CHAPTER II
THE HYBRID FLOWSHOP SCHEDULING PROBLEM

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

LIST OF ACRONYMS

FS	*Classical Flowshop*
GA	Genetic Algorithm
GP	Global Procedure
GPH	Global Planning Heuristic
GSH	Global Scheduling Heuristic
HCPL	Heuristic for Checking the Planned Load
HFS	Hybrid Flow Shop
JB	Job Bound
LB	Lower Bound
LPT	Longest Processing Time dispatching rule
LWR	Least Work Remaining
MMP	Makespan Minimization Problem when preemption is not allowed
MSS	Makespan of the Single Stage s subproblem
MWR	Most Work Remaining
PB	*Preemptive Bound*
SB	Set Bound
SBP	Shifting Bottleneck Procedure
SPT	Shortest Processing Time dispatching rule
SSB	*Subset Bound*

CHAPTER II
THE HYBRID FLOWSHOP SCHEDULING PROBLEM

Abstract

This chapter aims at presenting the HFS short-term scheduling problem. Scheduling consists in building a short-term production plan, once the upper-level planning system has determined the set and the lot sizes of items that need to be produced within one period. The HFS scheduling problem we are to study in this project is first introduced, and then a survey of the HFS scheduling literature is presented. Based on the state of the art of general scheduling solutions, we present the research orientation we have decided to pursue in this project, that is the study of the branch and bound techniques. The graph representation of the HFS scheduling problem helps to understand the problem and to design scheduling solutions. In such a graph representation, the HFS can be regarded as being made up of a set of single stage subproblems, each single stage being composed of parallel machines. Therefore, a relaxation of the HFS problem into a single stage subproblem can be inferred. This relaxation allows designing lower and upper bounds on the HFS global problem. The single stage subproblem will then be introduced and some known solutions recalled. Single stage subproblem solutions are used as subroutines in the algorithms developed in this thesis.

1. INTRODUCTION

In Chapter I we have presented a general study of the planning and scheduling problem in a hybrid flowshop facility. We proposed a Global Procedure (GP) or a decomposition-based heuristic whose main issue is the effective use of the available capacity. GP is composed of two main procedures. The Global Planning Heuristic (GPH) sets the lot sizes of items over the planning horizon. Once the plan is built, a 'Heuristic for Checking the Planned Load' (HCPL) checks the feasibility of the global planning load by scheduling the corresponding lot sizes over the planning horizon. If the plan is infeasible GPH is rerun with a fine tuned capacity estimation. The procedure is repeated until the plan has proved feasible. Eventually, the Global Scheduling Heuristic (GSH) is used to build a final schedule for the first period. The critical and most time-consuming part of GP is its scheduling part. Indeed, scheduling might be used several times by HCPL and once or few times by GSH.

At the beginning of our research a review of the literature (see Section 2) has shown that the multistage HFS scheduling problem has not been well-studied and effective algorithms for finding good solutions for large problems do not exist. Our research has then been oriented towards the scheduling of the HFS, which will be carried out through the design of branch and bound algorithms. We decided to study the HFS configuration where each stage is made up of identical parallel machines and where jobs do not have setup times. As a matter of fact, this problem is already a hard one, and has not been well studied. See later Section 2.

THE HFS SCHEDULING PROBLEM

The multiprocessor flowshop or Hybrid Flow Shop (HFS) consists of a multistage flowshop, where each stage $s \in \{1, ..., K\}$, $K \geq 2$, is composed of M_s parallel machines, identical in our case (Figure 2.1). Each machine is able to process one job at a time. A multistage flowshop is considered to be a hybrid flowshop if one stage at least is made up of more than one machine.

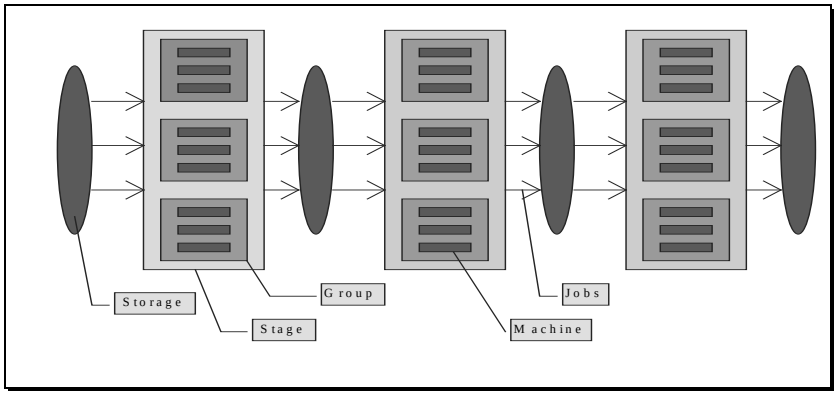

Figure 2.1: A three-stage Hybrid Flow Shop

Our task consists of scheduling a set of jobs $I = \{1, ..., n\}$, where a job consists of one operation for each stage. Those operations have to be realized sequentially, without overlapping between stages. Job preemption and job splitting are not allowed.

Each job i has a given processing time p_{is} at each stage s, $i \in I$ and $s \in \{1, ..., K\}$. Each job may have an initial release date r_{i1} at the first stage and a final tail q_{iK} at the last stage K.

More generally, for each stage $s \in \{1, ..., K\}$ and each job $i \in I$,

- The release date r_{is} stands for the period (hour, day, …) at the end of which job i is ready for processing at stage s. Job i can then start processing at stage s, at the beginning of period $r_{is}+1$,
- The tail q_{is} can be regarded as the minimum amount of time a job, after being processed at stage s, has to spend in the shop before the makespan (i.e. the time by which all jobs are completed) is reached. For $s<K$, the tail q_{is} can be considered to be larger than or equal to $\sum_{t=s+1}^{K} p_{it} + q_{iK}$.

The HFS model and the methods designed in this project can easily be generalized to the case where jobs have waiting times between stages. This is done by adapting the computation of release dates and tails (see Section 5). All data of the problem (i.e. p_{is}, r_{i1}, q_{iK}; $i \in I$, $s \in \{1,...,K\}$) are assumed integer.

We need to build a schedule with a minimum makespan, where the makespan μ defined by $\mu = \max_{i \in I}[t_{iK}+p_{iK}+q_{iK}-1]$, where $t_{iK}>r_{iK}$ is the starting time of the last (K^{th}) operation of job i, which is thus finished at time $t_{iK}+p_{iK}-1$. A schedule consists of assigning a specific machine to each operation of every job as well as sequencing all operations assigned to

each machine, so that successive operations of a job do not overlap and so that each machine processes at most one job at a time.

In such short term scheduling problems the set of jobs usually corresponds to a pre-defined or selected set of lots or orders which have to be produced within a short period of time, for example one week or one day. This set of lots has been defined either by a higher level or longer-term requirement planning system or as a selection among firm orders. Hence, the objective is to schedule the selected lots in the shortest possible time. In our planning approach, presented in Chapter I, Section 5, this set of lots is defined by the Global Planning Heuristic (GPH).

The subsequent chapters of this thesis aim at designing branch and bound algorithms through the study and development of upper and lower bounds as well as branching schemes. For the lower and upper bounds, several methods will be developed and compared to each other so as to select the best ones to use in a branch and bound algorithm. One chapter will be dedicated to each of these topics, i.e. Chapter III to upper bounds, Chapter IV to lower bounds and Chapter V to branch and bound algorithms. In the meantime, this chapter will present some introductory elements necessary to our study of the HFS scheduling problem, and to the solutions we have investigated.

The remainder of this chapter is organized as follows. Section **2** presents a survey of the literature concerning the HFS scheduling problem. A detailed plan of the research carried out in this project is presented in Section **3**. The HFS scheduling problem is best viewed and comprehended using a graph representation. The latter is presented in Section **4.** This graph representation is used later on in Chapter III: Upper Bounds, so as to define the neighborhood of an HFS schedule which in turn is used to design local search techniques. Also, the graph representation shows that the HFS scheduling problem can be relaxed into single stage subproblems. Section **5** presents this relaxation and defines the single stage subproblem, that is the parallel machines scheduling problem with release dates and tails. This relaxation is crucial to the research conducted in this project since most of the results presented here are based upon the single stage subproblem. Section **5**, therefore, also recalls the solutions used to solve the single stage subproblem, that is Jackson's heuristic, Carlier's branch and bound algorithm as well as some lower bounds. We will also show how the single stage lower bounds can be used to compute lower bounds for the K-stage HFS scheduling problem. Section **6** concludes this chapter.

2. SURVEY OF THE HFS SCHEDULING LITERATURE

The HFS shop floor can be seen as the combination of the classical flowshop with one machine at each stage (M_s=1, K>1) and the single stage parallel machine problem (M≥2,K=1) which have been extensively studied. General scheduling studies are found in Baker (1974), GOTHA (1993) and Pinedo (1995). We hereafter give some references for the flowshop and parallel machines. The reader can refer to these references and the references cited therein for an extensive presentation of these two topics.

