem and *enumerate all its alternate optima*, or one enumerates all subsets of cardinality $\leq n_1$, and there are $\binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n_1} \leq 2^n - \sum_{r=0}^{n/2-1} \binom{n}{r}$. The inequality is because if $n_1 > n/2$ then we switch to consider the jobs allocated to the set A.

Consider a subset of k jobs, $k \leq n_1$ that are to be scheduled before the gap. Denote the subset by $N_k(B)$, and denote their processing time on the first machine by $P_1(N_k(B))$.

That is,

$$P_1\left(N_k\left(B\right)\right) = \sum_{i \in N_k(B)} p_{i,1}.$$

Note that if $P_1\left(N_k\left(B\right)\right) > s_1$ then this subset of jobs is deleted since it is not eligible as candidate. This implies that we may end up evaluating fewer subsets than the upper bound given above. For the remainder of this discussion we assume that attention is limited to subsets with $P_1\left(N_k\left(B\right)\right) \leq s_1$. Clearly, if the jobs in $N_k\left(B\right)$ are the only jobs executed before the gap, then the remaining jobs, denoted by $N_k\left(A\right) = N - N_k\left(B\right)$ must start after the gap since

$$N = N_k\left(B\right) + N_k\left(A\right).$$

By Lemma 15 there exists a minimal makespan schedule such that the jobs in $N_k\left(B\right)$ are ordered in JO and the jobs in $N_k\left(A\right)$ are also ordered in JO. Denote the makespan as shown above by such ordering of the jobs by $C_{\max}\left(N_k\left(B\right)\right)$. Clearly $C_{\max}\left(N_k\left(B\right)\right)$ is an upper bound $(u.b.)$ on the value of the optimum, since it is a feasible schedule. Let $C_{i,1}\left(N_k\left(B\right)\right)$ denote the completion time of job i on the first machine in subset $N_k\left(B\right)$; and define $C_{i,2}\left(N_k\left(B\right)\right)$ similarly on the second machine. Denote the completion time of the jobs in $N_k\left(B\right)$ on the second machine by $C_2\left(N_k\left(B\right)\right)$[1]. Identify the first job after the gap generically by j_a. Its start time on the first machine is t_1. Denote its start time on the second machine by $s_{j_a,2} \geq t_1 + p_{j_a,1}$. The notation is explained in Fig.6-5. In the figure there is only one line representing the first machine, but there are two lines representing the second machine, depending on the completion time of the jobs in $N_k\left(B\right)$ on the second machine relative to the completion time of job j_a on the first machine. The first line of the second machine depicts the case in which $s_{j_a,2} = t_1 + p_{j_a,1}$. The second line depicts the case in which $s_{j_a,2}$ is strictly $> t_1 + p_{j_a,1}$.

Suppose that the set $N_k\left(B\right)$ is augmented by job h which is currently in the set

[1]Thus $C_2\left(N_k\left(B\right)\right) = C_{2,l}\left(N_k\left(B\right)\right)$ where l is the last job in the set B.

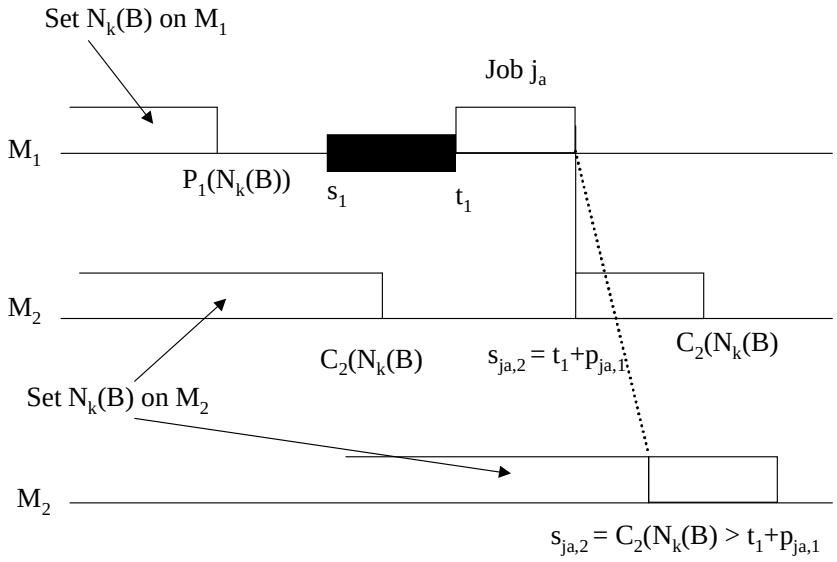

Figure 6-5: Two cases of completion time of set $N_k(B)$ on the second machine.

$N_k(A)$ so that we now have the set

$$N_{k+1}(B) = N_k(B) \cup \{h\}$$

that is supposed to be processed before s_1. Job h is accepted as an augmentation to $N_k(B)$ if $P_1(N_{k+1}(B)) \leq s_1$; otherwise it is rejected. The issue is to relate the minimal makespan $C_{\max}(N_{k+1}(B))$ to the old makespan $C_{\max}(N_k(B))$; which would enable us to write the DP extremal equation. (Recall that to achieve the minimal makespan under $N_k(B)$ for any $k \geq 2$ the jobs in $N_k(B)$ must be ordered in JO, and so are the jobs in $N_k(A)$.)

Denote the job immediately preceding job h in $N_k(A)$ generically by $h-1$, and denote the job immediately succeeding it in $N_k(A)$ generically by $h+1$. It is easy to deduce that if $N_k(B)$ is augmented by job h which is currently in the set $N_k(A)$ then the gain in the makespan, if any, is given by

$$v = s_{h+1,2}(N_k(A)) - \max\{C_{h-1,1}(N_k(A)) + p_{h+1,1}, C_{h-1,2}(N_k(A))\}.$$

Let $f(N_k(B))$ denote the minimal makespan under set $N_k(B)$. Let

$$N_{k-1}(B) = N_k(B) - \{h\}.$$

Then the DP extremal equation is given by

$$f(N_k(B)) = \min_{h \in N_k(B)} \left\{ f(N_{k-1}(B)) - v \right\}.$$

Iteration is initiated at $k = 1$ (*i.e.*, subsets containing exactly a single job, j) for which the makespan is determined following JO for the jobs in $N - \{j\}$. Iteration is terminated when all subsets of size n_1 have been considered. The unconditional optimum is secured

as

$$\min_{k=1,\cdots,n_1} \{f(N_k(B))\}. \tag{6.3}$$

The time complexity of this procedure is $O(2^{n_1} + n\log n)$.

The advantage of this model over others is that it is independent of the processing times $\{p_{i,j}\}$, $j = 1,2$. Therefore, if $n = 50$, and $n_1 = 5$, this model is of complexity $O(2^5 + 50 \times \log 50) \approx O(120)$, which is $\approx 10^{10}$ times smaller than the complexity of Lee's model ($O(n \times s_1 \times (\sum(p_{i,1} + p_{i,2})^2) \times p_{\max,1})$) [9] for the same size problem when the average processing time is 50 on either machine.

Example 18 *Consider the following set of jobs.*

Job i	***1***	***2***	***3***	***4***	***5***	***6***
$p_{i,1}$:	*2*	*6*	*10*	*4*	*10*	*4*
$p_{i,2}$:	*1*	*7*	*3*	*3*	*1*	*5*

$s_1 = 9$ *and* $g = 3$.

JR *would yield the sequence: 6,2,3,4,1,5 with makespan equal to 45. It is easy to deduce* $n_1 = 2$. *We give in the following table the dynamic programming iterations. The set of jobs which can be inserted before the gap is* $\{1,2,4,6\}$. *Observe that we enumerated only* $\binom{4}{1} + \binom{4}{2} = 10$ $(\ll 2^6 = 64)$ *subsets, and evaluated the makespan for only 8 subsets because two were infeasible.*

Stage	***Set*** B ***in*** JO	***Set*** A ***in*** JO	$C_{\max}$
1	$\{1\}$	$\{6,2,3,4,5\}$	*47*
	$\{2\}$	$\{6,3,4,1,5\}$	*43*
	$\{4\}$	$\{6,2,3,1,5\}$	*45*
	$\{6\}$	$\{2,3,4,1,5\}$	*45*
2	$\{2,1\}$	$\{6,3,4,5\}$	*41*
	$\{4,1\}$	$\{6,2,3,5\}$	*43*
	$\{6,1\}$	$\{2,3,4,5\}$	*43*
	$\{6,4\}$	$\{2,3,1,5\}$	*41*

The makespan given by dynamic programming procedure is 41 and the optimal sequences are 2,1,6,3,4,5 or 6,4,2,1,3,5.

6.3.3 A Combinatorial Approach (F2|nr-a(M1)|Cmax)

This approach may be viewed as a "shortcut" for the DP formalism discussed above. Observe that once the set $N_k(B)$ has been defined its *optimal* sequence is known, so is the optimal sequence of the complementary set $N_k(A)$. There is no need for optimization!.The makespan is easily determined, and it is retained as the "best in hand" if it is smaller than the last best in hand, else it is discarded.

A pertinent observation in this regard is that we are interested in "packing" as many jobs as possible before the time s_1, whence enumeration of subsets should start from the largest to the smallest. The superposition of simple elimination rule due to dominance (as in the branch-and-bound approach) should speed up the calculations.

Example 19 *Consider the pervious example. We knew that* $n_1 = 2$ *and that the set of jobs which can be in* B *is* $\{1, 2, 4, 6\}$. *Therefore we start with subsets of cardinality 2.*

$\{2, 1\}$	*feasible*	$C_{\max} = 41$
$\{4, 1\}$	*feasible*	$C_{\max} = 43$
$\{6, 1\}$	*feasible*	$C_{\max} = 43$
$\{4, 2\}$	*infeasible*	$\times$
$\{6, 2\}$	*infeasible*	$\times$
$\{6, 4\}$	*feasible*	$C_{\max} = 41$

Hence the optimal solution given by this approach is the sequence 2,1,6,3,4,5 or 6,4,2,1,3,5 with makespan equal to 41, as secured above.

6.3.4 Johnson's Rule As Heuristic (F2|r-a(M1)|Cmax)

Recall that application of *JR* in the absence of a gap (case $\emptyset$) is indeed optimal, $C^H_{\max}(\emptyset) = C^*_{\max}(\emptyset)$; easily secured in $n \log(n)$ time. Interestingly enough, *JR* need not be optimal

in the case of problem $F2|r - a(M1)|Cmax$. The following example illustrates this contention.

Example 20 *There are four jobs with the gap at* $s_1 = 3.5$, *of duration* $g = 1.9$, *And with the following processing times:*

Job i	***1***	***2***	***3***	***4***
$p_{i,1}$	*1*	*1.1*	*1.2*	*1.4*
$p_{i,2}$	*1.2*	*1.3*	*1.4*	*3.0*

JR *would yield the sequence as given: 1,2,3,4, with makespan equal to 9.6. Yet the optimal sequence is 1,4,2,3 for a minimal makespan of only 8. The start and completion times of the jobs on the two m/c's are shown in the following table;*

JO :

Job i	***1***	***2***	***3***	***4***
$s_{i,1}$	*0*	*1.0*	*2.1*	*3.3*
$C^H_{i,1}$	*1.0*	*2.1*	*3.3*	*6.6*
$s_{i,2}$	*1.0*	*2.2*	*3.5*	*6.6*
$C^H_{i,2}$	*2.2*	*3.5*	*4.9*	*9.6*

Opt. :

Job i	***1***	***4***	***2***	***3***
$s_{i,1}$	*0*	*1*	*2.3*	*5.4*
$C^*_{i,1}$	*1.0*	*2.4*	*5.4*	*6.6*
$s_{i,2}$	*1.0*	*2.4*	*5.4*	*6.6*
$C^*_{i,2}$	*2.2*	*5.4*	*6.7*	*8*

Therefore *JR* is considered as a heuristic, denoted by *H*. Its *wcpb* is denoted by $r(H)$, with

$$r\left(H\right) = \frac{\overline{C^H_{\max}}\left(G\right)}{C^*_{\max}\left(G\right)}, \tag{6.4}$$

where the bar on the $C^H_{\max}$ indicates that it is the worst case (maximal).

It should be evident that if the gap $s_1 < \min_i \{p_{i,1}\}$ then *JR* is optimal. Henceforth we shall assume that

$$s_1 > \min_i \{p_{i,1}\} > 0. \tag{6.5}$$

We are still concerned with the *wcpb* for the two machines flow shop with a *fixed gap* on the first machine of duration *g*, finite and known *a priori*. The makespan in the presence of the gap shall be denoted by $C^{(\rho)}_{\max}(G)$, and in its absence by $C^{(\rho)}_{\max}(\emptyset)$; $\rho =$ "H" or "$*$". We label the job that, in the case of the *stop resume* mode, would have started on

the first machine before the gap and completed after it as the "*straddling job*". Denote by J_1 the set of jobs that precede the straddling job, and by J_2 the complement of J_1. Let

$$p_{\max,1} = \max_i \{p_{i,1}\}, \quad p_{\min,1} = \min_i \{p_{i,1}\} \tag{6.6}$$

and define $p_{\max,2}$ and $p_{\min,2}$ similarly on the second machine. We also define

$$P_1 = \sum_i p_{i,1}, \quad P_2 = \sum_i p_{i,2}. \tag{6.7}$$

We state two obvious inequalities,

$$p_{\max,1} < C^H_{\max}(\emptyset). \tag{6.8}$$

$$\begin{aligned} C^*_{\max}(G) &\geq \max\{P_1 + g + p_{\min,2};\ P_2 + p_{\min,1}\} && (6.9)\\ &\geq \max\{p_{\max,1} + g + p_{\min,2},\ p_{\max,2} + p_{\min,1} + p_{\min,2}\} && (6.10) \end{aligned}$$

Lemma 21 *For problem* $F2|r - a(M1)|Cmax$,

$$C^H_{\max}(G) \leq C^H_{\max}(\emptyset) + g. \tag{6.11}$$

Proof. Denote the last job that completes on the first machine at or before time s_1 by b, and write its completion times as $C_{b,2}(\cdot)$ for simplicity. Denote the job immediately preceding it on the first machine by $b-1$ (with the understanding that it is the null job if b is the first job) and denote the job immediately following it by $b+1$. There are two cases to consider:

1. Case 1:

$$C^H_{b,2}(G) \geq C^H_{b+1,1}(G)$$

then the presence of the gap will have no impact whatsoever on the makespan and $C^H_{\max}(G) = C^H_{\max}(\emptyset) = C^*_{\max}(G)$.

2. Case 2:

$$C^H_{b,2}(G) \leq C^H_{b+1,1}(G)$$

then the delay in the makespan is at most g. ■

Evidently,

$$C^*_{\max}(G) \geq C^H_{\max}(\emptyset). \tag{6.12}$$

The following two propositions establish 2 as the bound on the error committed when using *JR*.

Proposition 22 *For problem* $F2|r - a(M1)|Cmax$, *and under the assumption that* $g \leq C^H_{\max}(\emptyset)$,

$$\left(r(H) \mid g \leq C^H_{\max}(\emptyset)\right) = 1 + \frac{g}{C^H_{\max}(\emptyset)} \leq 2. \tag{6.13}$$

Proof. Follows immediately from Lemma 21 and inequality (6.12),

$$\begin{aligned}
\left(r(H) \mid g \leq C^H_{\max}(\emptyset)\right) &\leq \frac{C^H_{\max}(\emptyset) + g}{C^*_{\max}(G)} \\
&\leq \frac{C^H_{\max}(\emptyset) + g}{C^H_{\max}(\emptyset)} \\
&= 1 + \frac{g}{C^H_{\max}(\emptyset)} \\
&\leq 2 \text{ (by hypothesis).}
\end{aligned}$$

To prove that the bound is tight, consider the following instance. There are 2 jobs with processing times as follows;

Job i	$p_{i,1}$	$p_{i,2}$
1	1	1
2	2	8

$s_1 = 2$, $g = 3$

If the jobs are processed according to *JR* they shall be processed in the order 1,2 and the

makespan is $C^H_{\max}(G) = 14$. However, the optimal sequence is 2,1 and the makespan is $C^*_{\max}(G) = 11$. In the absence of the gap the JR sequence is optimal with $C^H_{\max}(\emptyset) = 11$. Then

$$r(H) = \frac{14}{11}.$$

While the bound is

$$1 + \frac{g}{C_H(\emptyset)} = 1 + \frac{3}{11} = \frac{14}{11},$$

which demonstrates its tightness. ■

We now determine the bound in case the duration of the gap is larger than the optimal makespan in absence of a gap. Let

$$\Delta = g - C^H_{\max}(\emptyset) > 0. \tag{6.14}$$

Proposition 23 *For problem $F2|r - a(M1)|Cmax$, and under the assumption that $g > C^H_{\max}(\emptyset)$,*

$$\left(r(H) \mid g \geq C^H_{\max}(\emptyset)\right) = \frac{2C^H_{\max}(\emptyset) + \triangle}{C^H_{\max}(\emptyset) + \triangle + \varepsilon} < 2,$$

where $\varepsilon = P_1 + p_{\min,2}$, and P_1, $p_{\min,1}$ as defined in (6.6) and (6.7).

Proof. It is easy to see that

$$P_1 + g + p_{\min,2} \leq C^*_{\max}(G) \leq C^H_{\max}(\emptyset) + g.$$

Then we have that

$$\begin{aligned}\left(r(H) \mid g > C^H_{\max}(\emptyset)\right) &\leq \frac{2C^H_{\max}(\emptyset) + \triangle}{P_1 + g + p_{\min,2}} \\ &\leq \frac{2C^H_{\max}(\emptyset) + \triangle}{C^H_{\max}(\emptyset) + \triangle + \varepsilon} < 2.\end{aligned}$$

with the second inequality justified by (6.14).

To demonstrate the tightness of this bound, consider the following instance. There are two jobs with processing times as follows:

Job i	$p_{i,1}$	$p_{i,2}$
1	1	1
2	1.05	7

$s_1 = 1.05$, $g = 9.95$

It is easy to verify that JR may yield the sequence 1,2 for a makespan $C^H_{\max}(\emptyset) = 9.05$, whence $\triangle = 0.9$.We also have $P_1 = 2.05$; and $\varepsilon = 3.05$. The JR would sequence the jobs in the order 1,2 which result in $C^H_{\max}(G) = 19$, while the optimum is to sequence the jobs in order 2,1 to yield $C^*_{\max}(G) = 13.0$, whence $r(H) = \frac{19}{13}$. The bound is $\frac{C^H_{\max}(\emptyset)+\triangle}{C^H_{\max}(\emptyset)+\triangle+\varepsilon} = \frac{19}{13}$, which demonstrates its tightness. ■

We can conclude that in a Resumable case and under applicable cases (the duration of the gap is a small fraction of the optimal makespan in the absence of a gap), the bound given in this chapter can give tighter bounds. Hence in this case the bound can be denoted by the function $1 + \epsilon$ such as $0 \leq \epsilon << 1$.

6.3.5 Johnson's Rule As Heuristic $F2|nr - a(M1)|Cmax$

The following lemma asserts that the makespan under JR in the presence of a gap on the first is no larger than the makespan under JR in the absence of the gap plus the duration of the gap plus the difference between the start of gap and the completion time of the last job finished before the gap under JR.

Let δ denote this difference,

$$\delta = s_1 - C^H_{b,1}(G).$$

where b denotes the last job that completes on the first machine at or before time s_1.

Lemma 24

$$C^H_{\max}(G) \leq C^H_{\max}(\emptyset) + g + \delta. \tag{6.15}$$

Proof. Denote the completion time of job b on the first machine by $C^H_{b,1}(G)$. Denote the job immediately preceding it on the first machine by $b-1$ (with the understanding that it is null if b is the first job) and denote the job immediately following it by $b+1$. There are two cases:

1. **Case 1:**

$$C^H_{b,2}(G) \geq C^H_{b+1,1}(G),$$

then the presence of the gap will have no impact whatsoever on the makespan and $C_H(G) = C_H(\emptyset)$.

2. **Case 2:**

$$C^H_{b,2}(G) \leq C^H_{b+1,1}(G),$$

then the delay in the makespan is at most $g + \delta$. ■

The following proposition establishes bounds on the error committed when using JR under the assumption that the sum $g + \delta$ is no longer than the makespan in the absence of the gap.

Proposition 25 $\left(r(H) \mid g + \delta \leq C^H_{\max}(\emptyset)\right) = 1 + \frac{g+\delta}{C^H_{\max}(\emptyset)} \leq 2.$

Proof. Follows immediately from Lemma 24 and inequality (**??**),

$$\begin{aligned}
\left(r(H) \mid g + \delta \leq C^H_{\max}(\emptyset)\right) &\leq \frac{C^H_{\max}(\emptyset) + g + \delta}{C^*_{\max}(G)} \\
&\leq \frac{C^H_{\max}(\emptyset) + g + \delta}{C^H_{\max}(\emptyset)} \\
&= 1 + \frac{g + \delta}{C^H_{\max}(\emptyset)} \\
&\leq 2 \text{ (by hypothesis).}
\end{aligned}$$

To prove that the bound is tight, consider the following instance. There are 2 jobs with

processing times as follows;

Job i	$p_{i,1}$	$p_{i,2}$
1	1	1
2	2	8

$s_1 = 2$, $g = 3$

If the jobs are processed according to JR they may be processed in the order 1,2 and the makespan is $C^H_{\max}(G) = 15$. However, the optimal sequence is 2,1 and the makespan is $C^*_{\max}(G) = 11$. In the absence of the gap the JR sequence is optimal with $C^H_{\max}(\emptyset) = 11$, Then, $r(H) = \frac{15}{11}$. With $\delta = 1$, we have the bound equal to $1 + \frac{g+\delta}{C^H_{\max}(\emptyset)} = 1 + \frac{3+1}{11} = \frac{15}{11}$, which demonstrates its tightness. ■

We consider the two examples 26 and 27 given bellow. The first example illustrates the cases when the duration of the gap is more than the makespan in the absence of a gap ($g > C^H_{\max}(\emptyset)$). However the second example illustrates the case when the duration of the gap is a small fraction of the optimal makespan in the absence of a gap ($g << C^H_{\max}(\emptyset)$). We can conclude that also in Non-Resumable case and under applicable cases (the duration of the gap is a small fraction of the optimal makespan in the absence of a gap), the bound given in this chapter can give tighter bounds. The bound can be denoted by the function $1 + \epsilon$ such as $0 \leq \epsilon << 1$.

Example 26 *Consider the 6 jobs with the following data:*

Job i	***1***	***2***	***3***	***4***	***5***	***6***
$p_{i,1}$:	*2*	*4*	*5*	*8*	*9*	*11*
$p_{i,2}$:	*7*	*5*	*3*	*1*	*6*	*8*

The gap starts at $s_1 = 10$ *and terminates at* $t_1 = 51$, *whence* $g = 41$; *and* $P_1 = 39$. *The* JO *is 1,2,6,5,3,4 with makespan* $C^H_{\max}(\emptyset) = 40$. *Therefore*

$$r(H) = 1 + \frac{11}{40} = 1.275.$$

For this example,

$$C^{H}_{\max}(G) = 85,$$

and

$$C^{*}_{\max}(G) = 85,$$

whence JO *is optimal!*

Example 27 *Consider the 15 jobs with the following data:*

Job i	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
p_{1i} :	*6*	*4*	*5*	*4*	*6*	*5*	*6*	*6*	*2*	*5*	*4*	*6*	*5*	*4*	*4*
p_{2i} :	*4*	*7*	*1*	*3*	*4*	*3*	*5*	*6*	*7*	*6*	*2*	*7*	*6*	*3*	*2*

The gap starts at $s_1 = 25$ *and terminates at* $t_1 = 28$; *whence* $g = 3$. *The* JO *is 9,2,10,13,12,8,7,5,1,6,4,14,11,15,3 with makespan* $C^{H}_{\max}(\emptyset) = 73$. *We have,*

$$r(H) = 1 + \frac{25}{73} = 1.342,$$

Therefore we can assert that the JO *is within* 34.2% *from the optimum. After resolving the problem with dynamic programming (or the combinatorial approach), the optimal solution is the sequence 2,1,5,4,6,10,13,128,7,14,11,15,3 with* $C^{*}_{\max} = 76$. *The makespan given by* JO *is* $C^{H}_{\max}(G) = 79$. *Then,*

$$r(H) = \frac{79}{76} = 1.039,$$

which is well within the bound of 34.2% *of the optimum.*

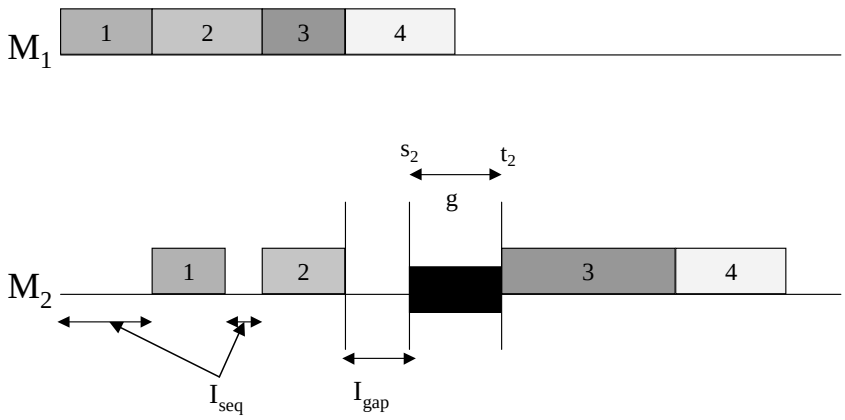

Figure 6-6: Idle times due to the sequence and the gap.

6.4 The Gap on The Second Machine

6.4.1 Optimality Conditions For The Johnson Order (F2|r,nr-a(M2)|C_{max})

We first illustrate in the Figure6-6 the two kinds of idle time and after we give some optimality conditions of Johnson's rule if the gap is on the second machine.

Proposition 28 *If the idle time due to the gap in Johnson's order is equal to zero then it is optimal.*

Proof. *The makespan under any sequence σ is given by*

$$C_{\max}(\sigma) = \sum_i p_{i,2} + \sum I_{\text{seq}} + I_{gap} + g \tag{6.16}$$

in which $\sum I_{\text{seq}}$ *is the sum of idle time on m/c2 due to the sequence of jobs, and* I_{gap} *the idle time due to the gap. Therefore in order to minimize the makespan one should find the sequence that minimizes the two of kinds of idle time. Now if the sum* $I_{gap} = 0$ *and since* JO *minimizes the sum of idle time* $\sum I_{\text{seq}}$ *then it is the optimal sequence.* ■

Example 29 *This example (see Figure6-7) illustrates the case when there exists a job b started on the first machine before the gap and finished after the gap (i.e* $C_{b,1} \geq t_2$*) and* $C_{b,1} = s_{b,2}$*. In this case the idle time due to the gap denoted by* I_{gap} *is equal to zero hence the Johnson's Order is the optimal solution.*

Example 30 *The example of the Figure6-8 focuses on the case when there exists a job b such that* $C_{b,1} = s_2$ *and* $s_{b,2} = t_2$*. In this case also* $I_{gap} = 0$ *then Johnson's Order is the optimal solution.*

Example 31 *When there exists a job b such that* $C_{b,2} = s_2$ *and* $s_{b+1,2} = t_2$ *(b+1 indicates here the job sequenced after the job b), Johnson's Order is the optimal solution. This case is illustrated by the Figure6-9.*

6.4.2 A Combinatorial Approach (F2|r,nr-a(M2)|Cmax)

The lower bound on the makepsan if the gap is on the second machine (*l.b.*) is secured as

$$l.b. = \max\left\{ P_1 + \min_i \{p_{2,i}\} \ , \ P_2 + \min_i \{p_{1,i}\} + g, \right\}. \tag{6.17}$$

And the upper bound (*u.b.*) may be secured by applying any heuristic (such as *JR*). Let T denote the Hypothesized makespan; then

$$l.b. \leq T \leq u.b.$$

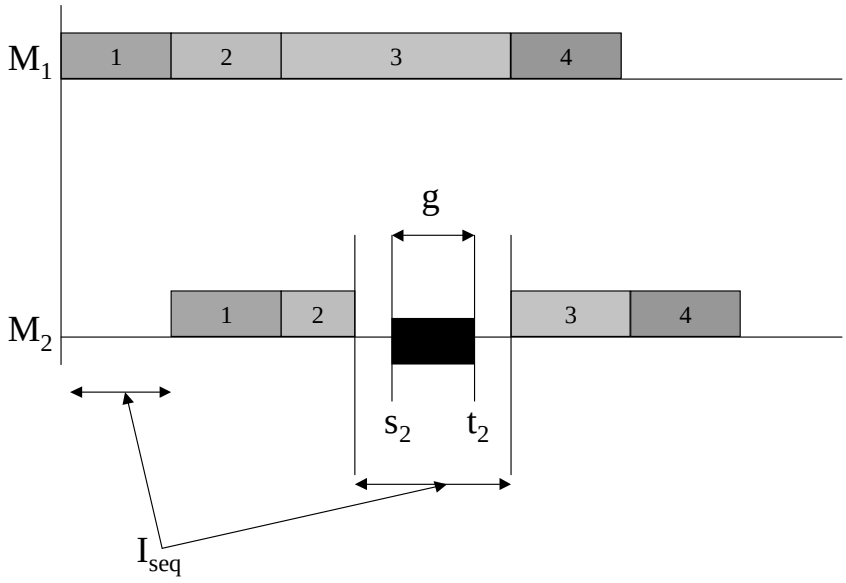

Figure 6-7: The first condition.

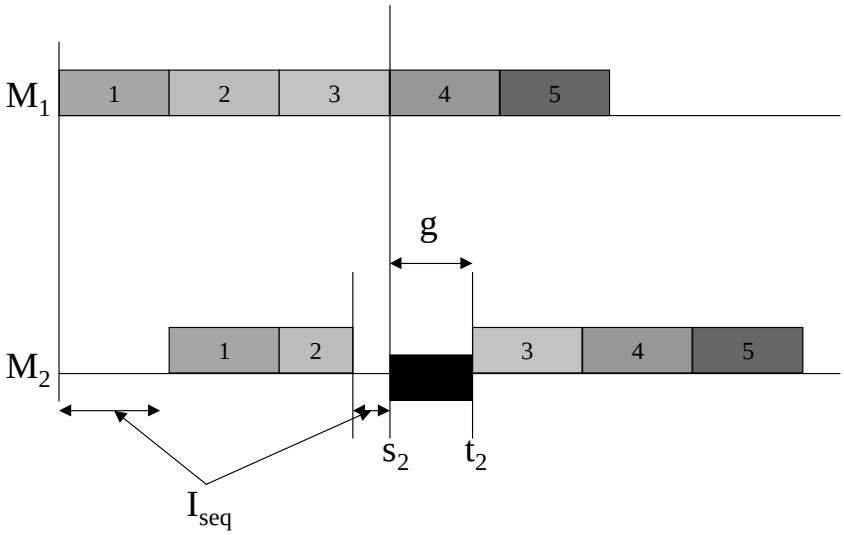

Figure 6-8: The second condition.

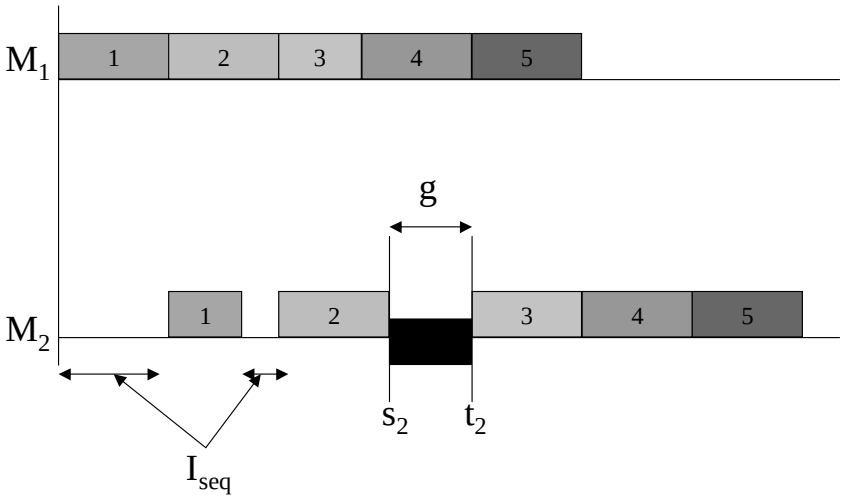

Figure 6-9: The third condition.

In this approach one may proceed by enumerating over the feasible values of T. Better still, one may use dichotomous search between the lower bound ($l.b$) and the upper bound ($u.b$) to pinpoint the smallest T for which a feasible schedule can be obtained by solving at each time the following knapsack problem. This determines the optimal makespan and the optimal sequence.

At the iteration r of the dichotomous search, the knapsack problem can be written as:

$$\begin{aligned} & \max \quad z = \sum_{i \in A} x_i \\ \text{s.t.} \quad & \sum_{i \in A} p_{i,2} \times x_i \leq T_r - t_2 \end{aligned}$$

T_r is the makespan at the iteration r in the dichotomous search.

Example 32 *Consider 5 jobs with the following data:*

Job i	***1***	***2***	***3***	***4***	***5***
$p_{i,1}$:	*1*	*3*	*9*	*10*	*5*
$p_{i,2}$:	*6*	*8*	*7*	*4*	*2*

$s_2 = 13$ *and* $g = 3$.

The lower bound on makespan l.b. $= \max\{28+2, 27+1\} = 30$. *The upper bound is derived from the schedule that results from using* JR *: u.b.* $= 37$. *Suppose we use dichotomous search,*

Finish at 33 *Sum* $P_2(A) \leq 33 - 16 = 17$

Solving the knapsack problem:

$$\begin{aligned} & \max \quad z = \sum_{i} x_i \\ \textit{s.t.} \quad & 6x_1 + 8x_2 + 7x_3 + 4x_4 + 2x_5 \leq 17 \end{aligned}$$

yields the result $z^* = 3$ *with the alternate optima:*

$$\begin{aligned} X^{1*} &= \{x_1^* = 1 = x_2^* = x_5^*\} \\ X^{2*} &= \{x_1^* = 1 = x_3^* = x_5^*\} \\ X^{3*} &= \{x_1^* = 1 = x_3^* = x_4^*\} \\ X^{4*} &= \{x_1^* = 1 = x_4^* = x_5^*\} \\ X^{5*} &= \{x_2^* = 1 = x_3^* = x_5^*\} \\ X^{6*} &= \{x_2^* = 1 = x_4^* = x_5^*\} \\ X^{7*} &= \{x_3^* = 1 = x_4^* = x_5^*\}. \end{aligned}$$

It is easy to verify that X^{1*}, X^{2*} *are infeasible because their corresponding completion time on m/c2 is* $> s_2$. X^{3*} *is feasible:*

$A = \{1,3,4\}:$ *feasible;* $B = \{2,5\}:$ $p_{1,2} + p_{2,2} + p_{2,5} = 13 = s_2$

The corresponding Gantt chart may be constructed from the following table:

Job	***start on m/c1***	***end on m/c1***	***start on m/c2***	***end on m/c2***
2	*0*	*3*	*3*	*11*
5	*3*	*8*	*11*	*13*
1	*8*	*9*	*16*	*22*
3	*9*	*18*	*22*	*29*
4	*18*	*24*	*29*	*33*

We forfeit analyzing the other solutions because we know that the new u.b. *is 33. The search space narrows down to the interval* $[30, 32]$. *Is 32 feasible? We solve the knapsack problem:*

$$\begin{aligned} & \max \quad z = \sum_i x_i \\ s.t. \quad & 6x_1 + 8x_2 + 7x_3 + 4x_4 + 2x_5 \leq 16 \end{aligned}$$

which has $z^* = 3$ *and the alternative optima:*

$$\begin{aligned} X^{1*} &= \{x_1^* = 1 = x_2^* = x_5^*\} \\ X^{2*} &= \{x_1^* = 1 = x_3^* = x_5^*\} \\ X^{3*} &= \{x_1^* = 1 = x_4^* = x_5^*\} \\ X^{4*} &= \{x_2^* = 1 = x_4^* = x_5^*\} \\ X^{5*} &= \{x_3^* = 1 = x_4^* = x_5^*\} \end{aligned}$$

We have already established that X^{1*} *and* X^{2*} *are infeasible. It is easy to verify that* X^{3*}, X^{4*} *and* X^{5*} *are also infeasible. Therefore the optimal sequence is 2,5,1,3,4 with makespan* $C^*_{\max} = 33$.