Flowshop problems: Johnson (1954), Campbell et al., (1970), Krone and Steiglitz (1974), Dannenbring (1977), Townsend (1977), Gelders and Sambandam (1978), King and

Spachis (1980), Adiri and Pohoryles (1982), Nawaz et al., (1983), Grabowski et al., (1983), Proust (1992), Dudek et al., (1992), Hunsucker and Shah (1992), Daniels and Mazzola (1994), Rajendran (1994), Das et al., (1995), Moccelin (1995) and Hisao Ishiuchi et al., (1995), Bansal (1997).

Parallel machine problems: Lawler and Labetoulle (1978), Dogramaci and Surkis (1979), Dogramaci (1984), Federgruen and Groenenvelt (1986), Lenstra et al., (1989), Cheng and Sin (1990), Leisten (1990), Taillard (1990), Echalier (1991), Elmaghraby and Guinet (1992), Guinet et al., (1992), Lash (1993), Guinet (1993), Li and Cheng (1994) and Guinet (1995).

We hereafter quote some recent studies carried out on the two-stage and the multi-stage HFS.

2.1 THE TWO-STAGE HFS

The two-stage HFS ($M_s \geq 1$, K=2) configuration has received much attention from many researchers. Johnson (1954) was the first to propose an exact algorithm for the two-stage flowshop makespan minimization problem with one machine at each stage, i.e. $M_s=1$, s=1, 2, and zero release dates and tails. (See Chapter III, Section 4). His algorithm has largely influenced the study of scheduling solutions for two-stage and K-stage HFS as we will see throughout this survey.

Shen and Chen (1972) were the first to tackle a two-stage HFS with $M_s \geq 2$, s=1,2. They proposed a permutation heuristic to solve the Makespan Minimization Problem when preemption is not allowed (MMP). Buten and Shen (1973) have studied the same problem but with jobs having precedence constraints, which they solved with a modified version of Johnson (1954) algorithm. Gupta (1988) has studied the MMP in a two-stage HFS with $M_1 \geq 2$ and $M_2=1$. He states that the MMP in a two-stage HFS is NP-hard when max (M_1, M_2) > 1. This result is very important because it shows that any MMP in a K-stage HFS is NP-hard, since the K-stage HFS can always be reduced to a two-stage HFS. He proposed a heuristic and lower bounds for the MMP case.

Sriskandarajah and Sethi (1989) have studied the worst and average performance of some heuristics for MMP in a two-stage HFS with $M_1 \geq 2$ and $M_2=1$ and with $M_1 \geq 2$ and $M_2 \geq 2$, their solutions are based on Johnson's rule. Gupta and Tunc (1991) have studied the MMP in a two-stage HFS with $M_1=1$ and $M_2 \geq 2$. They show that this problem is the inverse of the one presented in Gupta (1988) and that is also NP-hard. They proposed different upper and lower bounds, which they used in a branch and bound algorithm. The latter uses the branching scheme proposed by Brah and Hunsucker (1991). (See later in this section and in Chapter V of this thesis). Their heuristics give good results only when the number of jobs is smaller than 9.

Guinet et al., (1992), first, proposed a mixed integer formulation of the K-stage HFS and, then, based on Johnson's rule, proposed a heuristic for the MMP in a two-stage HFS. They compared this heuristic with the Shortest Processing Time (SPT) and the Longest Processing Time (LPT) dispatching rules. They conclude that LPT rule gives good results for the MMP in a two-stage HFS. Lee et al., (1993) studied the MMP in a two-stage HFS

when splitting is allowed at the first stage. Chen and Strusevich (1993) have analyzed the worst case of some heuristics for the MMP in a two-stage with $M_1 \geq 2$ and $M_2 \geq 2$.

Tsubone et al., (1996) have studied the impact of the lot sizing and sequencing decisions on manufacturing performance in a two-stage HFS with $M_1=1$ and $M_2 \geq 2$ where priority rules are used for scheduling. Their numerical tests show that $SPT_{1,2}$ rule, using the sum of processing times at the first and second stage, yields the best performance with regard to the makespan, and LPT rule, using the processing times at first stage only, yields the best performance with regard to the maximum work in process. Gupta et al., (1997) reconsider the MMP in a two-stage HFS with $M_1 \geq 2$ and $M_2=1$. They adapted and improved the lower bounds presented in Gupta and Tunc (1991). They also propose several heuristics, all based on Johnson's algorithm. Kadipasaoglu et al., (1997) studied the financial impact of the scheduling decisions using financial priority rules to static and dynamic (jobs release dates are unknown) two-stage HFS with $M_s \geq 2$, s=1, 2.

Artiba et al., (1998) have studied the MMP in a two-stage HFS with one machine at the first stage and two dedicated machines, i.e. assignments to machines are given, at the second stage. They proposed a dynamic programming, which approximated version (an already time consuming procedure) gives the best performance when compared to other heuristics. Elmaghraby and Soewandi (1998a), (1998b) have studied the MMP in a two stage HFS with identical and uniform machines, respectively. They conclude their papers that for the MMP in a two-stage HFS, the more successful approach so far is the one starting with some variant of Johnson's order. Complexity and approximation results about the flowshop scheduling problem with parallel machines and no-wait in process can be found in Goyal and Sriskandarajah (1988) and Hall and Sriskandarajah (1991).

2.2 THE MULTI-STAGE HFS

In Vignier et al., (1995a) the reader can find a description of state of the art methods for scheduling general hybrid flowshop problems. Hunsucker et al., (1989) have performed a simulation study in order to evaluate the performance of several priority rules for the mean flow time and minimum makespan criteria. Hunsucker and Shah (1994) have done a comparative performance analysis of priority rules in constrained multiprocessor flowshop, i.e. with a maximum number of jobs in the system. Like in Hunsucker et al., (1989), the tested priority rules are FIFO, LIFO, SPT, LPT, Most Work Remaining (MWR) and Least Work Remaining (LWR). They concluded that SPT and LPT rules yielded superior performance for the makespan criterion.

In order to find the minimum makespan schedule for the K-stage HFS, Brah and Hunsucker (1991) have introduced a branch and bound algorithm. They have adapted the "*Enumeration Method"* proposed by Bratley et al., (1975) for scheduling the single stage parallel machine problem. Their branching scheme consists of enumerating all job sequences over all the machines throughout the whole shop floor, (see Chapter V, Section 2 for a detailed description of this branching scheme). Their algorithm is very fast for very small instances but already time consuming for small ones. They solved a 6 jobs configuration with 5 stages, all stages are made up of two machines except the middle stage which was made up of 3 machines, in 12 hours running time on a PC XT computer.

Vignier et al., (1996b) have proposed the same branching scheme for the problem of minimizing the makespan among the no-late schedules (where each job meets its due date) but no numerical tests have been carried out. Vignier et al., (1996c) have also used the same branching scheme for minimizing the sum of completion times over jobs.

Portmann et al., (1996) have used a Genetic Algorithm (GA) heuristic for the minimum makespan problem. The GA was used to compute upper bounds at each node of a branch and bound algorithm using Brah's branching scheme. They tested the algorithm on different configurations, each of which has three instances. The configuration are with 5, 10 and 15 jobs in a 2, 3 and 5 stages, most of which are bottleneck configurations, i.e. at some stages the number of machines has been decreased by 1 or 2, compared to other stages, while maintaining, at all stages, the same processing time distribution. All tests were stopped after 2 hours running time on 486/33MHz PC. Only small instances (i.e. 5 jobs) have been solved optimally.

Guinet (1996a) has used a primal dual heuristic to schedule the HFS with either identical or different parallel machines. In the case of an HFS with identical parallel machines, the heuristic produces a deviation from a lower bound of 18%. Guinet and Solomon (1996b) were the first to use the classical flowshop (FS) heuristics to schedule the HFS. They have adapted different classical flowshop heuristics to schedule the HFS. Like in Portmann et al., (1996) the majority (90%) of tested configurations have bottleneck stages. They conclude that, from all the FS adapted heuristics, (Campbell, Dudek and Smith, (1970) (CDS), Nawaz et al., (1983) and Townsend, (1977)), CDS outperforms all the other heuristics on all tested HFS configurations (see a detailed description of this heuristic in Chapter III, Section 4).

2.3 LOWER BOUNDS

Different lower bounds have been developed for the two-stage HFS, see Gupta et al., (1997), and Elmaghraby and Soewandi (1998a), (1998b). Global lower bounds for the K-stage HFS can be found in Santos et al., (1995). Vandevelde et al., (1995) summarized the existing lower bound for the K-stage HFS and introduced new ones. In Chapter IV we recall all these lower bounds which are all based upon the relaxation of the HFS into a single stage subproblem with release dates and tails (see Section 5). This problem has been studied by Carlier (1987). Carlier's solutions to this problem are recalled in Section 5. A lower bound for the single stage subproblem is the preemptive bound, which is the optimal solution of the preemptive case. This solution can be obtained in polynomial time, but the computation complexity is too high for a use in a branch and bound framework, see Labetoulle et al., (1984) and Vandevelde et al., (1995).