The time complexity of this procedure is $O\left(n \times (ub - lb) \times (ub - t_2)\right)$. If we compare the time of this procedure with the time of Lee's model ($O(n \times s_2 \times (\sum(p_{i,1} + p_{i,2})^2) \times p_{\max,2})$) [10] applying to this example we can conclude that our procedure's complexity is 10^3 less than the complexity of Lee's model.

6.4.3 Johnson's Rule As Heuristic F2|r-a(M2)|Cmax

It should be evident that if the gap occurs earlier than the minimal processing time on the first machine; *i.e.,* if

$$s_2 \leq \min_i \{p_{i,1}\}$$

then JR is optimal. Henceforth we shall assume that

$$s_2 > \min_i \{p_{i,1}\}. \tag{6.18}$$

The following lemma asserts that the makespan under JR in the presence of a gap on the second machine is no larger than the makespan under JR in the absence of the gap plus the duration of the gap plus the maximum of the processing times on the first

machine.

Lemma 33

$$C_{\max}^{H}(G) \leq C_{\max}^{H}(\emptyset) + g. \tag{6.19}$$

Proof. Denote the last job that completes on the first machine at or before time s_2 by i_b, and we write its completion times as $C_b(\cdot)$ for simplicity. Denote the job immediately preceding it on the first machine shall be denoted by i_{b-1} (with the understanding that it is the null job if $b = 1$), and denote the job immediately following it by i_{b+1}. The latter either straddles the gap (it completes on the first machine at or after time $s_2 + g$; *i.e.*,$C_{b+1,1}^{H} \geq s_2 + g$), or it completes processing on the first machine before the end of gap on the second machine; $s_2 < C_{b+1,2}^{H} < s_2 + g$. There are three cases:

1. **Case 1:**

$$\left(\max\left\{C_{b,1}^{H}, C_{b-1,2}^{H}\right\} + p_{b,2} \leq s_2\right) \cap \left(C_{b+1,1}^{H} \geq s_2 + g\right)$$

then the presence of the gap will have no impact whatsoever on the makespan and $C_{\max}^{H}(G) = C_{\max}^{H}(\emptyset)$.

2. **Case 2:**

$$\left(\max\left\{C_{b,1}^{H}, C_{b-1,2}^{H}\right\} + p_{b,2} \leq s_2\right) \cap \left(s_2 < C_{b+1,1}^{H} \leq s_2 + g\right)$$

then job $i_b + 1$ will start processing on the second machine immediately following the gap, and the delay in the makespan is at most g.

3. **Case 3:**

$$\max\left\{C_{b,1}^{H}, C_{b-1,2}^{H}\right\} + p_{b,2} > s_2,$$

then job b will start processing on the second machine immediately following the gap, and the delay in the makespan is at most g.

Hence $C_{\max}^{H}(G)$ would be delayed by at most g as was to be shown. ■

The following inequality is obvious

$$C_{\max}^{*}(G) \geq C_{\max}^{H}(\emptyset). \tag{6.20}$$

The following propositions establishes bounds on the error committed when using JR under the assumption that the gap is of duration no longer than the half of makespan in the absence of the gap.

Proposition 34 $\left(r\left(H\right) \mid g \leq \frac{C_{\max}^{H}(\emptyset)}{2}\right) = 1 + \frac{g}{C_{\max}^{H}(\emptyset)} \leq \frac{3}{2}$.

Proof. Follows immediately from Lemma 33 and inequality (6.20),

$$\begin{aligned}\left(r\left(H\right) \mid g \leq \frac{C_{\max}^{H}\left(\emptyset\right)}{2}\right) &\leq \frac{C_{\max}^{H}\left(\emptyset\right)+g}{C_{\max}^{H}\left(\emptyset\right)} \\ &= 1+\frac{g}{C_{\max}^{H}\left(\emptyset\right)} \\ &\leq \frac{3}{2}\end{aligned}$$

■

The next proposition establishes a bound when the gap occurrs 'late', in particular when s_2 is no less than the larger of one half the makespan in the absence of a gap and the duration of the gap.

Proposition 35 *(a) If* $C_{\max}^{H}\left(\emptyset\right)/2 \leq s_2 \leq g$. *then* $r\left(H\right) = 1 + \frac{s_2}{C_{\max}^{H}(\emptyset)} \leq \frac{3}{2}$.
(b) If $s_2 \geq \max\left\{g,\, C_{\max}^{H}\left(\emptyset\right)/2\right\}$ *then* $r\left(H\right) = \frac{C_{\max}^{H}(\emptyset)+g}{t_2} \leq \frac{3}{2}$.

Proof.

(a) By Lee's lemma [9]

$$C_{\max}^{*}\left(G\right) + s_2 \geq C_{\max}^{H}\left(G\right)$$

We have that

$$\begin{aligned}\left(r\left(H\right) \,/\, C_{\max}^{H}\left(\emptyset\right)/2 \leq s_2 \leq g\right) &\leq \frac{C_{\max}^{*}\left(G\right)+s_2}{C_{\max}^{*}\left(G\right)} \\ &= 1+\frac{s_2}{C_{\max}^{H}\left(\emptyset\right)} \leq \frac{3}{2}\end{aligned}$$

(b)By hypothesis,

$$t_2 = s_2 + g \geq 2g.$$

It is easy to see that

$$\begin{aligned} \left(r\left(H\right)/s_2 \geq \max\left\{g,\, C^H_{\max}\left(\emptyset\right)\right\}\right) &\leq \frac{C^H_{\max}\left(\emptyset\right) + g}{t_2} \\ &\leq \frac{3}{2} \end{aligned}$$

■

6.5 Conclusion

We have studied the "resumable" and the "non-resumable" modes of the two machine flow shop problem with an availability constraint period (a gap) fixed either on the first machine or on the second machine, with the view of establishing improved optimization procedures and evaluating the performance of Johnson's Rule. We have focused on Johnson's Rule because of its simplicity and its common knowledge. We have specified the conditions under which Johnson's Rule gives the optimum if the gap is either on the first machine or on the second machine. We have proposed a dynamic program if the gap is on the first machine, and a combinatorial approach if the gap is on the second machine which are for some instances of the problem more efficient than currently available models, albeit still computationally demanding if the number of jobs is large and the gap occurs near the middle of the optimal sequence in the absence of the gap. We have demonstrated that under "normal" conditions (the duration of the gap is a small fraction of the optimal makespan in the absence of a gap) Johnson's Rule can give tighter bounds which are more better than the value of 2. One of our future research shall be directed towards the study of more general configurations of machine unavailability, such as more than one gap on either machine or gaps on both machines.

Bibliography

[1] **Allaoui, H.**, Artiba, A., Riane, F and Elmaghraby, S.E., (2004). "Scheduling two machine flow shop with an availability constraint on the first machine". Accepted to appear in *Inter J of Prod Eco.*

[2] **Allaoui, H.**, Artiba, A., Riane, F and Elmaghraby, S.E., (2003). "On scheduling of two machine flow shop with an availability constraint on the second machine". Proceedings of MISTA conference, vol1, 279-290. Nottingham, UK.

[3] Bellman, R.(1982)."Mathematical Aspects of Scheduling and Applications", Chapter 7. *Pergamon Press.*

[4] Blazewicz, J., Breit, J., Formanowicz, P., Kubiak, W., and Schmidt, G. (2001). "Heuristic algorithms for two-machine flowshop with limited machine availability", *Omega 29,* 599-608.

[5] Breit, J., Schmidt, G., and Strusevich, V.A. (2001). "Two-machine openshop with an availability constraint", *Oper.Res.Lett. 29* 65-77.

[6] Cheng, T.C.E. and Wang, G., (2000). "An improved heuristic for two-machine flowshop scheduling with an availability constriant", *Oper.Res.Lett. 26,* 223-229.

[7] Johnson, S.M. (1954). "Optimal two- and three-stage productionschedules with setup times included," *Nav. Res. Log. Quart. 1,* 61-67.

[8] Kubiak, W., Blazewicz,.J., .Formanowicz, P., and Schmidt, G. (2002). "Two-machine flowshop with limited machine availability", *Euro.J.Oper.Res. 136*, 528-540.

[9] Lee, C.-Y.,.(1997). "Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint", *Oper.Res.Lett. 20*, 129-139.

[10] Lee, C.-Y., (1999). "Two-machine flowshop scheduling with an availability constraint", *Euro.J.Oper.Res. 114,* 420-429.

[11] Lee, C.-Y., Lei, L., and Pinedo, M. (1997). "Curent trends in deterministic scheduling", *Ann.Oper.Res. 70,* 1-41.

[12] Lee, C.-Y. (1996). "Machine scheduling with an availability constraint", *J.Global Optimization.9,* 363-382.

[13] Pinedo, M. (2002). "Scheduling : Theory, Algorithms and Systems". 2nd ed., Prentice-Hall.

[14] Schmidt, G. (2000). "Scheduling with limited machine availability", *Euro.J.Oper.Res.121,* 1-15.

Chapter 7

SCHEDULING TWO STAGE HYBRID FLOW SHOP WITH AVAILABILITY CONSTRAINTS

7.1 Introduction

In this chapter we investigate the two-stage hybrid flow shop scheduling problem (see Figure7-1) with only one machine on the first stage and m identical machines on the second stage to minimize the makespan. At any time, every job can be processed by at most one machine and every machine can process at most one job. Jobs can wait between the two stages in unlimited storage. We assume that each machine is subject to at most one unavailability period. The start time and the end time of each period are known in advance (deterministic case) and only the *non-resumable* case is studied. This problem will be denoted as $F2(P)|(m_1 = 1, m_2 = m), nr - a|C_{\max}$.

The chapter is organized as follows: Section 7.3 discusses the complexity of the problem. The purpose of section 7.5 is to formulate a branch and bound algorithm to minimize the makespan in nonresumable case. This problem is reduced to two steps of taking decisions, the first is how to sequence jobs on the first stage and the second is how

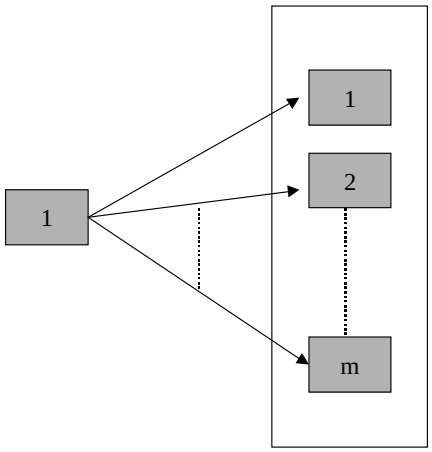

Figure 7-1: The two stage hybrid flow shop.

to assign jobs to machines on the second stage. Furthermore even if jobs are assigned to machines on the second stage, jobs should be started as early as possible in the optimal solution. Finally in the section 7.6 we calculate the error bounds of the LIST algorithm, LPT algorithm and H heuristic proposed by Lee [15] for the hybrid flow shop problem without availability constraints.

7.2 Notations

We adopt the following notations.

J_i : Job i, $i = 1, ..., n$.

M_j : Machine j on the second stage, $j = 1, ..., m$.

$p_{i,1}$: Processing time of J_i on the first stage.

$p_{i,2}$: Processing time of J_i on the second stage.

$MS1$: The sum of processing time on the first stage $(\sum_{i=1}^{n} p_{i,1})$.

$MS2$: The sum of processing time on the second stage $(\sum_{i=1}^{n} p_{i,2})$.

$C_{i,1}$: The completion time of J_i on the first stage.

$C_{i,2}$: The completion time of J_i on the second stage.

$C_{\max}$: The makespan.

s_1 : The start time of the gap on machine in the first stage.

t_1 : The finish time of the gap on machine in the first stage.

$s_{j,2}$: The start time of the gap on M_j in the second stage.

$t_{j,2}$: The finish time of the gap on M_j in the second stage.

7.3 Complexity Analysis

The only variant of the flow shop problem solved in polynomial time is the two machine flow shop $F2 \mid . \mid C_{\max}$ (see Johnson[10]). The problem $F3 \mid . \mid C_{\max}$ is NP-hard in the strong sense (see Gary[6]). The problem $F2(P) \mid . \mid C_{\max}$ is NP-hard in the strong sense

even if there is only one machine on the first stage and two machines on the second stage (see Hoogveen[9]).

Thus this proposition stands obviously.

Proposition 36 *The problem* $F2(P)|(m_1 = 1, m_2 = m), nr - a|C_{\max}$ *is NP-hard in the strong sense.*

Lee(see Lee[12]) studies the problem $Pm|\ |C_{\max}$ in which each machine has at most one gap and shows that if there is no machine which is always available then $\frac{C^{LPT}_{\max}}{C^{*}_{\max}}$ can be arbitrarily large even for the two machine problem. This implies that no polynomial approximation scheme exists unless $P = NP$. We can show this result for the two stage hybrid flow shop problem even if we consider the same assumption for the second stage.

Proposition 37 *No polynomial approximation scheme exists unless* $P = NP$ *for the problem* $F2(P)|(m_1 = 1, m_2 = m), nr - a|C_{\max}$ *even if there is no machine on the second stage which is always available.*

Proof. Consider a problem with the following instance:

Job	1	2	3
$p_{i,1}$	1	2	3
$p_{i,2}$	3	2	1

$m = 2$, $s_1 = t_1 = 0$, $s_{12} = s_{22} = 7$, $t_{1,2} = t_{2,2} = n$.

The sequence given by LPT is (3,2,1) with $C^{LPT}_{\max} = n + 3$. However the optimal sequence is (1,2,3) or (2,1,3) with $C^{*}_{\max} = 7$. Hence $\frac{C^{LPT}_{\max}}{C^{*}_{\max}} = \frac{n+3}{7}$ which can be arbitrarily large as n approaches infinity. ■

Therefore we assume in the following that there is at least one machine at the second stage which is always available.

7.4 Particularity of This Problem

In the classical hybrid flow shop without availability constraint, if machines on each stage are identical then we can reduce some enumerations. Let us illustrate this property by a

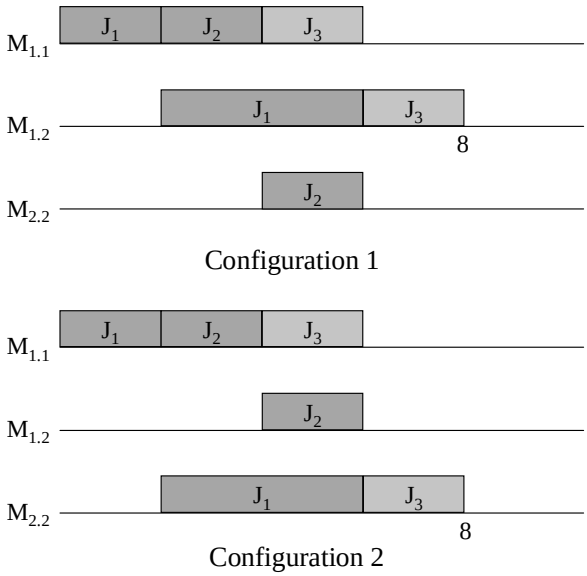

Figure 7-2: Without availability constraints.

simple example. We consider a two stage hybrid flow shop with one machine on the first stage and two machines on the second stage with tree jobs:

	$Stage1$	$Stage2$
J_1	2	4
J_2	2	2
J_3	2	2

We can see in the figure bellow (see Figure 7-2) that the two configurations (i.e. J_1 and J_3 on the first machine and J_3 on the second machine on the second stage or the inverse) are similar because the makespan stills unchanged which is equal to 8:

Now we will see that it is not generally the case if we take on consideration the unavailability constraint. Let us illustrate this particularity by the previous instance but

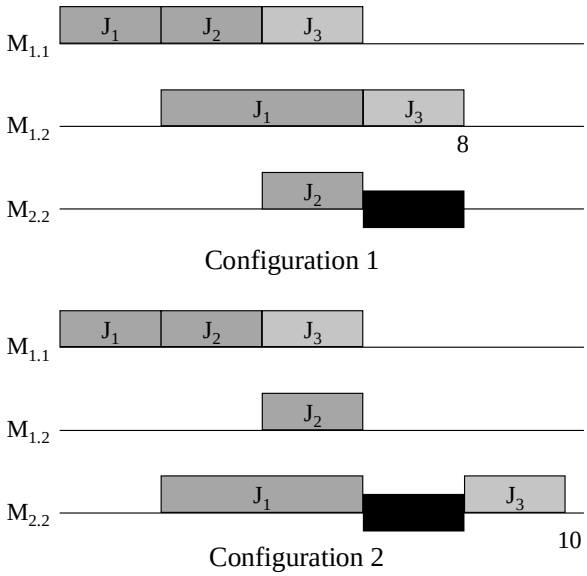

Figure 7-3: With availability constraints.

this time we consider that the second machine on the second stage is subject to one gap (see Figure 7-3) such that $s_{2,2} = 6$ and $t_{2,2} = 8$.

The two makespans on the two configurations are different, then we should enumerate the two possibilities. In this case machines are identical on term of the speed and the capacity but not on term of the availability. Thus the total number of all feasible schedules will be $n!m^n$ such that n is the number of jobs and m is the number of machines on the second stage.

7.5 Branch and Bound Algorithm

To solve the $k - stage$ hybrid flow shop scheduling problems, the only exact method used is a branch and bound of Brah and Hunsucker (see Brah[4]). In this chapter we use their concept of this approach to resolve the two stage hybrid flow shop with availability constraints such that there is only one machine on the first stage and m machines on the second stage. The branch and bound algorithm consists of three steps: the calculation of bounds, the branching procedure and node elimination.

7.5.1 Determination of lower bounds

We will use the same Brah and Hunssucker's approach to calculate a lower bound for the branch and bound. Thus our contribution is to update their bounds by integrating the availability constraints to calculations in order to have tight bounds.

Completion times

Before starting we introduce this notation needed to formulate the equations:

N :	A subset of all jobs
S :	A subset of some jobs such that $S \subset N$
S' :	A subset of all jobs on S and an other job J_q such that $J_q \notin S$ $(S' = S \cup \{J_q\})$
$\sigma_1(S)$:	The partial schedule of the subset jobs S on the first stage
$\sigma_2(S)$:	The partial schedule of the subset jobs S on the second stage
$C_0(\sigma_1(S))$:	The completion time of the partial schedule $\sigma_1(S)$ on the first stage
$C_j(\sigma_2(S))$:	The completion time of the partial schedule $\sigma_2(S)$ on machine M_j $(j = 1, ..., m)$ on the second stage

Thus the partial schedule $\sigma_k(S')$ $(k = 1, 2)$ is obtained by adding the job J_q to the partial schedule $\sigma_k(S)$. The makespan can be expressed as:

$$C_{\max} = \max_{1 \leq j \leq m} C_j(\sigma_2(N)) \tag{7.1}$$

Now we give the equations involving the completion times on the first stage and on every machine on the second stage:

$$C_0(\sigma_1(S')) = \left\{ \begin{array}{ll} t_1 + p_{q,1} & \text{if} \quad (C_0(\sigma_1(S)) \leq s_1 < C_0(\sigma_1(S)) + p_{q,1}) \\ C_0(\sigma_1(S) + p_{q,1} & \text{otherwise} \end{array} \right\} \tag{7.2}$$

$$C_j(\sigma_2(S')) = \left\{ \begin{array}{l} C_j(\sigma_2(S)) \text{ if } J_q \text{ is not assigned to } M_j \\ \left\{ \begin{array}{ll} t_{j2} + p_{q,2} & \text{if} \quad (Max(C_j(\sigma_2(S)) \ , \ C_0(\sigma_1(S'))) \leq s_{j,2} \\ < Max(C_j(\sigma_2(S)) \ , \ C_0(\sigma_1(S'))) + p_{q,2} & \\ Max(C_j(\sigma_2(S)) \ , \ C_0(\sigma_1(S'))) + p_{q,2} & \text{otherwise} \end{array} \right\} \text{ otherwise} \end{array} \right\} \tag{7.3}$$

Machine based bounds

We use the unprocessed work load at the any of the two stages to give a lower bound for minimizing the makespan at that stage. For any given partial schedule $\sigma_1(S')$ we denote the maximum completion time for this partial schedule on the first stage by:

$$MCT(\sigma_1(S')) = C_0(\sigma_1(S')) \tag{7.4}$$

And the average completion time and processing requirement for this partial schedule on the first stage by:

$$ACT(\sigma_1(S')) = C_0(\sigma_1(S')) + \sum_{i \in N - S'} p_{i,1} \tag{7.5}$$

For the second stage, the two amounts will be expressed as:

$$MCT(\sigma_2(S')) = \max_{1 \leq j \leq m} (C_j(\sigma_2(S'))) \tag{7.6}$$

$$ACT(\sigma_2(S')) = \frac{\sum_{j=1}^{m} C_j(\sigma_2(S'))}{m} + \frac{\sum_{i \in N - S'} p_{i,2}}{m} \tag{7.7}$$

For the problem studied in this chapter, we have only one machine on the first stage, then the following relation is always verified as we can conclude from the two equations 7.4 and 7.5:

$$MCT(\sigma_1(S')) \leq ACT(\sigma_1(S')) \tag{7.8}$$

Then the machine based lower bound for the branching node for the first stage can be given as:

$$LBM[\sigma_1(S')] = \left\{ \begin{array}{l} ACT(\sigma_1(S')) + \underset{i \in N-S'}{Min} \{p_{i,2}\} \quad \text{if} \\ (C_0(\sigma_1(S')) > t_1) \vee ((C_0(\sigma_1(S')) + \sum_{i \in N-S'} p_{i,1}) \leq s_1) \\ ACT(\sigma_1(S')) + (t_1 - s_1) + \varepsilon_{\min} + \underset{i \in N-S'}{Min} \{p_{i,2}\} \quad \text{otherwise} \end{array} \right\} \tag{7.9}$$

The amount $\varepsilon_{\min}$ is given at each node by resolving the *knapsack* problem:

$$Max \ (Y = \sum_{i \in N-S'} x_i \times p_{i,1})$$

$$\text{Such that : } C_0(\sigma_1(S')) + Y \leq s_1$$

$$x_i = \left\{ \begin{array}{c} 1 \ \text{ if } \ J_i \text{ is assigned before the gap} \\ 0 \quad \text{otherwise} \end{array} \right\}$$

Then

$$\varepsilon_{\min} = s_1 - Y^*$$

The complexity of the calculations at each node can be at most $O(card(N - S') \times (s_1 - C_0(\sigma_1(S'))))$. Thus it is a compromise to find between the time of calculations and the quality of bounds.

For the second stage the lower bound for the branching node is:

$$LBM[\sigma_2(S')] = Max(MCT(\sigma_2(S')), ACT(\sigma_2(S'))) \tag{7.10}$$

The effect of the availability constraint is introduced in the calculations of $C_j(\sigma_2(S'))$.

Job based bounds

The calculations for a job based bound focuses on the remaining processing required of each unscheduled job at each stage k ($k = 1, 2$). The job based bound for a hybrid flow shop can not be strong since there are alternate routes for the other jobs on the set which is not the case in the classical flow shop. It is easy to see that the job based lower bound on the first stage is given by:

$$LBJ[\sigma_1(S')] = C_0(\sigma_1(S')) + \underset{i \in N-S'}{Max} \{p_{i,1} + p_{i,2}\} \tag{7.11}$$

And the job lower bound on the second stage is given by:

$$LBJ[\sigma_2(S')] = \underset{j}{Min}(C_j(\sigma_2(S'))) + \underset{i \in N-S'}{Max} \{p_{i,2}\} \tag{7.12}$$

Thus the composite lower bound for the two stage hybrid flow shop with availability constraints for the branching node at stage $k(k = 1, 2)$ is as follows:

$$LBC[\sigma_k(S')] = Max \{LBM[\sigma_k(S')], LBJ[\sigma_k(S')]\} \tag{7.13}$$

7.5.2 The branching strategy

Since we take on consideration the availability constraints on machines, we propose a new branching strategy for the two stage hybrid flow shop problem which is different of the branching approach proposed by Brah and Hunssucker. Thus on the first stage the decision activity is the sequence of jobs and on the second stage is the assignment of jobs to a specific machine j among m parallel machines. The branching procedure will be presented by a tree. Two types of nodes will be used:

- The first following node (see Figure 7-4) means that the job J_i is sequenced on the

Figure 7-4: The square node.

Figure 7-5: The circle node.

position r on the first stage.

- The second following node (see Figure 7-5) means that the job J_i is affected to machine M_j on the second stage.

In order to develop the algorithm of this branching strategy we give the necessary rules to follow:

1. The first level after the dummy node concerns a sequencing decision.

2. On the branching tree we use two types of levels. The first one is the level of sequencing decision and the second one is the level of assignment decision. Thus we use the notation L_{ed} to distinguish each level. If the nodes concern the first stage (i.e sequencing decision) $d = 1$. However if the nodes concern the second stage (i.e assignment decision) $d = 2$. The number of the level in the branching tree if $d = 1$ or $d = 2$ is indicated by e such that $1 \leq e \leq n$.

3. At every time a sequencing decision is followed by an assignment decision, that means that we can not find a path in the branching tree with two successive nodes of the same kind of decision (i.e sequencing or assignment).

4. The number of nodes at the level L_{e1} is at most equal to $n - e + 1$, and at the level L_{e2} is at most equal to m.

5. No path can be considered as a feasible solution if the number of square nodes or circle nodes is less than the number of jobs n.

Adding to the decision taken, each node is represented by these three elements:

- The partial schedule $\sigma_1(S')$ for the square nodes and $\sigma_2(S')$ for the circle nodes such that S' is the subset of jobs.
- The adding job q after each decision. It is the left number on each node.
- The composite lower bound $LBC[\sigma_k(S')]$ $(k = 1, 2)$.

Example 38 *To illustrate this example we consider this example of two stage hybrid flow shop with one machine on the first stage and two machines on the second stage. Three jobs have to be scheduled to minimize the makepsan.*

Job i	***1***	***2***	***3***
$p_{i,1}$:	*1*	*2*	*3*
$p_{i,2}$:	*10*	*4*	*5*

($s_1 = 2$ and $t_1 = 5$); ($s_{1,1} = 8$ and $t_{1,1} = 14$).

the Figure7-6 we present the search tree of the branch and bound algorithm described above applied to this instance, with the value of the composite bound for each node. According to the next section the makespan given by the heuristic H is equal to 18. Hence we can take the ub = 18. The branch and bound procedure gives two optimal solutions with a makespan equal to 18. In one of these two solutions jobs are sequenced on the first stage on the order 1,3,2. Job 2 is affected to the first machine on the second stage and jobs 1 and 3 are affected to the second machine.

7.6 Heuristics Worst Case Performance

7.6.1 List scheduling heuristic (LS)

Given an arbitrary order of jobs, assign the job to the machine such that the job can be finished as early as possible.

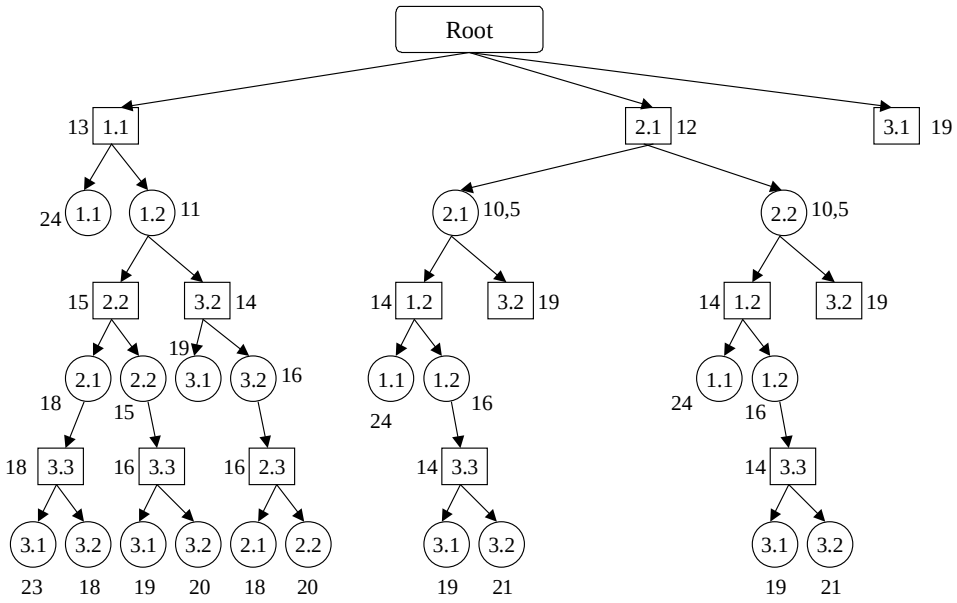

Figure 7-6: The branch and bound search tree.

Proposition 39 $\frac{C_{\max}^{LS}}{C_{\max}^{*}} \leq 2+m$, *where* m *is the number of machines on the second stage.*

Proof. Let us apply separately the list algorithm to n jobs in the first stage according to $\{p_{i,1}\}$ and in the second stage according to $\{p_{i,1}\}$. We denote the makespan on the first stage by $C_{\max}^{1}$ and the makespan on the second stage by $C_{\max}^{2}$. $C_{\max}$ will denote the makespan resulted by applying the list algorithm to our problem. It should be clear that:

$$C_{\max} \leq C_{\max}^{1} + C_{\max}^{2}. \tag{7.14}$$

$$C_{\max}^{1} \leq t_1 + MS1. \tag{7.15}$$

Now we have two cases:

i. $C_{\max}^{1} \leq s_1$ in this case it is obvious that:

$$C_{\max}^{1} = MS1 \leq C_{\max}^{*}. \tag{7.16}$$

ii. $C_{\max}^{1} > s_1$ in this case we have:

$$C_{\max}^{*} \geq t_1. \tag{7.17}$$

$$C_{\max}^{*} \geq (t_1 - s_1) + MS1. \tag{7.18}$$

From Lee et al. (see Lee[12]), we have:

$$\frac{C_{\max}^{2}}{C_{\max}^{2*}} \leq m. \tag{7.19}$$

Where $C_{\max}^{2*}$ is the optimal finish time if we schedule jobs on the second stage.

We have also:

$$C_{\max}^{*} \geq C_{\max}^{2*} \tag{7.20}$$

Using inequalities 7.14 we have:

$$\frac{C_{\max}}{C^*_{\max}} \leq \frac{C^1_{\max} + C^2_{\max}}{C^*_{\max}}$$

Using 7.15, 7.16, 7.17 and 7.18 we have:

$$\frac{C^1_{\max}}{C^*_{\max}} \leq 2 \tag{7.21}$$

From 7.19, 7.20 and 7.21 we have:

$$\frac{C^{LS}_{\max}}{C^*_{\max}} = \frac{C_{\max}}{C^*_{\max}} \leq 2 + m$$

■

7.6.2 LPT heuristic

In this heuristic we sort the set of jobs according to the processing time on the first stage in non-increasing order, then we use list scheduling algorithm.

Proposition 40 $\frac{C^{LPT}_{\max}}{C^*_{\max}} \leq 2 + \frac{m}{2}$, *where m is the number of machines on the second stage.*

Proof. Like the previous proof we have here also two cases:

i. $C^1_{\max} \leq s_1$ in this case it is obvious that:

$$C^1_{\max} = MS1 \leq C^*_{\max}. \tag{7.22}$$

ii. $C^1_{\max} > s_1$, let us denote the set of jobs before the gap by B and the set of jobs after the gap by A. In this case we have:

$$C^1_{\max} = t_1 + \sum_{i \in A} p_{i,1} = s_1 + (t_1 - s_1) + MS1 - \sum_{i \in B} p_{i,1} \tag{7.23}$$

$$\leq s_1 - \sum_{i \in B} p_{i,1} + C^*_{\max} \quad \text{because } C^*_{\max} \geq (t_1 - s_1) + MS1. \tag{7.24}$$

In the first case we have:

$$\begin{aligned}\frac{C_{\max}}{C^*_{\max}} &\leq \frac{C^1_{\max}+C^2_{\max}}{C^*_{\max}} \\ &\leq 1+\frac{C^2_{\max}}{C^{2*}_{\max}}\end{aligned}$$

From Lee et al.(see Lee[12]) we have:

$$\frac{C^2_{\max}}{C^{2*}_{\max}} \leq \frac{m+1}{2}. \tag{7.25}$$

Then:

$$\begin{aligned}\frac{C_{\max}}{C^*_{\max}} &\leq 1+(\frac{m+1}{2}) \\ &\leq 2+\frac{m}{2}\end{aligned}$$

In the second case and from 7.23 and 7.25 we have:

$$\begin{aligned}\frac{C_{\max}}{C^*_{\max}} &\leq \frac{C^1_{\max}+C^2_{\max}}{C^*_{\max}} \\ &\leq \frac{s_1-\sum_{i\in B}p_{i,1}+C^*_{\max}}{C^*_{\max}}+\frac{C^2_{\max}}{C^{2*}_{\max}} \\ &\leq \frac{s_1-\sum_{i\in B}p_{i,1}}{C^*_{\max}}+1+\frac{m+1}{2}\end{aligned}$$

We consider that the first job after the gap, on the first stage in the sequence given by LPT is in position f , then it is clear that:

$$s_1 - \sum_{i \in B} p_{i,1} < p_{[f]1} \tag{7.26}$$

Then we have:

$$\frac{C_{\max}}{C^*_{\max}} \leq \frac{p_{[f],1}}{p_{[f],1} + p_{[f-1],1}} + 1 + \frac{m+1}{2}$$

According to LPT order we have:

$$p_{[f],1} \leq p_{[f-1],1} \tag{7.27}$$

From 7.26 and 7.27 we conclude that:

$$\begin{aligned} \frac{C_{\max}}{C^*_{\max}} &\leq \frac{1}{2} + 1 + (\frac{m+1}{2}) \\ &\leq 2 + \frac{m}{2} \end{aligned}$$

■

7.6.3 H heuristic

Lee et al. (see Lee[15]) develop a heuristics based on Johnson algorithm and LBM rule for the problem of a two stage hybrid flow shop with m_1 machines on the first stage and m_2 machines on the second stage (m_1, $m_2 \geq 2$). They show that $\frac{C^H_{\max}}{C^*_{\max}} \leq 2 - \frac{1}{m}$. We will use the same heuristic for our problem which is a particular case on term of the number of machines on the first stage ($m_1 = 1$) and more general case on term of availability constraints.