2.4 COMPLEXITY

Hoogeveen et al., (1996) have shown that preemptive scheduling in a two-stage flowshop with at least two identical parallel machines in one of the stages so as to minimize makespan is NP-Hard. Their proof shows that decision version of preemptive scheduling in a two-stage HFS with two machines at the first stage and one machine at the second stage is equivalent to a *Partition* decision problem, which is known to be NP-Complete. We hereafter recall their proof.

Partition, P(V, b): Given an integer b and a set $V=\{a_1, \ldots, a_n\}$ of n integers with $\sum_{j=1}^{n} a_j = 2b$, is it possible to partition V into two disjoint subsets of equal sum b? Example: let $V=\{1,2,3,4\}$; $\sum_{j=1}^{n} a_j = 2*5 = 10$ and the two subsets are $V1=\{1,4\}$ and $V2=\{2,3\}$.

Two-stage HFS Decision problem (HFS-D), HFSD(I, p, δ): Given a two-stage HFS with two identical machines at the first stage and one machine at the second stage where the preemption is allowed, given a set of jobs $I=\{1, \ldots, k\}$ with respective processing times p_{1j} and p_{2j}, $j \in I$, is it possible to schedule the set of jobs I within the time limit δ?

The proof in Hoogeveen et al., (1996) consists of providing a polynomial transformation of a *Partition* instance into an *HFS-D* instance and showing that these two decision problems (*Partition and HFS-D*) are equivalent.

Proof. Given an instance of Partition, they construct the following instance of a two-stage HFS minimization problem *when preemption* is allowed, with two machines at stage 1 and one machine at stage 2. They choose any number $\alpha > 1$. For each integer $j=1,\ldots, n$ they define a partition job $J_j = (O_{1j}, O_{2j})$ with $p_{1j}=\alpha a_j$ and $p_{2j}=a_j$. In addition, they define four *separation jobs*, whose purpose is to create time slots for the execution of the partition jobs; their processing times are given in Table 2.1. Note that no schedule can be shorter than $\delta = 2(\alpha b+b)$ because the total processing time at stage 2 is at least equal to $2\alpha b + 2b$, i.e. sum of processing times of operations $O_{2,n+1}$, $O_{2,n+2}$ and $O_{2,1}, \ldots, O_{2,n}$ with $\sum_{j=1}^{n} p_{2j} = 2b$; and, because the total processing time at stage 1 is equal to $(2\alpha b + 4b + 2\alpha b)$. And, that a schedule of that length, if it exists, cannot contain idle time.

Case 1: a yes-instance of P(V, b) defines a yes-instance of HFSD(I, p, δ). With a subset $S \subset V$ with $\sum_{j \in S} a_j = b$, they construct a schedule of length $2(\alpha b+b)$ as depicted in Figure 2.2. Let M_1 and M_2 denote the first-stage machines, and M_3 be the second-stage machine. The processing time of the first-stage operations corresponding to the elements of S starts at time 0 on M_1; they are executed consecutively during a period of αb. The processing of the corresponding second-stage operations starts at time αb on M_3 and is preceded by the execution of $O_{2,n+1}$. Operation $O_{1,n+2}$ starts at time 0 on M_2 and is executed without interruption; operation $O_{2,n+2}$ is processed without delay on M_3. The execution of the remaining partition jobs (N-S) starts at time $\alpha b+b$ on M_2 and at time $2\alpha b+b$ on M_3. Finally, the operations $O_{1,n+3}$ and $O_{1,n+4}$ are scheduled on M1 and M2, respectively, to complete at time $2(\alpha b+b)$. Hence, a partition $S \subset V$ with $\sum_{j \in S} a_j = b$ certifies that we have a yes-instance of HFSD(I, p, δ).

Case 2: Given a yes-instance of HFSD(I, p, δ) then P(V, b) is yes-instance of Partition. Conversely, suppose that a *preemptive* schedule σ of length $2(\alpha b + b)$ exists. So, without loss of generality, given any preemptive schedule, it is easy to obtain a schedule of the same length where the second stage operations are executed without preemption in order of the completion times of the corresponding first-stage operations. Since σ contains no machine idle time and J_{n+1} is the only job with first-

stage processing time equal to 0, operation $O_{2,n+1}$ complete at time αb. For similar reasons, operations $O_{1,n+3}$ and $O_{1,n+4}$ complete at time $2(\alpha b + b)$.

Jj	J_{n+1}	J_{n+2}	J_{n+3}	J_{n+4}
p_{1j}	*0*	$\alpha b + b$	$\alpha b + 2b$	*b*
p_{2j}	αb	αb	*0*	*0*

Table 2.1: Processing times of the separation jobs

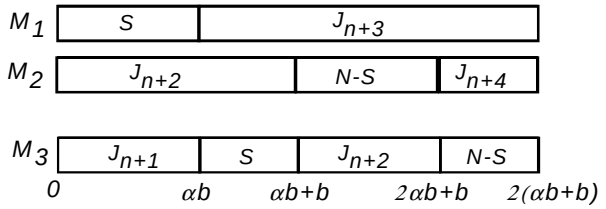

Figure 2.2: A schedule with partition sets S and N-S

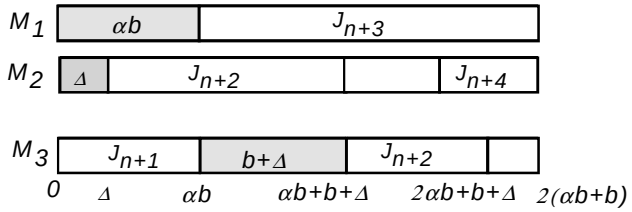

Figure 2.3: A schedule with partition set jobs, when $O_{1,n+2}$ completes at time $\alpha b + b + \Delta$

Let $O_{1,n+2}$ complete at time $\alpha b + b + \Delta$, with $\Delta > 0$, (see illustration in Figure 2.3). This Δ may correspond to some processing time fractional part because preemption is still allowed at stage 1. Machine M_3 has to perform partition jobs from time αb to time $\alpha b + b + \Delta$, since J_{n+2} finishes completion time at $\alpha b + b + \Delta$. Their first-stage operations must be performed by M_1 and M_2 before J_{n+3} and J_{n+2} start. The execution of these operations takes at least time $\alpha(b + \Delta)$, and the amount of time available is $\alpha b+\Delta$ (αb on M1 and Δ on M2). From $\alpha(b+\Delta) \leq \alpha b + \Delta$ and $\alpha > 1$, it follows that $\Delta=0$ so that the total processing time of the partition jobs in question is equal to αb on M_1 and to b on M_3. So, on machine M_3, the schedule σ is not preemptive and partition jobs at stage 2 are scheduled from αb to $\alpha b+b$ and from $2\alpha b+b$ to $2(\alpha b+b)$. Hence, a schedule of length $2(\alpha b + b)$ certifies that we have a yes-instance of Partition.

The K-stage HFS can also be relaxed into the single stage with jobs having release dates and tails (see Section 5) which is an NP-hard problem (Carlier 1987). Actually, even the one machine problem with release dates and tails is NP-hard, see Bratley et al., (1971) and Lenstra (1976). We therefore conclude that our general problem is NP-Hard.

2.5 PROPOSED RESEARCH

According to this survey, one can observe that the K-stage HFS scheduling problem is not well solved, in the sense that good solutions do not exist yet for problems with industrial size. Also, the proposed solutions are mainly adaptations of existing classical flowshop and job shop scheduling solutions. Furthermore, the HFS solutions found in the literature are disparate and some one interested in finding a good solution cannot easily choose the best heuristic among the existing ones without having to compare them all.

Indeed, no comparative study exists. We therefore propose to study some of the existing heuristics and introduce new ones as well as conduct a comparative study so that one can choose the best set of heuristics for the right HFS configuration.

To the best of our knowledge, the only existing exact method for solving the general K-stage HFS is the one proposed by Brah and Hunsucker (1991). Their branching scheme consists in enumerating all job sequences over all the machines throughout the whole shop floor. The sequence enumeration technique has been largely used for general scheduling algorithms. However, its computational complexity is very high, when used in the HFS context. Indeed, in Brah and Hunsucker (1991), Vignier et al. (1996c) and Portmann et al., (1996) only very small (5 to 6 jobs) instances are solved optimally in reasonable time. We therefore propose to study the branch and bound technique, so as to improve the sequence enumeration and to introduce a new branching scheme. The development of a fast branch and bound algorithm requires good and easily obtained lower bounds, which we also propose to study.

The next section gives an overview of the research carried out in this project. We present the research orientation pursued as well as the specific development that will be made in each of the subsequent chapters, corresponding to the three topics: upper bounds, lower bounds as well as branch and bound algorithms.

3. RESEARCH OUTLINE

3.1 UPPER BOUNDS

Several approaches that schedule the HFS one stage at a time can be designed so as to build an HFS schedule. One, for instance, can schedule the HFS using different stage sequencing orders, starting from the first stage and going forward to the last one, from last to first or by first scheduling the stage having a higher priority among the not yet scheduled ones. At each stage, building a schedule requires assignment and sequencing decisions for each job. At each stage a job should be assigned to a specific machine and should have a specific order within the set of jobs assigned to this machine. These two decisions have no priority between them. One can first sequence and assign later on or inversely. List heuristics, for example, using an ordered list of jobs according to some criterion, start by first sequencing the jobs.