LBM rule

Step 1. Set $t_k := T$ for $k = 1, ..., m$.

Step 2. Let j be the last unscheduled job of S and m a machine with largest t_k.Schedule job j on machine k to finish at time t_k.

Step 3. Set $t_k := t_k - p_j$ and $S := S - \{j\}$. If $S \neq \varnothing$ then goto Step 2 else Stop.

Heuristic H adapted to our problem

Step 1. Apply JA on the first stage tasks with respect to the processing times $\{((p_{i,1}), (\frac{1}{m}p_{i,2})), i = 1, ..., n\}$. Let S be the resulting sequence.

Step 2. Apply the LBM rule on the second stage tasks of the sequence S.

Step 3. On each stage 2 machine k, reorder the tasks assigned to k (during step 3) according to nonincreasing completion times of the corresponding

stage1 tasks. Let S_k be the resulting order for $k = 1, ..., m$.

Step 4. On each stage 2 machine m,schedule the tasks in S_m in this order, as soon as possible.

The next lemma will help us to find the error bound of the heuristic H. For this lemma we introduce an auxiliary problem AP. This problem will be a classic two machine flow shop with only one gap on the first machine. The processing time on the first machine of J_i is $p_{i,1}$ and the processing time on the second machine of J_i is $\frac{1}{m}p_{i,1}$.

Example 41 *Consider 5 jobs to schedule on two stage hybrid flow shop* $(F2(P)|(m_1 = 1, m_2 = m), nr - a|C_{\max}$ *) with the following data:*

Job i	***1***	***2***	***3***	***4***	***5***
$p_{i,1}$:	*1*	*2*	*3*	*4*	*6*
$p_{i,2}$:	*10*	*4*	*5*	*10*	*15*

($s_1 = 2$ *and* $t_1 = 5$*); (*$s_{1,1} = 8$ *and* $t_{1,1} = 14$*).*

sequence given by JR *is (1,2,3,4,5). Hence jobs are sequenced in this order on the single machine of the first stage. Applying LBM rule to the second stage tasks, jobs 2 and 5 should be affected to the first machine on the second stage and jobs 1,3 and 4 to the second machine. The Gantt chart of this example is given in Figure7-7.*

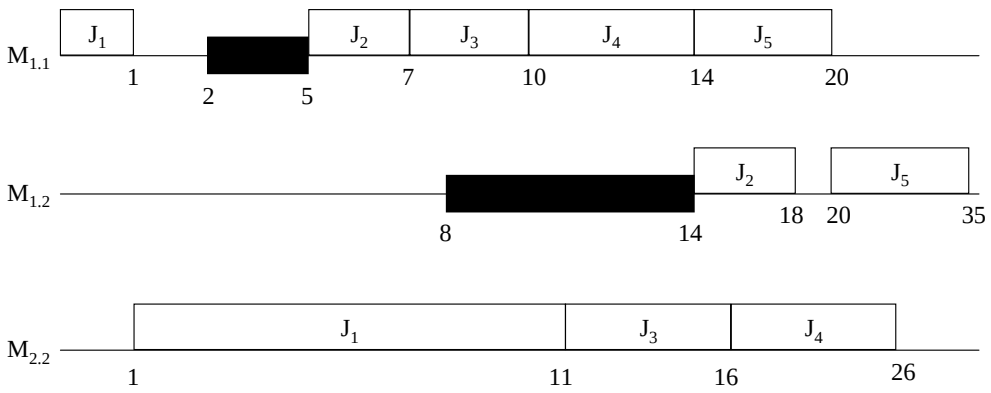

Figure 7-7: The Gantt chart of an application of H.

Lemma 42 *Let $C^{AP}_{\max}$ be the makespan resulted by applying JA to the AP. Then we have,* $C^{AP}_{\max} \leq 2C^{H}_{\max}$.

Proof. Let us denote S^* as the optimal solution of our problem and S the set of jobs ordered in nondecreasing order of completion times of the stage 1 tasks in S^*. And consider a partial schedule consisting of the first i stage 1 tasks of S and the last n-i+1 stage 2 tasks of S. Then we have:

$$C^{*}_{\max} \geq \sum_{j=1}^{i} p_{j,1} + \frac{1}{m}\sum_{j=i}^{n} p_{j,2} \text{ for every } 1 \leq i \leq n. \tag{7.28}$$

Let us apply the jobs according to the sequence S to the problem AP, then the makespan resulted C_S will be:

$$C_s = \sum_{j=1}^{i_0} p_{j,1} + \frac{1}{m}\sum_{j=i_0}^{n} p_{j,2} \tag{7.29}$$

Where J_{i_0} is the last job whose start on stage 2 exactly at its finish time on the first stage.

Then from 7.28 and 7.29 we have $C_s \leq C^{*}_{\max}$. According to Lee([11]), for the flow shop scheduling problem with only one gap on the first machine in semiresumable case (The error bound of JA is 2) we have:

$$\frac{C^{AP}_{\max}}{2} \leq C^{*} \leq C_s \leq C^{*}_{\max} \tag{7.30}$$

Where C^* is the optimal solution of the problem AP. Then from 7.30 we have $C^{AP}_{\max} \leq 2C^{*}_{\max}$. ■

Proposition 43 $\frac{C^{H}_{\max}}{C^{*}_{\max}} \leq 4 - \frac{1}{m}$, *where m is the number of machines on the second stage.*

Proof. Let us those notations:

$-A_1$: denotes the set of jobs whose are processed after the gap on the first stage.

$-B_1$: denotes the set of jobs whose are processed before the gap on the first stage.

$-A_2$: denotes the set of jobs whose are processed after any gap on the second stage.

$-B_2$: denotes the set of jobs whose are processed before any gap on the second stage.

The completion time of each J_i on the first stage according to the step 1 can be given by:

$$C_i \leq \sum_{j=1}^{i} p_{j,1} + \left\{ \begin{array}{c} 0 \quad \text{if} \quad J_i \in B_1 \quad \text{(COND1)} \\ p_{\max,1} + t_1 - s_1 \quad \text{if} \quad J_i \in A_1 \quad \text{(COND2)} \end{array} \right\} \tag{7.31}$$

At step 2 of H, suppose that the LBM rule is applied on the stage 2 tasks with respect to $T := (4 - \frac{1}{m})C^*_{\max}$. Let D_i denote the starting time of J_i on the second stage upon completion of step 2. We will show that $C_i \leq D_i$ for all i. Because of using LBM rule in step 2 we have for all J_i :

$$D_i \geq \left\{ \begin{array}{c} T - \frac{1}{m}\sum_{j=i}^{n} p_{j,2} - \frac{m-1}{m} p_{i,2} \quad \text{if} \quad J_i \in B_2 \quad \text{(COND3)} \\ T - \frac{1}{m}\sum_{j=i}^{n} p_{j,2} - \frac{m-1}{m} p_{i,2} - g_i \quad \text{if} \quad J_i \in A_2 \quad \text{(COND4)} \end{array} \right\} \tag{7.32}$$

Such that g_i is the duration of the gap which is on the machine where J_i is processed (J_i is processed after this gap).

In all configurations whose depend on the position of jobs (i.e before or after gaps) and from 7.31 and 7.32 we have:

$$D_i - C_i \quad \geq \quad (4 - \frac{1}{m})C^*_{\max} - \sum_{j=1}^{i} p_{j,1} - \frac{1}{m}\sum_{j=i}^{n} p_{j,2} - \frac{m-1}{m} p_{i,2} - \left\{ \begin{array}{c} 0 \quad \text{if} \quad \text{COND1 and COND3} \\ p_{\max,1} + (t_1 - s_1) \quad \text{if} \quad \text{COND2 and COND3} \\ g_i \quad \text{if} \quad \text{COND1 and COND4} \\ p_{\max,1} + (t_1 - s_1) + g_i \quad \text{if} \quad \text{COND2 and COND4} \end{array} \right\}$$

Note that :

$$\begin{aligned}
\sum_{j=1}^{i} p_{j,1} + \frac{1}{m}\sum_{j=i}^{n} p_{j,2} &\leq C_{\max}^{AP} \leq 2C_{\max}^{*} \quad \text{for every } 1 \leq i \leq n \\
\frac{m-1}{m} p_{i,2} &\leq \frac{m-1}{m} C_{\max}^{*} \\
p_{\max,1} + (t_1 - s_1) &\leq C_{\max}^{*} \quad \text{if} \quad \text{COND2 and COND3} \\
g_i &\leq C_{\max}^{*} \quad \text{if} \quad \text{COND1 and COND4} \\
p_{\max,1} + (t_1 - s_1) + g_i &\leq C_{\max}^{*} \quad \text{if} \quad \text{COND2 and COND4}
\end{aligned}$$

Then

$$D_i - C_i \geq 0$$

Hence $C_{\max}^{H}$ can not be more than $(4 - \frac{1}{m})C_{\max}^{*}$. ■

Remark 44 *The heuristic H gives better than LPT heuristic if and only if* $m \geq 4$.

Proof. Suppose that $(4 - \frac{1}{m}) \geq 2 + \frac{m}{2}$ hence $(m-2)^2 \geq 2$, i.e, $m \geq \sqrt{2} + 2$ then $m \geq 4$. ■

7.7 Conclusion

In this chapter we have studied the non-resumable regime of the two stage hybrid flow shop with one machine on the first stage and m identical machines on the second stage ($m \geq 1$) problem under availability constraints, with the view of establishing improved optimization procedure and evaluating the performance of some heuristics. We have considered the problem of minimizing the makespan. First we have analyzed the complexity of this problem. To solve optimally the small size problems we have proposed the branch and bound algorithm. For large size problems the performance of three heuristics were given: List Algorithm (LS), LPT heuristic and H heuristic. We have proved that the heuristic H gives better than the LPT heuristic if and only if the number of machines

on the second stage is more than or equal to four machines. Our future research shall be directed towards the heuristics based on branch and bound or dynamic programming and the comparison of their performance with the performance of the three heuristics proposed in this chapter. In the next chapter we intend to apply metaheuristics such as the simulated annealing to the general hybrid flow shop taking on consideration both of preventive maintenance and breakdowns.

Bibliography

[1] **Allaoui, H.**, Artiba, A. and Riane, (2004). "Hybrid flow shop scheduling with availability constraints". Proceedings of MDP-8 conference, vol2, 1249-1254. Cairo, Egypt.

[2] Arthanary, L.S. and Ramamurthy, K.G. (1971). "An extension of two machines sequencing problem". *J.Oper.Res.41,* 641-648.

[3] Blazewicz, J., Breit, J., Formanowicz, P., Kubiak, W., and Schmidt, G. (2001). "Heuristic algorithms for two-machine flowshop with limited machine availability", *Omega 29,* 599-608.

[4] Brah, S.A. and Hunsucker, J.L. (1991), "Branch and bound algorithm for the flow shop with multiple processors", *Euro.J.Of.Oper.Res. 51,* 88-99.

[5] Chen, B., (1995). "Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage". *Journal of Operational Research Society 46* , 234–244.

[6] Garey, M.R. and Johnson, D.S., (1979). "Computers and Intractability: A Guide to the Theory of NP-Completeness". , Freeman, New York .

[7] Gupta, J.N.D., (1998). "Two stage hybrid flowshop scheduling problem". *Journal of Operational Research Society 39 4* (1988). 359–364.

[8] Gupta, J.N.D., Hariri, A.M.A. and Potts, C.N., (1994). "Scheduling a two-stage hybrid flow shop with parallel machines at the first stage". *Annals of Operations Research 69*, 171–191.

[9] Hoogeveen, J.A., Lenstra, J.K. and Veltman, B. (1996). "Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard", *Euro.J.Of.Oper.Res. 89,* 172-175.

[10] Johnson, S.M. (1954). "Optimal two- and three-stage productionschedules with setup times included," *Nav. Res. Log. Quart. 1,* 61-67.

[11] Lee, C.-Y.,.(1997). "Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint", *Oper.Res.Lett. 20*, 129-139.

[12] Lee, C.-Y., (1991). "Parallel machine scheduling with nonsimultaneous machine availabile time", *Discrete Applied Mathematics, 30,* 53-61.

[13] Lee, C.-Y., Lei, L., and Pinedo, M. (1997). "Curent trends in deterministic scheduling", *Ann.Of.Oper.Res. 70,* 1-41.

[14] Lee, C.-Y., Chen, Z.-L. (2000). "Scheduling Jobs and Maintenance Activities on Parallel Machines", *Nav. Res. Log. Vol 47,* 61-67.

[15] Lee, C.-Y. and Vairaktarakis, G.L., (1994). "Minimizing makespan in hybrid flowshops". *Oper.Res.Lett. 16* . 149–158.

[16] Narasimhan, S.L. and Panwalkar, S.S., (1984). "Scheduling in a two-stage manufacturing process". *Int. J. Prod. Res. 22*, 555–564.

[17] Ohno, K., Jin, Z.H. and Elmaghraby, S.E., (2002). "Scheduling hybrid flowshops in printed circuit board assembly lines", *Production and operations Management, 11,2,* 216.

[18] Rajendran, C. and Chaudhuri, D., (1992). "Scheduling in n-job, m-stage flowshop with parallel processors to minimize makespan". *Int. J. Prod. Econ. 27*, 137–143.

[19] Sahni, S., (1976). "Algorithms for scheduling independent tasks". *J. ACM 23.* 116–127.

[20] Schmidt, G. (2000). "Scheduling with limited machine availability", *Euro.J.Of.Oper.Res.121,* 1-15.

[21] Schuurman, P.and Woeginger, G.J., (2000). "A polynomial time approximation scheme for the two-stage multiprocessor flow shop problem", *Theoritical Computer Science, 237,* 105-122.

[22] Sriskandarajah, C. and Sethi, S.P., (1989). "Scheduling algorithms for flexible flow shops: worst and average case performance". *Euro.J.Of.Oper.Res. 43,* pp. 143–160.

Chapter 8

SCHEDULING K-STAGE HYBRID FLOW SHOP WITH MAINTENANCE CONSTRAINTS

8.1 Introduction

It is apparent that there is a gap between the literature of scheduling and the real life industry. The scheduling problems generally are considered with a simplistic way. For instance machines are assumed always to be available during the scheduling horizon. However this chapter is proposed to deal with a practical and stochastic problem. The stochastic character of this problem is represented by taking on consideration the machine breakdowns. We try to optimize several scheduling criteria such as the makespan, the delays and the advances under the constraints of maintenance, cleaning, setup and transportation in an hybrid flow shop. This environment of scheduling is a generic model which can be used to model production systems with serial workshops such as the semiconductor manufacturing systems.

The main objectives of this chapter are three. The first one (section 8.2) is to show how we can integrate simulation using RAO simulator and optimization using a simulated

annealing heuristic combined with some dispatching rules to tackle to the double complexity (algorithmic complexity-structural and functional complexity) of this practical problem. The second one (section 8.5) is to illustrate that the performance of heuristics applied to this problem can be infected by the level of breakdown times. The third one (section 8.6) is to propose an architecture of a Decision Support System (DSS) which gives and compares solutions to schedule complex systems in a reasonable time.

8.2 Integrating Optimization and Simulation

Generally the real life manufacturing systems present two kinds of complexity, an algorithmic complexity and a structural and functional complexity [11] (see Figure8-1).

Although the processing complexity is a major direction of development, it is clearly shown in the literature review that the most of the work is on the 2-stage problems. While interesting results have been obtained for these cases, there has been less work done on the k-stage ($k \geq 3$) ones (see Linn[16]). As it was established by Gupta (see Gupta[12]) the two stage hybrid flow shop is NP-hard on the strong sense. The algorithmic complexity would increase significantly as it extends to three or higher number of stages which is the case for the problem treated in this chapter. To deal with this inconvenient we propose to use a simulated annealing heuristic combined with some dispatching rules like SPT, LPT and EDD. On the other hand additional constraints and objectives of real life industry problems are difficult to express and to take into account by the mathematical modelling. Therefore in order to tackle to the structural and functional complexity of the problem studied in this chapter we propose to use the RAO simulator which comprises a powerful specialized language for simulating complex discrete systems. In order to deal with the double complexity we use a method based on the integration of simulation and simulated annealing (SA) heuristic combined with the dispatching rules cited above for solving this stochastic problem. Such an approach uses SA to guide the simulation. Indeed, the simulation module evaluates the solutions generated by the heuristic module

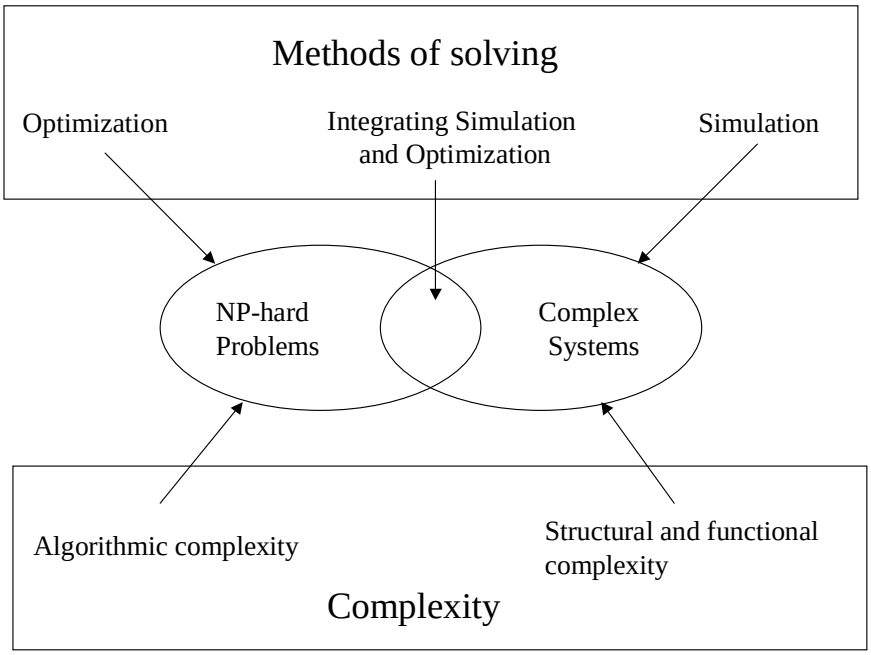

Figure 8-1: The double complexity.

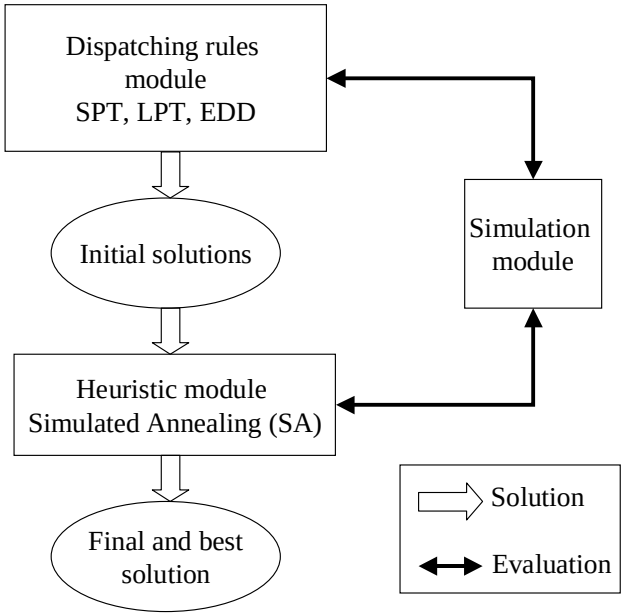

Figure 8-2: Integrating simulation and optimization approach.

(see Figure8-2). In our implementation a solution is a sequence of jobs and which are assigned to machines according to the FAM rule (the first available machine). The dispatching rules module is used to compute initial solutions for the simulated annealing heuristic. Heuristic and dispatching rules are implemented with DELHI programming tool.

8.3 The Simulation Model

For many complex stochastic systems, deterministic numerical methods of analysis have extremely limited utility. By making it feasible to analyze the performance of such systems, simulation has become one of the most widely applied tool of operations research

(see Avramidis[6]). We propose a model for a generic hybrid flow shop which serves two goals: (1) to evaluate the schedule obtained by the scheduling module, taking into account the additional constraints neglected while scheduling, or (2) to construct a schedule taking into account all constraints, assignment rules and objectives of the scheduler. The generic model we propose admits a flexible view capable of capturing hybrid flow shop manufacturing data, materials, process and decisions, providing a highly structured view of the system, from either a data, a process or a decision-making perspective. The model is supported by the RAO simulator (ressource-actions-operations) which is based RAO method for presenting knowledge about complex discrete systems and processes (see Artiba[5]). The system's resources set is the data base (DB) for that production system while the system's knowledge base (KB) consists of the set of operations (actions) fulfilled by the resources and described by the modified production rules (**If** (conditions) **Then** (action 1) **Wait** (time interval) **Then** (action 2)). The generic hybrid flow shop model is described using the ressources-actions-operations approach which requires the definition of the following objects: jobs, machine tools, stocks and transport facilities as depicted in Figure8-3.

The model simulates, among others, the following operations: allocation of jobs to machines, setup, transporting, processing, cleaning, preventive maintenance, stochastic breakdowns.

- Setup time is calculated in accordance with a setup matrix, depending in general case on the current job i, the precedent job j and the considered machine of the current stage. The setup times are represented in the simulation model as a function (see the example bellow with two machines and tree jobs):

$Function F_Setup_time_table : real
$Type = table
$Parameters
machine_Number : integer [1..2]

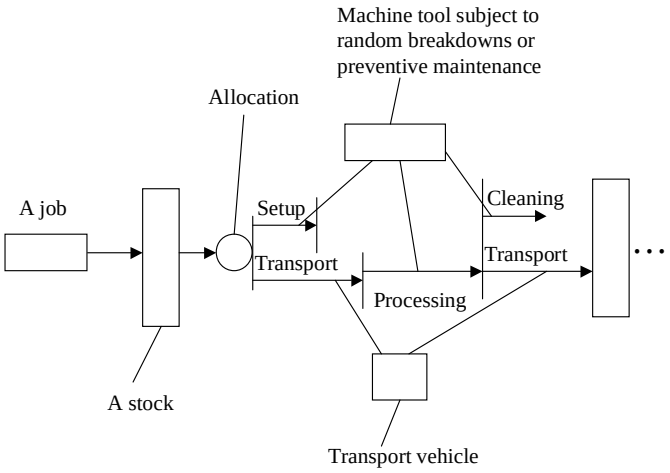

Figure 8-3: A schema of the process simulated.

job_to : integer [1..3]

job_from : integer [1..3]

$Body

0.1	0.1	0.1
0.2	0.2	0.2
0.3	0.3	0.3
0.1	0.1	0.1
0.2	0.2	0.2
0.3	0.3	0.3

$End

- A matrix in the model gives the number of the group of transport facilities for each job, each process stage and each machine tool. Transportation and cleaning times are also represented by matrixes used in the simulation model. For example the matrix giving the number of transport facilities for each job to each machine is illustrated in this example:

$Function F_Transp_number_to : integer

$Type = table

$Parameters

machine_Number : integer [1..3]

job_Number : integer [1..3]

$Body

1	2	1
2	4	2
1	2	1

$End

- The preventive maintenance is represented in the simulation model by two parameters the duration and the periodicity. Each of them is given by a matrix depending

in general case on the considered machine and the type of preventive maintenance (see the example bellow representing the function of duration for tree machines and fore types of preventive maintenance):

```
$Function Start_Preventive_time : real
$Type = table
$Parameters
Preventive_Number : integer [1..4]
Machine_Number : integer [1..3]
$Body
    0.1    0.2    0.3    0.4
    0.1    0.2    0.3    0.4
    0.1    0.2    0.3    0.4
$End
```

- The breakdowns are represented in the simulation model by two random variables the duration of down time (we suppose that down time = repair time) and the start time (see in the example bellow representing the start time of breakdowns by an exponential random variable:

```
$Sequence Start_Breakdown_time
$Type = exponential
$End
```

The job enters the hybrid flow shop and waits in a stock to be allocated for the first stage according to an order S (for the other stages a job is affected to a machine such that it can be finished as early as possible). Once the job is allocated, two actions are started: set-up and transportation. Set-up time is given by the set-up matrix. Transportation can only start if there is a free transport vehicle in the correspondent group of transport facilities for each job, each process stage and each machine tool. Transportation time is also given by a matrix. Furthermore a machine can be subject to a preventive period

during or not during the processing of a job. However a machine can be subject to a random breakdown only during the processing operation of a job. The model is adaptable to any particular industrial case if one can easily describe all the resources of the real system (machine tools, transport facilities and stocks), the set of jobs and their characteristics and all the matrix determining times of preventive maintenance, breakdowns, setup, processing, etc.

8.4 Optimization Approach to HFS Scheduling

In the HFS scheduling literature, heuristics and B&B are the two mostly employed. When we are confronted with the fact that the problem is NP-hard and the optimal solution is unattainable in reasonable time by exact methods which is the case of our problem then it is reasonable to sacrifice optimality and settle for a "good" feasible solution that can be computed efficiently. Of course, we would like to sacrifice as little optimality as possible, while gaining as much as possible in efficiency. In this chapter we propose to use the simulated annealing (SA). SA is an iterative optimization technique that frees from local optima with following method: a new solution that is accepted with a probability called the acceptance function. This function is computed from the evolution of the optimized criteria and a control parameter called the temperature. This short description of SA shows that the algorithm is fully defined with the following elements:

- The representation of the solution and the evaluation function. These two elements are generally strongly connected to the problem but they have an effect on the global behavior of the algorithm;
- The definition of the neighborhood, i.e. the method used to choose a new solution from the current solution;
- The outline of the acceptance function. This outline depends on the expression used to compute the probability and it depends on the temperature from an iteration to

the next.

In the scheduling problem of concern to us, the objective is to minimize a function:

$$H : X \rightarrow R$$
$$S \rightarrow H(S)$$

Where S is a permutation of the jobs $\{1, ..., n\}$ and X is the associate space with a finite but exponentially large of points. We define a topology on X as follows: given a solution $S = \{\pi(1), ..., \pi(n)\}$, its neighbors might be all sequences generated by interchanging two jobs randomly chosen. The pseudo code of the SA algorithm is given bellow:

- Set the control parameters $(T_0, T_{\max i}, Nb_{iter})$ and initialize $T = T_{\max i}$;
- $S =$ **INITIAL_SOLUTION**;
- Evaluate the objective function $H(S)$;
- $Best\_Solution = S$;
- While $T \succeq T_0$ do

{

- Repeat Nb_{iter} times

{

- $S\prime = NEIGHBOR(S)$;
- **Evaluate_objective_func** $H(S\prime)$ and let $\triangle H = H(S') - H(S)$
- If $H(S') - H(S) \prec 0$ accept $S\prime$;
  - If $H(S') \prec H(Best\_Solution)$ then $Best\_Solution = S'$;
- Else;

- Evaluate $p = e^{-\left(\frac{\triangle H}{T}\right)}$ and generate r, a random number uniformly distributed between 0 and 1.
- If $p > r$, accept the new solution $S\prime$ otherwise reject $S\prime$ and retain the previous solution S.

}

- DECREASE (T);

}

Because of the stochastic character of this problem *30* separate problems are generated for each of the problem size. For each randomly generated case and at each evaluation of the objective function $H(S)$ which can be one of the criteria cited above, the procedure **Evaluate_objective_func** executes the RAO simulator and then recuperates the result of the simulation. For each randomly generated instance the initial solution **INITIAL_SOLUTION** is given by either SPT, LPT or EDD dispatching rule. If the initial solution is given by the rule R we denote the heuristic by $SA(R)$. For each of the 30 randomly generated instances we evaluate the final solution given by the heuristic. Then we calculate the average of the 30 simulations.

8.5 Experimental Study

According to the literature review presented in this paper, the problem of scheduling a k-stage hybrid flow shop with such constraints has never been studied before. In this paper although heuristics are applied in a simplistic way to assign jobs to machines, but in order to illustrate the important role of our approach of integrating optimization and simulation to deal with a such complex system we evaluate in this section by an experimental study the three heuristics based on the three widely used dispatching rules LPT, SPT and EDD and the simulated annealing.

For our experimentations several runs of the simulation model are needed since we deal with a stochastic problem (breakdowns) and heuristics are stochastic (based on

simulated annealing). The aim of this experimental study is to evaluate the impact of one of breakdown parameters which is the percentage of repair time P_r, on the performances of the heuristics SA(SPT), SA(LPT) and SA(EDD) with respect to the criteria $C_{\max}$, $\overline{F}$, $\overline{T}$, $T_{\max}$, and N_t, according to the two processing cases resumable (r) and non-resumable (nr).

We fix the number of jobs to 50, the number of stages to 4 and the maximum number of machines per stage to 5. The parameters of the simulated annealing except the initial solution are also fixed. The number of iterations to 100, the maximum of temperature to 100, the minimum of temperature to 0.1 and the cooling factor to 0.9. All other instance parameters (preventive, cleaning, setup and transportation) are fixed. Our approach of integrating simulation and optimization with the listed data runs in about 30 min on a Pentium IV 1.5 GHz. The percentage of repair time P_r takes either 0%, 3% or 5%. All calculations are given by using WEIBULL distribution of random breakdowns. For each heuristic, each column shows the mean value reached for the corresponding criterion. The mean is computed over the 30 randomly generated instances.

8.5.1 Resumable case

Table 1 presents the different criteria $C_{\max}$, $\overline{C}$, $T_{\max}$, $\overline{T}$ and N_t for the three heuristics $SA(LPT)$, $SA(SPT)$ and $SA(EDD)$, in dependence of the percentage of the repair time P_r in resumable case. For minimizing the $C_{\max}$, the heuristic SA(LPT) is the best heuristic and its performance is not infected by the percentage of the factor P_r. We can also conclude that the heuristic SA(SPT) minimizes the $\overline{C}$ objective and its performance is not in dependence of P_r.

Table1: The criteria in dependence of P_r in resumable case.

Heuristic	P_r	$C_{\max}$	$\overline{C}$	$T_{\max}$	$\overline{T}$	N_t
$SA(LPT)$	0%	835.11	531.25	52.32	33.26	11
	3%	847.25	591.97	54.88	35.45	11
	5%	942.16	731.50	61.08	36.94	12
$SA(SPT)$	0%	841.4	525.24	49.76	25.15	8
	3%	861.06	545.62	51.48	26.15	8
	5%	951.72	725.53	59.60	33.21	12
$SA(EDD)$	0%	892.20	528.93	15.64	9.36	3
	3%	914.96	568.19	19.32	12.88	6
	5%	954.28	728.06	56.21	31.75	12

However for the criteria $T_{\max}$, $\overline{T}$ and N_t the SA(EDD) is significantly the best solution for the two values $P_r = 0\%$ and $P_r = 3\%$. For the value $P_r = 5\%$ the heuristic SA(EDD) gives almost or the same thing of the other two heuristics (see the example of $\overline{T}$ in Figure8-4). We can conclude that, in resumable case, the performance of heuristics can be infected by the percentage of repair time only when we deal with criteria based on the tardiness.

8.5.2 Non-Resumable case

Table 2 presents the different criteria $C_{\max}$, $\overline{C}$, $T_{\max}$, $\overline{T}$ and N_t for the three heuristics $SA(LPT)$, $SA(SPT)$ and $SA(EDD)$, in dependence of the percentage of the repair time P_r in non-resumable case.

Table2: The criteria in dependence of P_r in non-resumable case.

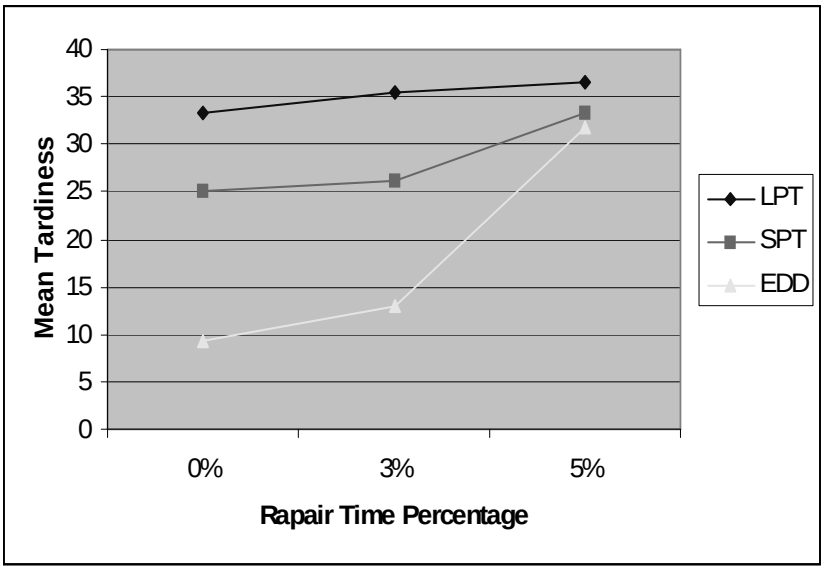

Figure 8-4: $\overline{T}$ in dependence of P_r.

Heuristic	P_r	$C_{\max}$	$\overline{C}$	$T_{\max}$	$\overline{T}$	N_t
$SA(LPT)$	0%	872.68	544.31	53.60	34.78	11
	3%	911.53	591.25	58.76	41.39	11
	5%	1201.76	825.26	72.62	52.27	14
$SA(SPT)$	0%	875.12	527.89	51.20	27.43	8
	3%	905.11	572.62	55.61	34.15	8
	5%	1215.21	830.97	69.32	50.91	14
$SA(EDD)$	0%	895.25	538.52	17.04	10.11	3
	3%	923.23	587.34	21.32	15.63	6
	5%	1232.56	833.63	65.45	49.84	14

For minimizing the makepan $C_{\max}$, and for the two values $P_r = 0\%$ and $P_r = 5\%$ the heuristic $SA(LPT)$ is the best solution. However for the value $P_r = 3\%$ the best solution is the heuristic $SA(SPT)$.Hence we can conclude that, in non-resumable case, the performance of heuristics can be infected by the repair time percentage when we deal with the $C_{\max}$ criterion, (see Figure8-5).

To minimize the mean flow time $\overline{C}$ in non-resumable case the heuristic $SA(SPT)$ is always the best solution except for the value $P_r = 5\%$ where the heuristic $SA(LPT)$ is the optimal solution. The performance of heuristics can also be infected by the factor P_r. Finally like the resumable case the performance of heuristics are in dependence of P_r for minimizing criteria based on tardiness.