One can notice that there are several ways for building a heuristic, by using different stage sequencing orders and different job assignment and sequencing rules. We hereafter propose five different scheduling classes for the HFS problem using five different approaches. In order to build an HFS schedule, one can:

- simply use random scheduling.
- schedule the HFS one stage at a time using different stage sequencing orders. This approach is derived from job shop solutions. See Adams et al., (1988).
- use ideas derived from the classical multistage flowshop with a single machine per stage. This approach has been proposed by Guinet and Solomon (1996b).

- use classical priority rules. This approach is derived from the job shop literature. See also Hunsucker et al., (1989) and Hunsucker and Shah (1994).
- or use randomized local search techniques.

As opposed to the approaches where specific assignment and sequencing rules are used, *random scheduling* uses different random sequencing and assignment rules. The heuristics using random scheduling are fast and easy to implement. They will be used as a basis for comparing all the heuristics presented in this thesis so as to verify whether elaborate heuristics are worth developing or whether any random heuristic should suffice to find a good solution.

Another approach consists of scheduling the HFS one stage at a time, and proposes a global solution for the HFS problem by using *different stage sequencing orders*. Two sets of heuristics are put forward. The first one defines the sequencing order for the stages of the HFS using the Shifting Bottleneck Procedure SBP, which has firstly been used by Adams et al., (1988) to solve the job shop scheduling problem. The second set of heuristics schedules the stages of the HFS sequentially starting at the first (the last) stage and going forward (backward) until the last (first) stage is reached. In order to schedule the single stage subproblem, either a heuristic or an exact algorithm will be used. The single stage subproblem and some of its solutions are presented in Section 5.

The class of heuristics using ideas derived from the classical flowshop and the one using classical priority rules are mainly "*list*" heuristics. They operate in two steps. The first step generates an ordered list of the jobs. The second step schedules the HFS according to these lists. In order to generate the lists, one class uses *ideas derived from the two-stage classical flowshop algorithms* while the other uses *classical priority rules*. Both classes apply different assignment policies in order to assign the machines to the jobs and to come up with an HFS global schedule. The last approach uses *randomized local search techniques*.

All these approaches will be used in Chapter III to develop solutions for the HFS scheduling problem. There, we shall present a number of existing and new heuristics, and classify them according to these approaches. We shall also report on a comparative study of the five classes of heuristics on the same instances. Extensive numerical tests as well as statistical and global analysis of their performances will be carried out so as to select the best set of heuristics for several problem configurations.

3.2 LOWER BOUNDS

Lower bounds are needed to evaluate the performance of the heuristics as well as to develop branch and bound algorithms. The lower bounds we develop are all based upon the relaxation of the HFS main problem into single stage subproblems with release dates and tails (see Section 5). Many lower bounds do exist for the single stage subproblem. We recall some of them and develop new ones in Chapter IV.

A lower bound on the single stage subproblem is the Preemptive Bound (PB), which is the optimal solution of the preemptive case. See Vandevelde et al., (1995) and Chapter III. PB consists in building a *preemptive schedule* where the makespan is minimized for the single stage subproblem. This solution can be obtained in polynomial time, but the

computation complexity is too high for use in a branch and bound framework. See Labetoulle et al., (1984) and Vandevelde et al. (1995). This bound is computed in $O(p_{max}n^3)$ where n is the number of jobs and p_{max} is the largest processing time. A less time consuming but less tight lower bound is the subset bound (SSB). Vandevelde et al., (1995) have shown that this lower bound can be computed in $O(n^2)$ without having to enumerate all subsets.

In order to develop an effective branch and bound algorithm, tight and computationally fast lower bounds are needed. We introduce a heuristic for the subset bound computed in O(nm) where n is the number of jobs and m is the number of machines. Two new tighter lower bounds will be introduced. The first one is a dual heuristic derived from a time-indexed formulation for the makespan minimization problem. The second lower bound is obtained by constructing a partially preemptive schedule. The latter is obtained by building a special time-indexed formulation, which is a transportation problem. As a matter of fact, the time-indexed formulation constrains the preemptive assignment of the jobs. This lower bound is tighter than PB but its running time complexity is higher.

Since some of the lower bounds use the maximum flow problem as a subproblem, a specific section will be dedicated to recalling the "*maximum flow problem"* as well as the "*preflow push***"** algorithm we decided to use to solve it. Ahuja et al., (1997) have conducted an extensive study on a variety of maximum flow algorithms on several classes of networks, and concluded that the "*preflow push"* algorithm outperforms all the other tested algorithms. We also present an improved version of the "*preflow push*" algorithm for bipartite graphs that runs faster than the original algorithm.

Eventually, all known single stage lower bounds are tested on the same instances so as to compare their practical complexities, and to select the ones to use in the designed branch and bound algorithms.

3.3 BRANCH AND BOUND ALGORITHMS

Minimizing the makespan for the HFS scheduling problem is NP-Hard. We study some branch and bound techniques as exact methods. These branch and bound algorithms are used as heuristics when stopped before finding the optimal solution, i.e. truncated after running a certain amount of time. Two branch and bound algorithms are studied in Chapter V.

- The *"Sequence enumeration"* method has largely been used, as a branching scheme, so as to develop branch and bound algorithms for scheduling problems. The "*sequence enumeration"* consists in enumerating all possible sequences of jobs over all machines and through the whole shop floor. Brah and Hunsucker (1991) have proposed a *sequence enumeration* branch and bound algorithm for the HFS scheduling problem. Their algorithm is an adaptation of the branching scheme introduced by Bratley et al., (1975) for the parallel machine problem.
  We recall the *sequence enumeration* algorithm proposed by Brah and Hunsucker (1991) and introduce improvements to the branching scheme as well as to the bounding operations. The original and new algorithms will be compared based on numerical tests to appreciate the new improvements.

- Carlier (1987) has used the "*interval method"* to develop a branching scheme for the single stage parallel machines problem with release dates and tails. Based on some known makespan or upper bound, the "*interval method"* consists of assigning a feasible starting time interval to each job in order to obtain a better solution than the incumbent one. We generalize the *interval method* and design a branch and bound algorithm for the HFS scheduling problem.

The branch and bound algorithm is a very sensitive technique and the time allocated to each part of the algorithm, so as to find a solution, should be wisely used. Indeed, some of this time could be wasted computing non-effective lower bounds, i.e. the effectiveness of discarding nodes or of getting good solutions quickly in the search tree is the same as the one obtained with less time consuming lower bounds. It could also be wasted generating suboptimal nodes that can be discarded with good or better lower or upper bounds. According to the results and the conclusions presented in Chapter III and Chapter IV, we present in Chapter V for use in branch and bound algorithms several upper and lower bounding strategies. Extensive simulation and testing will allow us to fine tune the use of these bounds. Eventually, the two algorithms, based on *the sequence enumeration* and the *interval method*, are tested on the same instances and compared.

3.4 OBJECT ORIENTED IMPLEMENTATION

Extensive study and design of several scheduling algorithms have led us to implement a large variety of scheduling algorithms. For efficiency's sake and in order to reduce the complexity and time needed for computer implementation, object oriented modeling has been used so as to design a global Object Model for Scheduling Algorithms Implementation (OMSAI). This model is presented in the appendix of this thesis.

In the next section, we present the graph representation of the HFS scheduling problem. The latter is essential to better understand and design scheduling heuristics. It is used in Chapter III to define the neighborhood of an HFS schedule which in turn is used to design local search techniques.

4. GRAPH REPRESENTATION OF THE HFS SCHEDULING PROBLEM

GRAPH REPRESENTATION

Adams et al., (1988) have used a graph representation for the jobshop scheduling problem. The hybrid flowshop scheduling problem can also be depicted by a directed graph formed by a set of nodes and two sets of arcs, i.e. conjunctive and disjunctive arcs.

- *The nodes* represent the operations of the jobs, one node (i, s) for each operation s of each job i, $i \in I$, s=1, ..., K.
- *The conjunctive arcs* represent the precedence constraints between the successive operations (i,s-1) and (i,s) of each job i for s=2, …, K. (operations overlapping is not permitted).
- *The disjunctive arcs* represent the sequencing of the jobs on the different machines. For all (i, s), $i \in I$, s=1, …, K, there is a disjunctive arc linking node (i, s) to any node (j, s) representing an operation at the same stage s. These arcs are referred to as

disjunctive because their existence and orientation have to be decided in the solution process to represent precedence or sequencing constraints. The arc liking node (i, s) and node (j, s) exists or appears on the graph only when operation i is followed by operation j on the same machine. Its orientation defines the precedence relation between (i, s) and (j, s).

- *The weight or length of any arc* coming from node (i, s) to node (j, t) represents the processing time p_{is} of the operation s of job i, represented by node (i, s).
- *Nodes (0) and (f)*, respectively, represent the first and the final fictitious operations of the global schedule.