8.6 A Decision Support System

Dealing with the real life production planning and scheduling problems, the main objective is to find a good schedule which minimizes the cost and the time of production and to respect deadlines with a good high machine and labour utilization. To meet this objective a decision support system for a general hybrid flow shop scheduling system is developed for entering and validating the scheduling problem data, building various

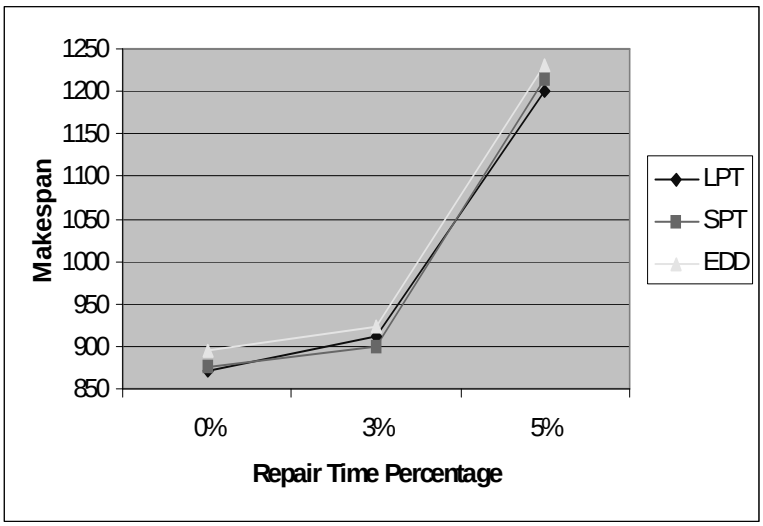

Figure 8-5: $C_{\max}$ in dependence of P_r.

solutions schedules, and evaluating their performance.

The goal was to design and develop an interactive flexible scheduling system that can be easily adapted to many different manufacturing environments. It was decided to start with supporting an Hybrid Flow Shop environment with some Heuristics and Meta-heuristics, but the system is implemented to be flexible to add other environments and algorithms. An other advantage of this system is that it integrates practical constraints such as maintenance activities, setup, transportation and cleaning.

Our system is implemented to deal with stochastic and deterministic data. Hence it is built around four major modules (see Figure8-6 and Figure8-7): a database module to deal with deterministic data, a stochastic data module to deal with stochastic data such as stochastic processing times or breakdown times, scheduling module to optimize some criteria in the environment described in database or stochastic data module, and tools module which give us the possibility to show and analyze system performances.

8.6.1 The interface module

First the interface module gives the users the possibility to introduce the necessary inputs to describe the instance of the hybrid flow shop scheduling problem (See Figure8-8). Thus the number of stages, the number of jobs and the processing case should be introduced.

After the scheduling environment, the constraints of the problem and the criteria to optimize may be described by introducing:

- Data of the hybrid flow shop environment and scheduling constraints. That means that the user can introduce the number of stages in the hybrid flow shop, the number of machines in each stage. Concerning the constraints, users can for example describe preventive maintenance activities by introducing the number of gaps on each machine which should be fixed and known in advance (See Figure8-9).

- Data giving the set of jobs to schedule. In other term the user can introduce for example, the number of jobs and the processing time of each job (See Figure8-10).

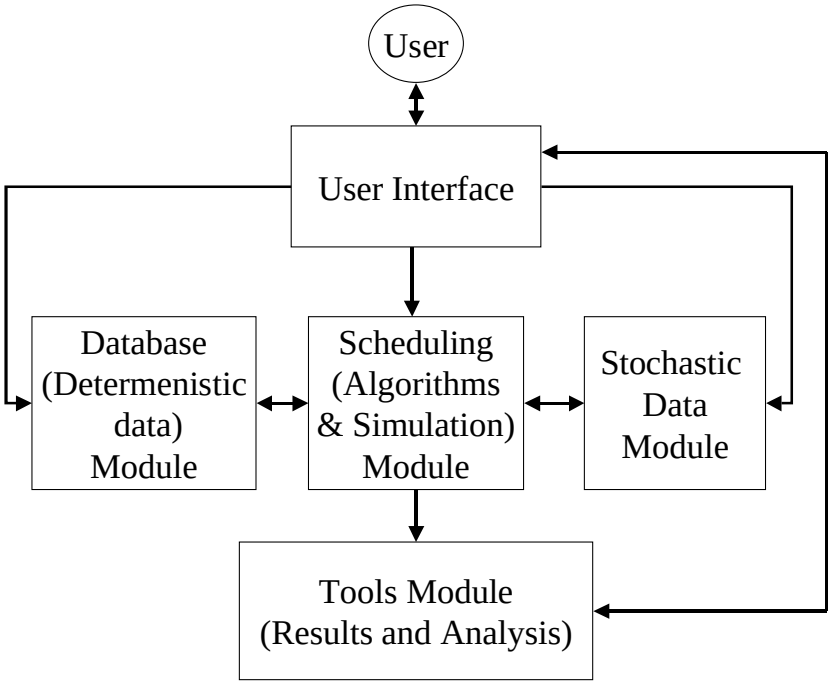

Figure 8-6: The DSS architecture.

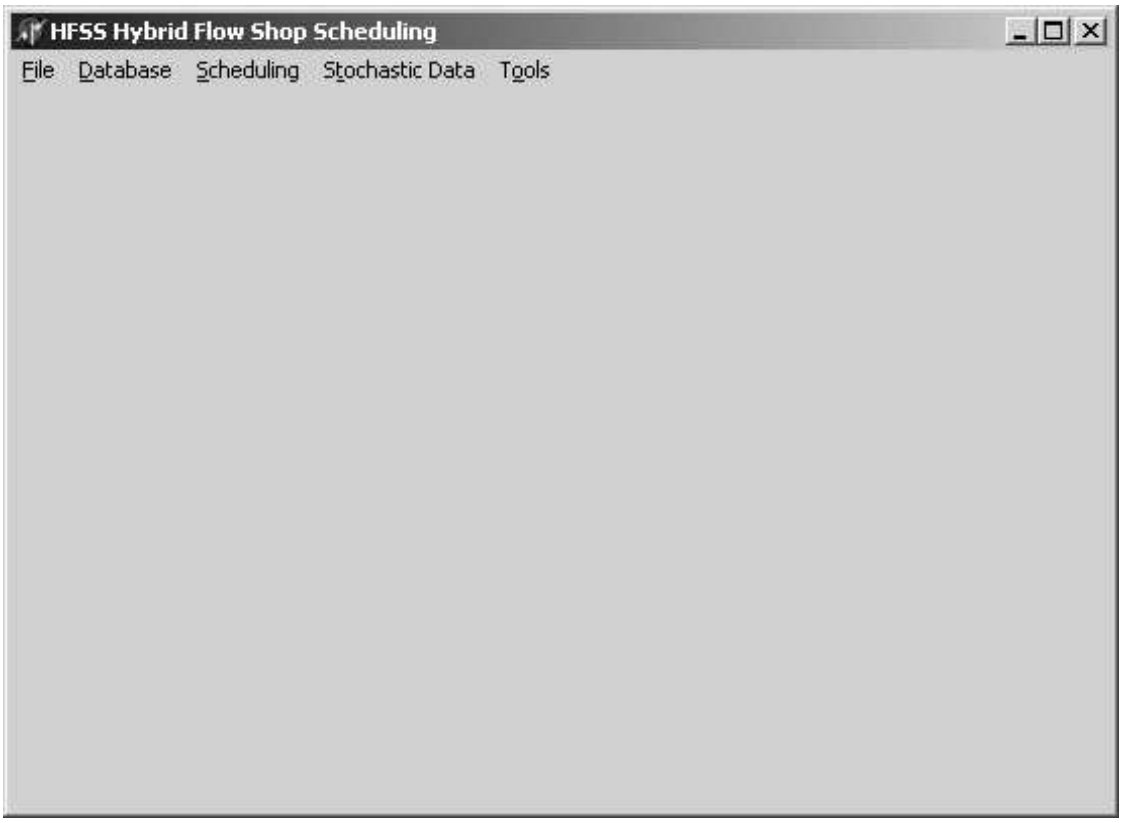

Figure 8-7: The DSS user interface.

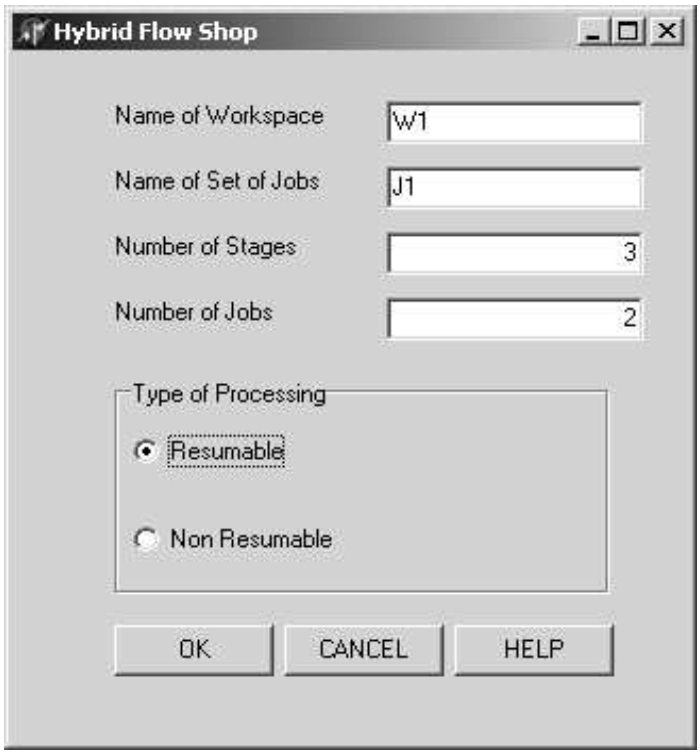

Figure 8-8: The hybrid flow shop problem.

- The choice of criteria to minimize (See Figure8-11). After using one of solution methods proposed in this DSS the performances of this method can be posted by choosing "Performances" in Tools module (see Figure8-12).

8.6.2 The database module

The database is basically organized into two parts: job data and parkmachine (or the workspace) data. The handling of the job data is the domain of the job manager (See Figure8-13). The parkmachine data are handled by the environment manager. For each specific scheduling problem, a set of files are created to define the layout of the stages and machines as well as the set of jobs to be scheduled.

8.6.3 The scheduling module

The scheduling module consists of five submodules (See Figure8-14). The first one contains some simple dispatching rules like SPT, LPT and EDD. The second one contains some heuristics which are known to give good performances for an hybrid flow shop problem like CDS. The third one contains two metaheuristics, simulated annealing and tabu search. The metaheuristics require from the user to introduce parameters like the number of iterations, the temperature for simulated annealing and aspiration factor for a tabu search (See Figure8-15). If the complexity of the problem provides the use of simulation, the fourth submodule "SimOpt" proposes the approach of integrating optimization and simulation illustrated in section 8.2. Some solution methods can be applied to any instance of an hybrid flow shop problem and others can be applied only for some particular instances. The last one will give the possibility to introduce a manual scheduling by the users.

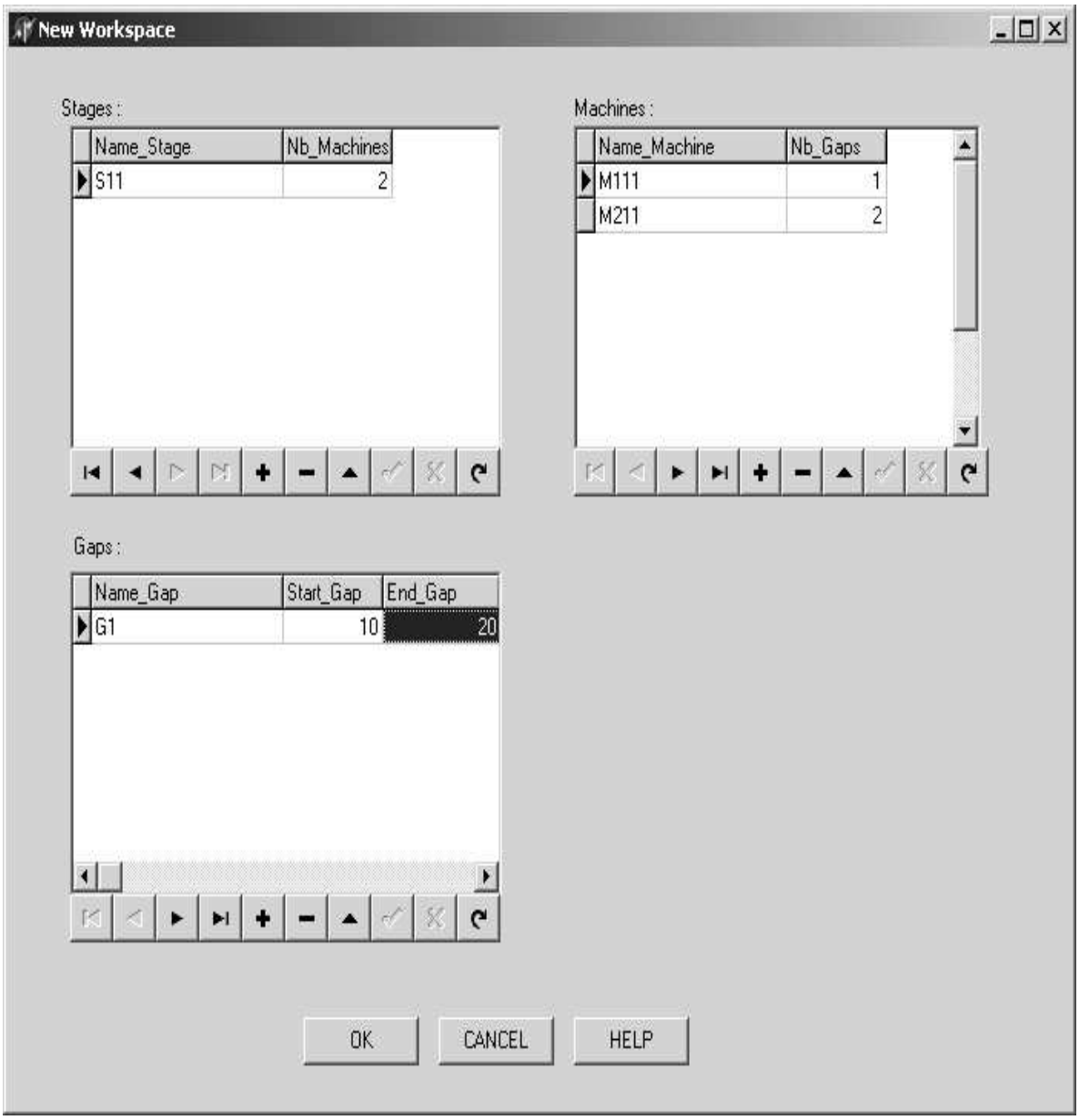

Figure 8-9: Workspace description.

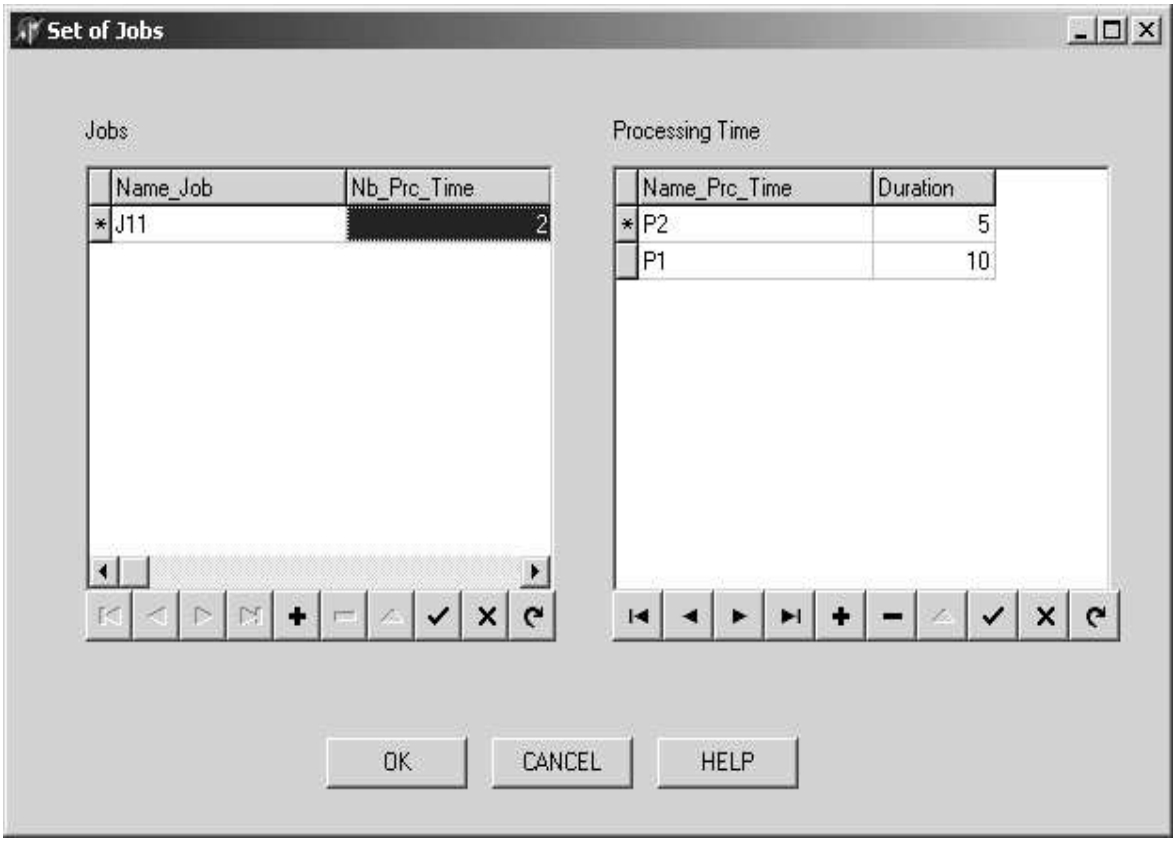

Figure 8-10: Set of jobs.