For each job i, the fixed release date r_{i1} of operation (i, 1) is represented by a conjunctive arc coming from node (0) to node (i, 1) with weight or length r_{i1}. For each job i, the node (i, K) is linked to the final node (f) by a conjunctive arc with weight $(p_{iK} + q_{iK})$ in order to take into account both the processing time of job i and its final tail at stage K.

DEFINING A SCHEDULE

Before scheduling, the graph contains the conjunctive arcs only, i.e. the precedence constraints between the operations of each job. The disjunctive arcs with specific orientations will be added to the graph according to a chosen schedule. Thus, for two different solutions or schedules, in terms of the assignment of jobs to machines and their sequencing, we obtain two different graphs in terms of chosen disjunctive arcs and their orientations.

In order to schedule the shop we have to assign machines to operations and to sequence the operations on the machines, at each stage. These sequences can be represented in the graph by paths of disjunctive arcs, at each stage. The number of paths at any stage is equal to the number of available machines at the corresponding stage. Indeed, a path corresponds to a sequence of jobs on one machine and the nodes belonging to one path correspond to the jobs assigned to one machine. A node (i, s), $i \in I$, s=1, …, K, in the graph representation of a schedule, has to be connected to exactly one such path of disjunctive arcs, i.e. assigned to one machine at each stage s.

The resulting graph composed of the conjunctive arcs and the selected disjunctive arcs corresponds to a feasible schedule only if the graph is acyclic. The longest path from node (0) to node (f), i.e. global path, gives the length or makespan of the schedule. Our task then consists in minimizing the length of this longest path.

EXAMPLE

We hereafter give an example of a three-stage HFS scheduling problem where each stage is made up of two identical parallel machines, K = 3 and $M_s = 2$ for s=1, 2, 3. A set of jobs I (n= |I| =5) needs to be scheduled in the three-stage HFS. The jobs have non-zero initial release dates and final tails, ($r_{i1} \geq 0$; $q_{iK} \geq 0$ $\forall$ i=1, ..., 5). Table 2.2 presents the problem data.

i	r_{j1}	p_{j1}	p_{j2}	p_{j3}	q_{j3}
1	0	5	7	3	10
2	2	8	4	2	7
3	1	5	3	5	9
4	3	3	4	6	5
5	2	4	9	7	12

Table 2.2: A three-stage HFS with two machines at each stage

In Figure 2.4 we schedule the 5 jobs. Each job is made up of 3 operations. For example, job 1 is made up of operations (1,1), (1,2) and (1,3) which are to be processed respectively at stages 1, 2 and 3. Operation (1,3) cannot start before operation (1,2) has been completed, and operation (1,2) cannot start before operation (1,1) has been completed. Each stage is composed of 2 machines. We thus need to draw 2 paths of disjunctive arcs by stage, one for each machine. The chosen paths for this example are:

- Path {(1,1); (2,1)} and path {(5,1); (3,1); (4,1)} for stage 1
- Path {(1,2); (2,2); (3,2)} and path {(5,2); (4,2)} for stage 2
- Path {(2,3); (1,3); (3,3)} and path {(5,3); (4,3)} for stage 3

Each path represents the sequence of jobs on one machine at the corresponding stage. A schedule, whose makespan corresponds to the length of the longest path, is obtained by starting each operation (i, s) after a duration equal to the longest path in the graph from node (0) to node (i, s). The longest path from node (0) to node (f) in this graph is {(0); (1,1); (2,1); (2,2); (2,3); (1,3); (3,3), (f)} and has a length of 36 units of times.

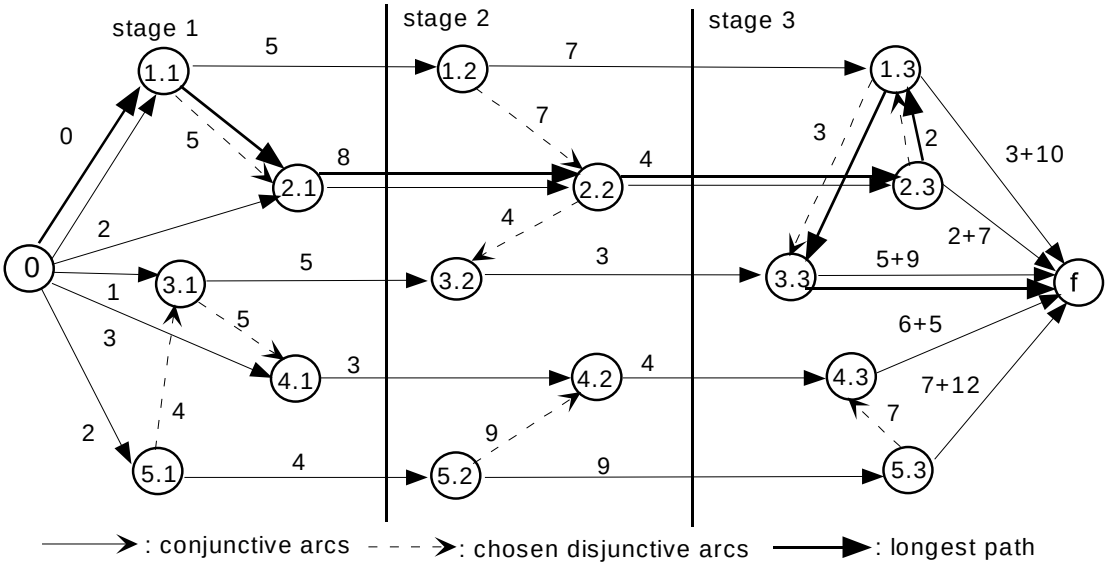

Figure 2.4: A graph representation of a three-stage HFS schedule

5. THE SINGLE STAGE SUBPROBLEM

In this section we relax the HFS scheduling problem into single stage subproblems. Such a relaxation is very important to our research since it is used in lower bound computations as well as in designing heuristics and branch and bound algorithms. In some of the heuristics, we need to solve the single stage subproblem as a subroutine. The latter is solved either by Jackson's heuristic or by Carlier's branch and bound (1987). To the best of our knowledge and according to Carlier (1987) these algorithms give very good results and are the best existing solutions for the single stage subproblem. Jackson's heuristic and Carlier's branch and bound are then briefly described. Eventually, we present some single stage lower bounds and show how they can be used to compute lower bounds an HFS problem.

5.1 THE RELAXATION

If we relax the capacity at all but one stage, i.e. we have as many machines as jobs except at one stage s we are then faced with the problem of scheduling a set of jobs I, with release dates r_{is} and tails q_{is}; $i \in I$, on M_s identical parallel machines. This holds because for the other stages $t \neq s$, scheduling is not needed since each machine should have been assigned one job at most. In order to define the corresponding single stage subproblem we need to know when the jobs are going to be released in order to be processed at stage s, i.e. release dates r_{is}. We also need to know, when leaving stage s, how long each job will spend in the shop until the completion time of the last operation processed at the last stage, i.e. tails q_{is}. See Adams et al., (1988) and Section 1 in this chapter for the release date and tail definitions.

Note that at each stage the release dates and tails of the jobs can only be known when a schedule is fixed. Because here the capacities are relaxed except at stage s, we can set the release date r_{is} and tail q_{is} of each job i at stage s to their respective lower bounds which are equal to the sum of the processing times of job i in the preceding and subsequent stages, respectively. For each stage $s \in \{1, \ldots, K\}$, we can define a single stage relaxation whose corresponding release dates and tails are computed as follows:

$$r_{is} = r_{i1} + \sum_{t=1}^{s-1} p_{it} \quad ; \qquad q_{is} = \sum_{t=s+1}^{K} p_{it} + q_{iK} \qquad i \in I \qquad (2.1)$$

Adding r_{i1} and q_{iK} respectively to the computation of the release date and tail of job i means that our model can cope with jobs having release dates at the first stage and tails at the last stage.

This model and the methods developed in this project can easily be generalized to the case where jobs have to satisfy minimum waiting times in-between stages by adapting the computation of release dates and tails. Let w_{is} be the minimum time a job has to wait between stage s-1 and stage s before starting its processing at stage s. The release dates and tails, for each relaxed single stage subproblem $s \in \{1, \ldots, K\}$, are then computed as follows:

$$r_{is} = r_{i1} + p_{i1} + \sum_{t=2}^{s-1} (w_{it} + p_{it}) + w_{is} \quad i \in I$$

$$q_{is} = \sum_{t=s+1}^{K} (w_{it} + p_{it}) + q_{iK} \qquad i \in I$$

As far as the heuristics presented in this project are concerned, the scheduling of the HFS is done one stage at a time. When the processing of a job has been completed at stage s-1, we only need to wait w_{is} units of time before releasing that job at stage s. Hence, when scheduling, the release date r_{is} of job i at stage s must satisfy $r_{is} \geq c_{is-1} + w_{is}$ where c_{is-1} is the completion time of job i at stage s-1.

As it has already been mentioned, the model and heuristics tested in this project tackle only problems with initial release dates, final tails and zero waiting time in-between stages.

MINIMIZING THE MAKESPAN OF THE SINGLE STAGE SUBPROBLEM

We need to build a schedule for the *single stage s subproblem* with a minimum makespan, where the makespan δ defined by $\delta = \underset{i \in I}{\text{Max}} [t_{is} + p_{is} + q_{is} - 1]$ where $t_{is} > r_{is}$ is the starting time of job i at stage s, which is thus finished at time $t_{is}+p_{is}-1$. A schedule consists of assigning a specific machine (among those at stage s) to each operation s of every job as well as sequencing all operations assigned to each machine (at stage s) so that each machine processes at most one job at a time and so that job preemption is not allowed.

In the remainder of this thesis we refer to the problem of assigning jobs in set I to machines, and of sequencing their operations at stage s while taking into account operations release dates and tails at stage s, as the *single stage subproblem*. Minimizing the makespan of the single stage s subproblem (i.e. r_{is}, p_{is}, q_{is}, $i \in I$ and M_s identical machines, for s fixed) in the non-preemption case is NP-Hard (Carlier (1987)). In order to solve this problem one can either use Jackson's heuristic or Carlier's branch and bound algorithm. We hereafter give a brief description of these methods, which are used as subroutines in some of the algorithms developed in this project.

In the remainder of this section all our notations refer to the single stage s subproblem. Hence, in our notation we will ignore the subscript s and assume that these data are always related to stage s subproblem.

5.2 JACKSON'S HEURISTIC

Jackson's heuristic (Carlier (1987)) starts from time 1 and schedules the machines by moving forward in time. Each time a machine becomes available, the job with the largest tail, among the already released ones, starts. Figure 2.5 depicts a schedule constructed using Jackson's heuristic.

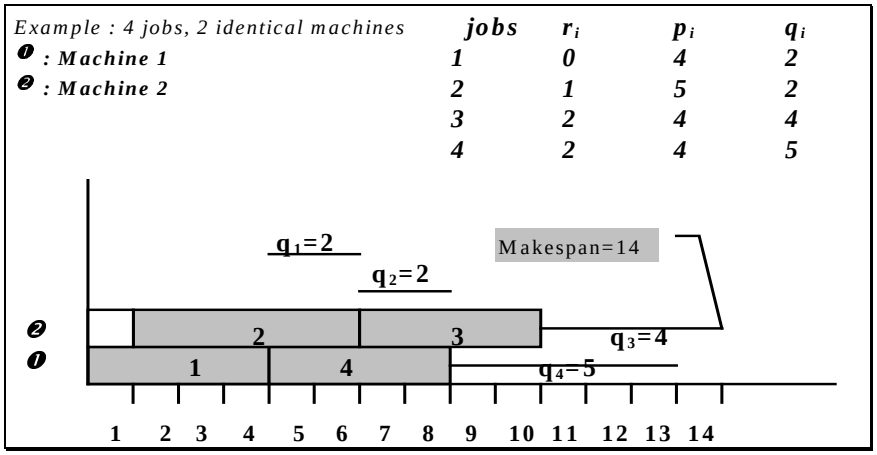

Figure 2.5: A single stage schedule using Jackson's heuristic

5.3 CARLIER'S BRANCH AND BOUND

Carlier proposes a branch and bound algorithm to cope with the single stage subproblem. His algorithm uses Jackson's heuristic at each branching node so as to build a feasible

schedule, and computes lower bounds to decide whether to eliminate the node (the branching node) or continue branching. The best schedule found is maintained so that the search can be stopped at any time with the best-found solution. We give a brief description of this algorithm. Its complete description can be found in Carlier (1987).

NODE REPRESENTATION

For node N of the branch and bound tree and each job j, let $r_j(N)$ and $q_j(N)$, respectively, be the release date and tail of job j at node N. Let Best_UB be the best known makespan so far. Notice that since we already have an upper bound on the solution, in this branching scheme we are only interested in solutions strictly smaller than Best_UB. We can therefore assign to each job j a due date $d_j(N)$ = Best_UB - $q_j(N)$ – 1. Note that d_j is not the usual due date constraint in scheduling but it is a due date that has to be met in order to obtain a better solution value than Best_UB. Indeed, in order to meet the makespan (Best_UB –1), job j can only start processing within the time interval [$r_j(N)$ + 1, $d_j(N) – p_j + 1$]. Let $margin_j = d_j(N) - r_j(N) - p_j$ be the so called margin of job j where $margin_j + 1$ represents the remaining flexibility in terms of possible number of start dates for job j at node N.

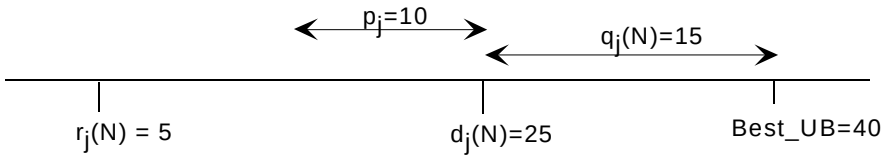

Figure 2.6: (a) The branching job j data at node N

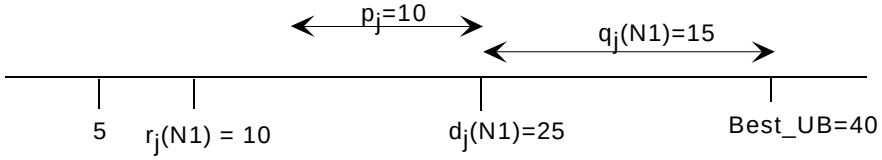

Figure 2.6: (b) The branching job j data at node N1

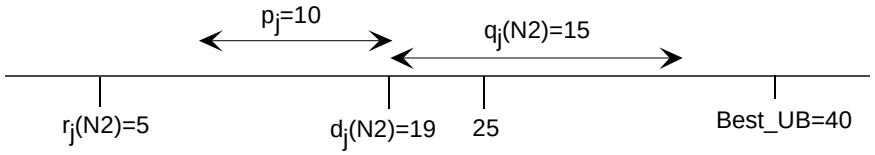

Figure 2.6: (c) The branching job j data at node N2

BRANCHING OPERATION

At node N, when branching on a given job j, Carlier creates two new subproblems or nodes N1 and N2. The data (p_i, r_i, q_i, $i \in I$) for both nodes N1 and N2 remain the same as for node N, except for the release date r_j and the due date d_j of the branching job j, which are set as follows:

For N1, $r_j(N1) = r_j(N) + \lceil margin_j / 2 \rceil$ and $d_j(N1) = d_j(N)$

For N2, $r_j(N2) = r_j(N)$ and $d_j(N2) = d_j(N) - \lfloor margin_j / 2 \rfloor - 1$

In other words, the starting time interval of job j (i.e. $[r_j(N) + 1, d_j(N) – p_j + 1]$) is split into two non-overlapping parts:

At node N1, job j can start in the interval $[r_j(N) + \lceil margin_j /2 \rceil + 1, d_j(N) - p_j + 1]$
At node N2 job j can start in the interval $[r_j(N) + 1, d_j(N) - \lfloor margin_j /2 \rfloor - p_j]$.

Figure 2.6(a) shows the data of the branching job j at node N. Figure 2.6(b) and Figure 2.6(c) show how j is modified to generate node N1 and node N2, respectively. The starting time interval of job j at node N is [6, 16]. The two non-overlapping starting time intervals of job j at node N1 and N2 are [11, 16] and [6, 10], respectively.

For branching, Carlier chooses the node N with the best or smallest lower bound LB(N). He computes an upper bound UB(N) using Jackson's heuristic. If the node needs to be explored (i.e. LB(N) < Best_UB(N)) he chooses the branching job j which maximizes the sum $Z_j = LB(N1^j) + LB(N2^j)$ where $LB(N1^j)$ and $LB(N2^j)$ are the lower bounds obtained at node N1 and N2, respectively, when branching on job j.

5.4 LOWER BOUNDS FOR THE SINGLE STAGE SUBPROBLEM

A set of lower bounds of various quality and complexity regarding the problem of minimizing the makespan has been proposed in the literature. In Vandevelde et al., (1995) the interested reader can find an extensive development of these bounds which we summarize in Chapter IV. We hereafter list some of the single stage subproblem lower bounds that will be used in the computation of the HFS lower bounds the Job Bound JB, the Set Bound SB and the Subset Bound SSB.

$$JB(I) = \max_{i \in I} (r_i + p_i + q_i) \qquad (2.2)$$

$$SB(I) = \lceil (r_{[1]} + r_{[2]} + \ldots + r_{[M]} + q_{[1]} + q_{[2]} + \ldots + q_{[M]} + \sum_{i \in I} p_i) / M \rceil \qquad (2.3)$$

where $r_{[1]}, r_{[2]}, ..., r_{[M]}$ are the M smallest r_i over $i \in I$
$q_{[1]}, q_{[2]}, ..., q_{[M]}$ are the M smallest q_i over $i \in I$

$$SSB(I) = \max_{V \subset I, |V| > M} SB(V) \qquad (2.4)$$

The job bound (2.2), of complexity *O(n),* relaxes the number of machines by assuming that the number of machines is as large as the number of jobs. In such a relaxed problem the makespan is given by $\max_{j \in I} (r_j + p_j + q_j)$.