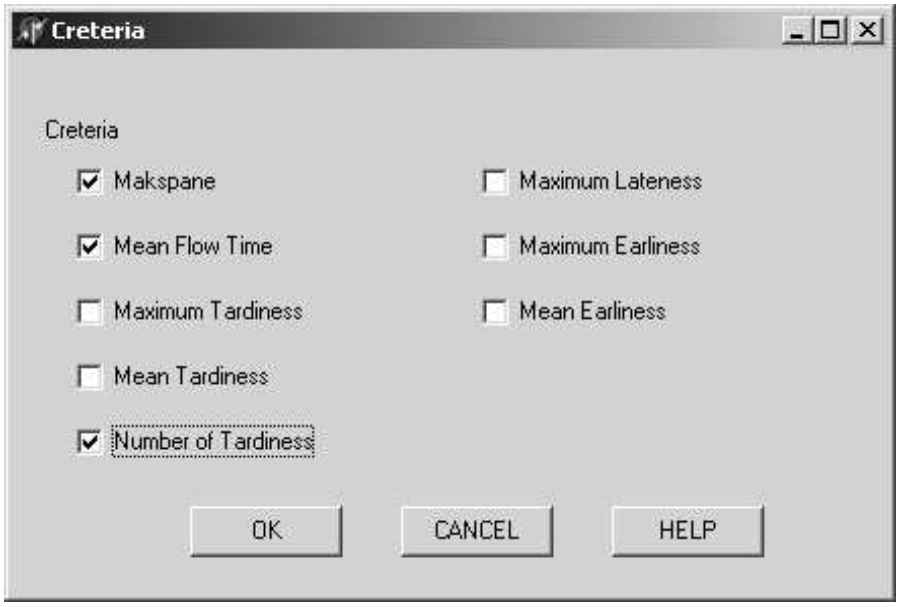

Figure 8-11: Creteria to optimize.

8.7 Conclusion

In this chapter we have investigated the problem of scheduling a general hybrid flow shop to minimize flow time and due date based criteria under maintenance constraints. Setup, cleaning and transportation times are considered. Because of the higher algorithmic and functionnal-structural complexity of this problem we have investigated the approach of integrating the simulation and the optimization to deal with. In such scheduling environment we have illustrated that in resumable case the performance of heuristics can be infected by the percentage of repair time only when we deal with due date based criteria. However in non-resumable case both flow time and due date based criteria are in dependence of this factor. A decision support system tool is proposed to enable users to compare many solution methods applied to any instance of an hybrid flow shop. Many other additional developments are required to finalize this tool in order to validate it with real life industry applications.

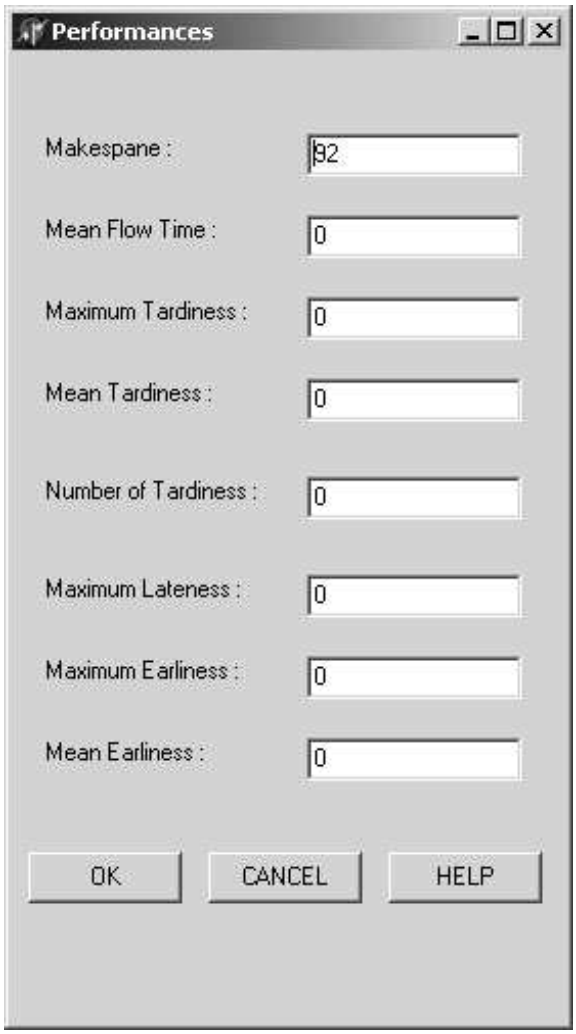

Figure 8-12: Solution method's performances.

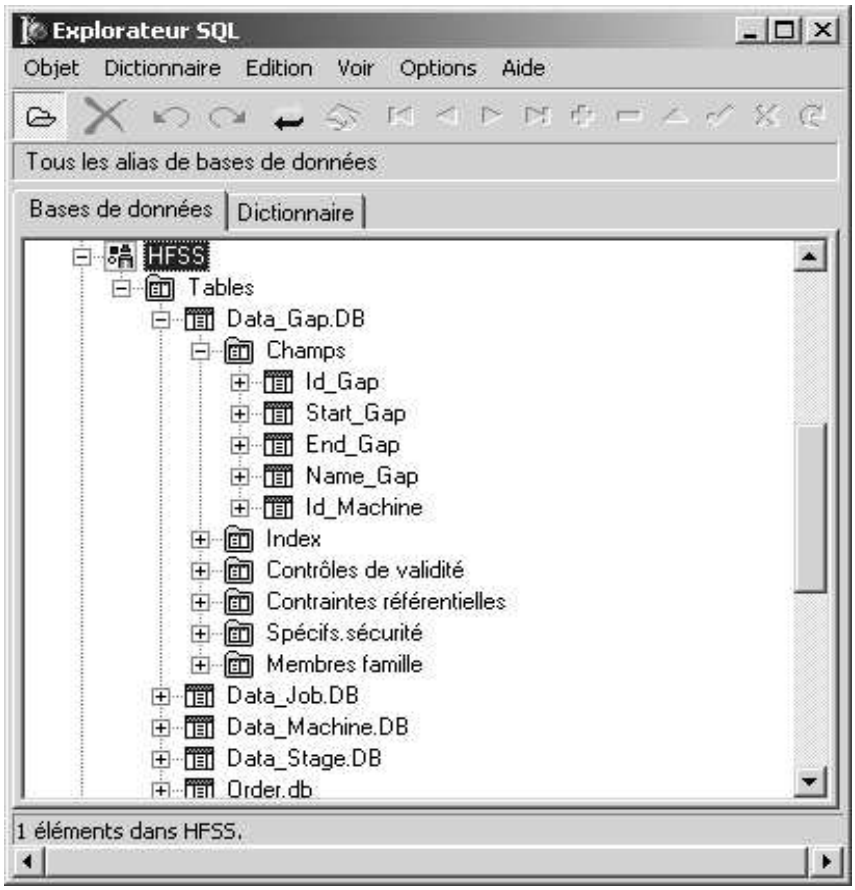

Figure 8-13: Database of the scheduling system.

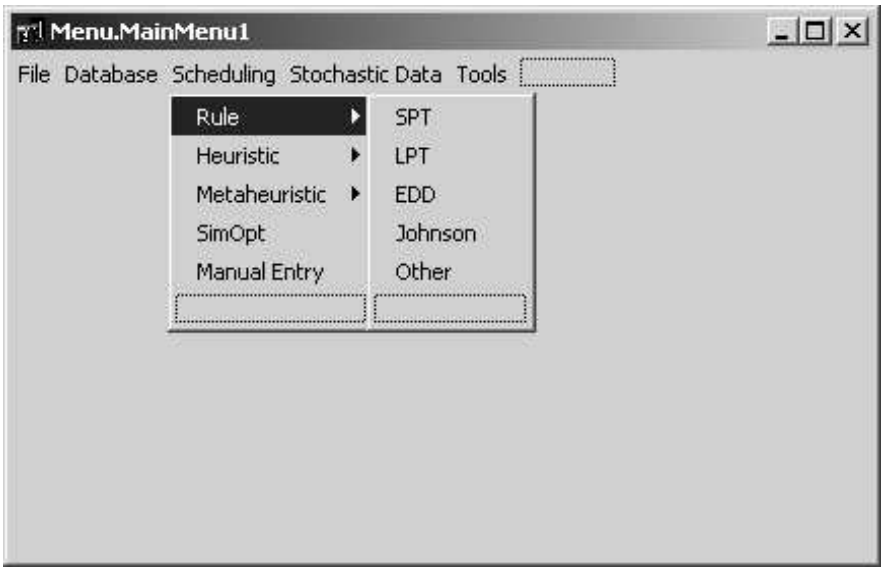

Figure 8-14: Scheduling Module.

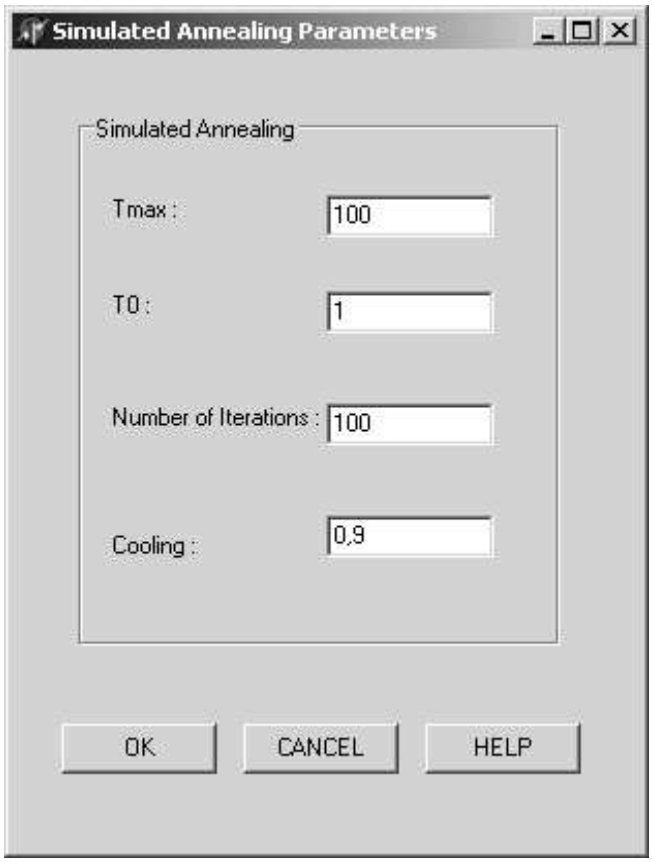

Figure 8-15: Simulated annealing parameters

Bibliography

[1] Allahverdi, A., Mittenhal, J., (1994). "Scheduling on M parallel machines subject to random breakdowns to minimize expected mean flow time. *Nav. Res. Log. Quart.41.* 677-682.

[2] **Allaoui, H.**, Artiba, A. and Lamouri, S., (2004). "Integrating simulation and optimization to schedule an hybrid flow shop with maintenance constraints". To appear in Comp & ind eng.

[3] **Allaoui, H.**, (2002). "Scheduling hybrid flowshops with availability constraint : Dynamic Programming and Heuristics". DEA at Mons Hainault University, Belgium.

[4] **Allaoui, H**., Pichel, D. and Iassinovskaia, G., (2002,2003,2004). "WORKSHOP: Conception d'un outil de gestion stratégique et opérationnelle des systèmes productifs.". Research Reports 1,2,3,4 and 5. CREGI laboratory, FUCAM university.

[5] Artiba, A., Emelyanov, V., and Iaasinovski, S.(1998). "Introduction to Intelligent Simulation: The RAO Language", *Kluwer Academic Publishers*, Dordercht.

[6] Avramidis, A.N., and Wilson, J.R., (1996). "Integrated variance reduction strategies for simulation". *Operations Research 44(2).* 327-346.

[7] Birge, J., Frenk, J.B.G., Mittenhall, J., and Rinnooy Kan, A.H.G., (1990), "Single-machine scheduling subject to stochastic breakdowns", *Nav. Res. Log. Quart.37.* 661-677.

[8] Chen, B., (1995). "Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage". *Journal of Operational Research Society 46* , 234–244.

[9] Garey, M.R. and Johnson, D.S., (1979). "Computers and Intractability: A Guide to the Theory of NP-Completeness". , Freeman, New York .

[10] Glazebrook, K.D., (1984). "Scheduling stochastic jobs on a single machine subject to breakdowns". *Nav. Res. Log. Quart.31.* 251-264.

[11] Grangeon, N. "Métaheuristiques et modèles d'évaluation pour le problème du flow shop hybride hiérarchisé: Contexte déterministe et contexte stochastique", Thesis at Université Blaise Pascal.

[12] Gupta, J.N.D., Hariri, A.M.A. and Potts, C.N., (1994). "Scheduling a two-stage hybrid flow shop with parallel machines at the first stage". *Annals of Operations Research 69*, 171–191.

[13] Johnson, S.M. (1954). "Optimal two- and three-stage productionschedules with setup times included," *Nav. Res. Log. Quart. 1,* 61-67.

[14] Langston, M.A. (1987). "Interstage transportation planning in the deterministic flow-shop environment", *Oper.Res.35(4),* 556-564.

[15] Lee, C.-Y.,.(1997). "Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint", *Oper.Res.Lett. 20*, 129-139.

[16] Linn, R. and Zhang, W., (1999). "Hybrid flow shop scheduling: a survey". *Comp & Ind Enginneering.* 37, 57-61.

[17] Mittenthall, J., and Ragavachari, M. (1993). "Stochastic flowshops", *Operations research.30,* 148-162.

[18] Rajendran, C. and Chaudhuri, D., (1992). "Scheduling in n-job, m-stage flowshop with parallel processors to minimize makespan". *Int. J. Prod. Econ. 27*, 137–143.

[19] Schmidt, G. (2000). "Scheduling with limited machine availability", *Euro.J.Of.Oper.Res.121,* 1-15.

[20] Sriskandarajah, C. and Sethi, S.P., (1989). "Scheduling algorithms for flexible flow shops: worst and average case performance". *Euro.J.Of.Oper.Res. 43,* pp. 143–160.

[21] Wittrock, R.J., (1988). " An adaptable scheduling algorithm for flexible flow lines". *J. Oper. Res.* 36, 445-53.

Chapter 9

CONCLUSION

9.1 Summary

Motivated by the idea of integrating the two functions production and maintenance, and by a scheduling environment commonly found in many industrial applications which is the hybrid flow shop, this thesis studied the hybrid flow shop scheduling with maintenance constraints. The problem of machine scheduling with availability constraints has attracted much attention in the scheduling field recently.

In this thesis we have presented a detailed survey of problems treated on the two fields hybrid flow shop and scheduling with availability constraints. The problem of scheduling hybrid flow shop with maintenance constraints was investigated in this thesis on two categories namely the two stage and k-stage hybrid flow shop. On the first category two problems are studied, the two machine flow shop problem and the two stage hybrid flow shop problem with only one machine on the first stage and m machines on the second stage.

In chapter six we have studied the resumable and the non-resumable modes of the two-machine flow shop problem with an availability constraint either on the first machine or on the second machine to minimize the makespan, with the view of establishing improved optimization procedures and evaluating the performance of Johnson's Rule. We have

focused on Johnson's Rule because of its simplicity and its common knowledge. We have specified the conditions under which Johnson's Rule gives the optimum. We have proposed a dynamic programming if the gap is on the first machine and a combinatorial approach if the gap is on the second machine. Both algorithms are more efficient than currently available algorithms if the number of jobs is large and the gap occurs near the middle of the optimal sequence in the absence of the gap. We have demonstrated that the performance bound of Johnson's Rule can be more better than 2 under "normal" conditions (the duration of the gap is a small fraction of the optimal makespan in the absence of a gap).

The two stage hybrid flow shop with one machine on the first stage and m machines on the second stage ($m \geq 1$) problem under availability constraints, was investigated in chapter seven. We have considered the problem of minimizing the makespan. First we have analyzed the complexity of this problem. To solve optimally the small size problems we have proposed the branch and bound algorithm. For the large size problems the performance of three heuristics were given: List Algorithm (LS), LPT heuristic and H heuristic. We have proved that the heuristic H gives better than the LPT heuristic if and only if the number of machines on the second stage is more than or equal to four machines.

In chapter eight we have investigated the problem of scheduling a k-stage hybrid flow shop to minimize flow time and due date based criteria under maintenance constraints. Setup, cleaning and transportation times are considered. We have presented the approach of integrating the simulation and the optimization to deal with this problem. Three dispatching rules LPT, SPT and EDD, the simulated annealing (SA) heuristic and a flexible simulation model are used in this approach. The results of the experimental study reveal that in resumable case the performance of heuristics can be infected by the percentage of repair time only when we deal with due date based criteria. However in non-resumable case both flow time and due date based criteria are in dependence of this factor.

9.2 Outlook Of The Future Research

Scheduling is an exciting and challenging area. Because of their huge complexity some problems were not or not well studied. The research contained within this thesis may be extended in a number of directions. Some of these are:

- To deal with more general configurations of a two machine flow shop problem with availability constraints. For example to consider multiple gaps on the two machines. Other criteria can be studied for this problem based for example on due date or release date.

- For the two stage hybrid flow shop, one possible direction is to develop heuristics based on branch and bound and dynamic programming algorithms and compare their performances with those of heuristics studied in this thesis and methaheuristics in both cases resumable and non-resumable.

- A multi-criteria study can be detailed to optimize a k-stage hybrid flow shop problem investigated in this thesis using the approach of integrating simulation and optimization in both cases resumable and non-resumable.

- The decision support system proposed shall be validated with industrial applications.

- It will be very interesting to study the problem to have the decision of maintenance and the decision of job-scheduling made jointly.

Constraint Logic Programming (CLP) has been established as a practical tool for solving combinatorial problems and some industrial sectors like transportation, telecommunication and manufacturing. As one application of manufacturing a scheduling problems can be suitably modelled in terms of a set of constraints and then solved by constraint logic programming. One of our future research will be directed to the study of application of this tool to the scheduling under maintenance constraints problem.