The set bound (2.3) computed in *O(n),* comes from the observation that the makespan must contain at least the M smallest release dates and tails as well as the total processing time at stage s. At best, this time can be evenly distributed among the M parallel machines. SSB (2.4) relaxes the problem by considering only a subset of jobs and uses SB for the computation of the lower bound. Vandevelde et al., (1995) have shown that SSB can be calculated in $O(n^2)$ without enumerating all subsets of I.

5.5 COMPUTING LOWER BOUNDS FOR THE HFS

A global lower bound HLB on the HFS makespan is the maximum optimal Makespan over all Single Stage subproblems s=1, …, K, say $MSS^*(s)$. Since the complexity of finding the optimal makespan of the single stage subproblem can be very high, any lower bound LB(s) on that makespan is also a lower bound on the makespan of the HFS. A lower bound on the makespan of the HFS scheduling problem can then be computed as follows:

$$HLB1 = \max_{s\in\{1,\ldots,K\}} MSS^*(s) \qquad (2.5)$$

$$HLB2 = \max_{s\in\{1,\ldots,K\}} LB(s) \qquad (2.6)$$

In Chapter IV lower bounds for the HFS scheduling problem are studied in details.

6. CONCLUSION

In this chapter, we have introduced the hybrid flowshop scheduling problem we are about to study in the subsequent chapters. Based on the literature survey, we presented a detailed plan of the scheduling research we decided to pursue in the project. We study branch and bound algorithms through the development of upper and lower bounds as well as branching schemes. As far as upper bounds are concerned, we defined five classes of heuristics using different approaches to come up with an HFS schedule. We propose to develop tighter lower bounds and improve existing ones. We also suggest to study branch and bound algorithms by improving the existing *sequence enumeration* branching scheme as well as by designing a new one based on the so-called *interval method*.

Some introductory elements necessary to the algorithms developed in this project have also been presented. The graph representation of an HFS schedule, used for designing scheduling algorithms, is introduced. The HFS problem can be relaxed into single stage subproblems. In the same way such a relaxation is used in lower bound computations and for designing heuristic solutions as well as branch and bound algorithms. We recalled Jackson's heuristic and Carlier's branch and bound we had decided to use for solving single stage subproblems. Eventually, we presented some single stage lower bounds and showed how these bounds can be used to compute lower bounds on the HFS problem.

Based on the five classes of heuristics presented in this chapter the following chapter aims at presenting heuristic solutions. The main objective is to carry out a comparative study for the sake of selecting the best set of heuristics for each HFS configuration. This study will also contribute to the design of effective branch and bound algorithms.

REFERENCES

1. Adams J., E. Balas, and D. Zawack, (1988). The Shifting Bottleneck Procedure, Management Science, vol. 34, 391-401, n°3, March 1988.

2. Adiri I., and D. Pohoryles, (1982). Flowshop/no-idle or no-wait scheduling to minimize the sum of completion times, Naval Research Logistics Quarterly, vol. 29, 3, 495-504, 1982.
3. Ahuja R. K., M. Kodialam, A. K. Mishra, and J. B. Orlin, (1997). Computational investigations of maximum flow algorithms, European Journal of Operational Research, 97, 509-542.
4. Artiba A., S.E. Elmaghraby and F. Riane, (1998). Sequencing Hybrid two-stage flow shops with Dedicated machines, Technical Report, North Carolina University, Raleigh, North Carolina, USA, FUCAM, MONS, Belgium, 1998.
5. Baker K.B., (1974). Introduction to Sequencing and Scheduling, John Wiley & Sons Inc, New York, 1974.
6. Bansal S.P., (1997). Minimizing the sum of completion times of n jobs over m machines in a flowshop – a branch and bound approach, AIIE Transactions, vol. 9, 3, 306-311, 1997.
7. Brah S. A., and J.L. Hunsucker, (1991). Branch and bound algorithm for the flow shop with multiprocessors, European Journal of Operational Research, 51, 88-89, 1991.
8. Bratley P., M. Florian and P. Robillard, (1971). On sequencing with earliest starts and due dates with applications to computing bounds for the n / m / G / F_{max} problem, Naval Research Logistics Quarterly, 20, 57-67, 1971.
9. Bratley P., M. Florian and P. Robillard, (1975). Scheduling with earliest start and due date constraints on multiple machines, Naval Research Logistics Quaterly, vol. 22, n°1, 165-173, 1975.
10. Buten R.E. and V.Y. Shen, (1973). A scheduling model for computer systems with two classes of processors, Proceedings of the "1973 Sagamore computer conference on parallel processing", 130-138.
11. Campbell H.G., R.A. Dudek and M.L. Smith, (1970). A heuristic algorithm for the n job, m machine sequencing problem, Management Science, Vol. 16, n°10, 630-637.
12. Carlier J., (1987). Scheduling jobs with release dates and tails on identical machines to minimize the Makespan, European Journal of Operations Research, 29, 298-306.
13. Chen Bo and Vitaly A. Strusevich, (1993). Worst-case analysis of heuristics for open shops with parallel machines, European Journal of Operational Research, 70, 379-390, 1993.
14. Chen Bo, (1995). Analysis of Classes of Heuristics for Scheduling a Two-Stage Flow Shop with Parallel Machines at One Stage, Journal of the Operational Society, 45, 234-244.
15. Cheng T.C.E. and C.C.S. Sin, (1990). A state-of-art review of parallel-machine scheduling research, European Journal of Operational Research, 47, 271-292, 1990.
16. Daniels R.L. and J.B. Mazzola, (1994). Flow shop scheduling with resource flexibility, Operations Research, vol. 42, 11, 1174-1182, 1994.
17. Dannenbring D. G., (1977). An evaluation of the flow shop sequencing heuristics, Management Science, 23/11, 1174-1182.
18. Das S.R., J.N.D. GUPTA and B.M. Khumawala, (1995) A savings index heuristic algorithm for flowshop scheduling with sequence dependent setup times, Journal of the Operational Research Society, vol. 46, 1365-1373, 1995.
19. Dogramaci A. and J. Surkis, (1979). Evaluation of a heuristic for scheduling independent jobs on parallel identical processors, Management Science, vol. 25, 12, 1208-1216, 1979.

20.Dogramaci A., (1984). Production scheduling of independent jobs on parallel identical processors, International Journal of Production Research, vol. 22, 4, 535-548, 1984.
21.Dudek R. A., S. S. Panwalkar, and M. L. Smith, (1992). The lessons of flowshop scheduling research, Operations Research, vol. 40, 1, 7-13, 1992.
22.Echalier F., (1991). Problèmes d'ordonnancement sur machines parallèles: apport du recuit simulé, Thèse de Doctorat, Université Claude Bernard –Lyon I, 1991.
23.Elmaghraby S.E. and A. Guinet, (1992). Scheduling on parallel machines: an alternate approach, Third International Workshop on Project Management and Scheduling Como, Italy, 1992.
24.Elmaghraby S.E. and H. Soewandi, (1998a). Sequencing Jobs on Two-Stage Hybrid Flowshop with Uniform Machines to Minimize makespan, Department of Industrial Engineering, North Carolina State University, Raleigh, NC, USA (1998a).
25.Elmaghraby S.E. and H. Soewandi, (1998b). Sequencing Jobs on Two-Stage Hybrid Flowshop with Identical Machines to Minimize makespan, Department of Industrial Engineering, North Carolina State University, Raleigh, NC, USA (1998b).
26.Federgruen A. and H. Groenenvelt, (1986). Preemptive scheduling on uniform machines by ordinary network flow techniques, Management Science, Vol. 32, 3, 341-349, 1986.
27.Gelders L.F. and N., Sambandam, (1978). Four simple heuristics for scheduling a flow shop, International Journal of Production Research, vol. 16, 3, 221-231, 1978.
28.GOTHA, (1993). Les problèmes d'ordonnancement / Scheduling problems, R.A.I.R.O. Recherche Opérationnelle / Operations Research, vol. 27, 1, 77-150, 1993.
29.Goyal S. K., and C. Sriskandarajah, (1988). No-wait shop scheduling: Computational complexity and approximation algorithms, Opsearch, 25, 220-244.
30.Grabowski J., E. Skubalska and C. Smutnicki, (1983). On flowshop scheduling with release and due dates to minimize maximum lateness, Journal of the Operational Research Society, vol. 34, 615-620, 1983.
31.Guinet A., F. Echalier and A. Dussauchoy, (1992). Scheduling jobs on parallel machines: a servey, EURO 12, TIMS XXXI, Helsinki, Finland, 1992.
32.Guinet A., (1993). Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria, International Journal of Production Research, vol. 31, 7, 1579-1594, 1993.
33.Guinet A., (1995). Scheduling independent jobs on uniform parallel machines to minimize tardiness criteria, Journal of Intelligent Manufacturing, vol. 6, 95-103, 1995.
34.Guinet A., (1996a). Integrated and Sequential Approaches to schedule hybrid flowshops in make to stock Environment. Workshop on production planning and control (Proceedings), FUCAM (Belgium), 299-302, September 1996.
35.Guinet A., M. Solomon, (1996b). Scheduling hybrid flows hops to minimize maximum tardiness or maximum completion time, Int. J. Prod. Res., 34, 1643-1654.
36.Gupta J.N.D., (1988). Two-stage, hybrid flowshop scheduling problem, J. Op. Res. Soc., 39, 359-364.
37.Gupta J.N.D. and E.A. Tunc, (1991). Schedules for a two-stage hybrid flowshop scheduling with parallel machines at the second stage, Int. J. Prod. Res., 29, 1489-1502.
38.Gupta J.N.D., A.M.A. Hariri, and C. N. Potts, (1997). Scheduling a two-stage hybrid flow shop with parallel machines at the first stage, Annals of Operations Research, Vol. 69, 171-191.

39.Hall N.G. and C. Sriskandarajah, (1991). Machine scheduling problems with no-wait in process, Working Paper N° 91-05, Department of industrial Engineering, University of Toronto, Canada.
40.Hisao Ishiuchi, Shinta Misaki and Hideo Tanaka, (1995). Modified simulated annealing algorithms for the flowshop sequencing problem, European Journal of Operational Research, 81, 388-398, 1995.
41.Hoogeveen J.A., J.K. Lenstra and B. Veltman, (1993). Minimizing makespan in a multiprocessor flowshop is strongly NP-Hard. Unpublished manuscript. (See also B. Veltman, (1993). Multiprocessor Scheduling with Communication Delays. Ph.D. Thesis, CWI, Amsterdam, The Netherlands).
42.Hoogeveen J. A., J. K. Lenstra and B. Veltman, (1996). Preemptive scheduling in a two-stage multiprocessor flowshop is NP-Hard, European Journal of Operations Research, 89(22), 172-175, 1996.
43.Hunsucker J.L., S.A. Brah and D.L. Santos, (1989). Simulation study of a flow shop with multiple processors scheduling, TIMS/ORSA joint National Meeting in New York City, October 16-18.
44.Hunsucker J.L. and J.R. Shah, (1992). Performance of priority rules in a due date flowshop, OMEGA Int. J. of Mgmt. Sci., vol.20, 1, 73-89, 1992.
45.Hunsucker J.L., and J.R. Shah, (1994). Comparative performance analysis of priority rules in a constrained flow shop with multiple processor environment, European Journal of Operational Research, 72, 102-104.
46.Jackson, J. R., (1955). Scheduling a production line to minimize maximum tardiness, Research 43, Management Science Research Project, University of California Los Angeles.
47.Johnson S. M., (1954). Optimal two and three stage production schedules with setup time included, Naval Res. Logist., Q.1, 61-68.
48.Kadipasaoglu S.N., W. Xiang and B.M. Khumawalas, (1997). A comparison of sequencing rules in static and dynamic hybrid flow systems, Int. J. Prod. Res., Vol. 35, n°5, 1359-1384, 1997.
49.King J.R. and A.S. Spachis, (1980). Heuristics for flow shop scheduling, International Journal of Production Research, vol. 18, 3, 345-357, 1980.
50.Krone M.J. and K. Steiglitz, (1974). Heuristic programming solution of a flowshop scheduling problem, Operations Research, vol. 22, 629-638, 1974.
51.Labetoulle J., E.L. Lawler, J. K. Lenstra, and A.H.G. Rinnooy Kan, (1984). Preemptive scheduling of uniform machines subject to release dates, Progress in Combinatorial Optimization, 245-261, Academic Press, Florida.
52.Lash S, (1993). Genetic algorithms for weighted tardiness scheduling on parallel machines, Technical Report 93-01, Northwestern University, Evanston, USA, 1993.
53.Lawler E.L. and J. Labetoulle, (1978). On preemptive scheduling of Unrelated parallel processors, Journal of the Association for computing Machinery, vol. 25, 4, 612-619, 1978.
54.Leisten R., (1990). Flowshop sequencing problems with limited buffer storage, International Journal of Production Research, vol. 28, 11, 2085-2100, 1990.
55.Lenstra J.K., (1976). Sequencing by Enumerative Methods, Mathematisch Centrum, Amsterdam, 1976.
56.Lenstra J.K., D.B. Shmoys and E. Tardos, (1989). Approximation algorithms for scheduling unrelated parallel machines, Mathematics Programming, 1989.

57.Li C.-L., and T.C.E. Cheng, (1994). The parallel machine Min-Max weighted absolute lateness scheduling problem, Naval Research Logistics, vol. 41, 33-46, 1994.
58.Matsuo H., C. J. Suh and R. S. Sillivan, (1988). A controlled search simulated annealing method for the general jobshop scheduling problem, Working paper #03-04-88.
59.Moccelin J.V., (1995). A new heuristic method for the permutation flow shop scheduling problem, Journal of the Operational Research Society, vol. 4, 3, 215-224, 1995.
60.Moursli O. and Y. Pochet; (1996). Heuristics for scheduling a Multiprocessor flowshop, Advances in Industrial Engineering Applications and Practice I. Houston Texas, (USA). Conference proceeding 456-463.
61.Moursli O., (1997a). Branch and bound lower bounds for the hybrid flowshop. Workshop on Intelligent Manufacturing Systems, (IMS'97). Seoul (Korea), Conference proceeding 107-112.
62.Moursli and Pochet (1997b) A branch and bound for the hybrid flowshop. IEMP'97 Lyon France. Conference proceeding 179-188.
63.Nawaz M., Jr. E.E. Enscore and I. Ham, (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, OMEGA 11/1 91-95.
64.Pinedo, M., (1995). Scheduling: Theory, Algorithms, and Systems, Prentice-Hall, Englewood Cliffs, New Jersey 07632, 1995.
65.Portmann M-C, A. Vignier, D. Dardilhac and D. Dezalay, (1996). Some Hybrid flowshop scheduling by crossing branch and bound and genetic algorithms, 5th International Workshop on Project Management and Scheduling (PMS'96), Poznan (Poland), 186-189. To appear in EJOR.
66.Proust C., (1992). Using JOHNSON's Algorithm for solving Flowshop Problems. 12ème Congrès International, IFOR'S 1990.
67.Rajendran C., (1994). A no-wait flowshop scheduling heuristic to minimize makespan, Journal of Operational Research Society, 45, 4, 472-478, 1994.
68.Santos D.L., J.L. Hunsucker, D.E. Deal, (1995). Global lower bounds for flow shops with multiple processors, European Journal of Operational Research, 80, 112-120.
69.Shen V.Y. and Y.E. Chen, (1972). A sheduling strategy for the flow-shop problem in a system with two classes of processors, Proceedings of the Conference on Information and System Science, 645-649.
70.Sriskandarajah C. and S.P. Sethi, (1989). Scheduling algorithms for flexible flowshops: Worst and average case performance, European Journal of Operational Research, 43, 143-160.
71.Taillard E., (1996). Some efficient heuristic methods for the flowshop sequencing problem, European Journal of Operational Research, vol. 47, 65-74, 1990.
72.Townsend W., (1977). Sequencing n jobs on m machines to minimize tardiness: a branch and bound solution, Management Science, vol. 23, 9, 1016-1019, 1977.
73.Tsubone H., M. Ohba and T. Uetake, (1996). The impact of lot sizing and sequencing on manufacturing in a two-stage hybrid flow shop, Int. J. Prod. Res., Vol. 34. No. 11, 3037-3053, 1996.
74.Vandevelde A., H. Hoogeveen, C. Hurkens and J. K. Lenstra, (1995). Lower bounds for the multiprocessor flowshop. Working paper, Eindhoven University of Technology, The Netherlands.
75.Van Laarhoven P. J.M., E.H.L. Aarts and J.K. Lenstra, (1992). Job shop scheduling by simulated annealing, Operations Research, 40, 1992, 113-125.

76. Vignier A., J-C Billaut and C. Proust, (1995a). Les problèmes de type flow shop hybride : état de l'art, in Journée d'études : Affectation et Ordonnancement, CNRS / GdR Automatique/ Pôle SED /GT3, Tours, France, 7-47, 1995. To appear in RAIRO RO/OR. See also Vignier A., Contribution à la résolution des problèmes d'ordonnancement de type monogamme, multimachine ("Flow-shop hybrid"), Ph.D. thesis. Laboratoire d'Informatique (UPRES-EA 2101) de l'Université de Tours. Ecole d'Ingénieurs en Informatique pour l'Industrie (E3i), France, 1997.
77. Vignier A., J-C Billaut and C. Proust, (1996b). Solving the k-stage hybrid flowshop scheduling problems, in CESA96 Computational Engineering in systems applications, Ecole Centrale de Lille (France), 9-12, July 1996. IEEE-SMC.
78. Vignier A., D. Dardilhac, D. Dezalay and C. Proust, (1996c). An optimal approach to solve a k-stage hybrid flowshop, Workshop on production planning and control (Proceedings), FUCAM (Belgium), 315-319, September 1996.