ELSEVIER

European Journal of Operational Research 109 (1998) 96–110

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

Reduction of job-shop problems to flow-shop problems with precedence constraints

Alain Guinet *, Marie Legrand

INSA de Lyon, Laboratoire PRISMa / LISPI, Groupe de Recherches en Productique Informatique des Systèmes Manufacturiers, GPR – Bâtiment 403, 20 Avenue Albert Einstein, 69621 Villeurbanne, France

Received 1 November 1994; accepted 1 March 1997

Abstract

We study the problem of scheduling N independent jobs in a job-shop environment. Each job must be processed on M machines according to individual routes. The objective is to minimize the maximum completion time of the jobs. First, the job-shop problem is reduced to a flow-shop problem with job precedence constraints. Then, a set of flow-shop algorithms are modified to solve it. To evaluate the quality of these heuristics, several lower bounds on the optimal solution have been computed and compared with the heuristic solutions for 3040 problems. The heuristics appear especially promising for job-shop problems with 'flow-like' properties. © 1998 Elsevier Science B.V.

Keywords: Scheduling theory; Optimization; Heuristics; Job-shop; Flow-shop

1. Introduction

In this paper, the problem of scheduling N jobs in a job-shop environment is considered. An approach based on the reduction of job-shop problems to flow-shop problems with precedence constraints between entire jobs, enables us to use polynomial and efficient algorithms. Previous research is briefly reviewed.

1.1. The job-shop problem

The following characteristics define the problem studied. N jobs have to be processed on M machines. The order in which jobs visit machines is arbitrary. A machine can process only one job at a time. The processing of a job on a machine is called an operation. Each job consists of exactly M operations. The jobs can wait between two machines and the intermediate storage is unlimited. No operation preemption is allowed and the jobs are independent. The objective is to minimize the maximum completion time of the jobs, i.e., the makespan.

1.2. Previous works

Job-shop problems are NP hard (Rinnooy Kan, 1976), except for the following cases: with two jobs (Akers, 1956) and with two machines (Jackson, 1956). The proof of NP complexity of the job-shop problem is based on a reduction of this problem to a knapsack problem (Rinnooy Kan, 1976).

The two machine job-shop problems can be solved

* Corresponding author. Fax: +33-72-438538; e-mail: guinet@gprhp.insa-lyon.fr.

in polynomial time with Johnson's rule (Johnson, 1954). The job-shop problems with two jobs can be modelled to a shortest path problem in a graph with forbidden zones (Akers, 1956; Hardgrave and Nemhauser, 1963; Brucker, 1988). For the case with M machines ($M > 2$) enumerative methods have been considered by many authors and heuristics have been designed.

The enumerative methods yield optimal schedules, but are typically limited to cases with no more than 10 jobs and 10 machines. They are computation time intensive. Most of these methods are branch and bound procedures (Balas, 1969; Rinnooy Kan, 1976; Lageweg et al., 1977; Barker and MacMahon, 1985; Carlier and Pinson, 1989; Carlier and Pinson, 1994). Others are based on branch and cutting plane algorithms (Applegate and Cook, 1991), relaxation techniques (Fisher et al., 1983) and mixed integer linear models (Liao and You, 1992). The most efficient enumerative methods appear to be the branch and bound procedures which try to generate a sub-set of the feasible schedules (for example active schedules, see Baker, 1974) which is dominant for the criterion. Carlier and Pinson (1989, 1994) have specified a collection of powerful elimination rules to limit the number of the solutions enumerated.

The heuristic methods are designed differently. They try to optimize the value of the solution found (Storer et al., 1992) or they minimize the computation time needed to calculate a good solution (Baker, 1974, 1984). Some of the most significant works are given below:

- Panwalkar and Iskander (1977), Baker (1974, 1984) compared several job ordering rules which can be used locally for scheduling each machine or widely used for scheduling the entire system. There is no clearly dominant rule for scheduling job-shop systems, however the MWKR (Most Work Remaining) rule is usually more effective than other rules for the makespan criterion while the SPT (Shortest Processing Time) and LWKR (Least Work Remaining) rules are usually more effective than other rules for the mean flow-time criterion (Baker, 1974; French, 1982).
- Charalambous and Hindi (1991) reviewed several expert systems extending the above mentioned heuristics.
- Adams et al. (1988) proposed an approximation algorithm which solves the job-shop scheduling problem dividing it into M single machine scheduling problems which are individually solved with a branch and bound method (Carlier, 1982) based on Schrage's algorithm (Rinnooy Kan, 1976). The branch and bound algorithm takes into account job release date constraints, job tail constraints and minimizes the makespan criterion. After solving the single machine problem, the result with release date and tail constraints is propagated to each previously scheduled machine and they are re-optimized. This process is repeated until all the machines have been considered (such a consideration is called a full cycle) and no improvement is obtained for a full cycle.
- Storer et al. (1992) presented an interesting local search method which is based on heuristic neighborhood scans instead of solution neighborhood scans (Nakado and Yamada, 1991). It combined heuristics on subsets of jobs in order to obtain new solutions. Different local search approaches are also proposed by other authors: Simulated Annealing (Van Laarhoven et al., 1992), Tabu search (Dell'Amico and Trubian, 1993),...

2. Problem formulation

In this section, an approach for the reduction of the job-shop problem to a flow-shop problem with job precedence constraints is presented. A brief review of flow-shop algorithms is included.

2.1. Reduction

Frequently, an industrial job-shop problem is in fact a flow-shop problem for most of the jobs. A lot of jobs use the machines in the same order and the job sequencing can be modelled as a flow-shop problem for these jobs. Some industries have used this job characteristic to organize and manage their production with a Just-in-time system, or to optimize the use of a circular conveyor. In this latter case, the most common job routing precedence relations allow the manager to define the machine layout on the conveyor in order to minimize its use, e.g., to minimize the distance travelled by pallets.

The basic idea of this paper is to show that an interesting approach for solving the job-shop problem is: first to find the most common route through the machines, i.e., the most common job routing precedence relations, and second to show that the machines can be used in the same order if the jobs are partitioned properly. The second step, job partitioning, is first presented in order to gain a better understanding of the problem objective of finding the most common route.

2.2. The job partitioning

The second step (job partitioning) is illustrated with the example presented next. For a job-shop problem with three machines $\{A,B,C\}$, there are 15 possible job routings: *A*, *B*, *C*, *A-B*, *A-C*, *B-A*, *B-C*, *C-A*, *C-B*, *A-B-C*, *A-C-B*, *B-A-C*, *B-C-A*, *C-A-B*, *C-B-A*. If the hypothesis of using the machines in order *A-B-C* is retained, the 15 previous routings which model fifteen unrelated jobs (called old jobs) can be partitioned in 28 new jobs with precedence constraints in the following way:

$A \rightarrow A\text{-}\emptyset\text{-}\emptyset$
$A\text{-}B \rightarrow A\text{-}B\text{-}\emptyset$
$B\text{-}C \rightarrow \emptyset\text{-}B\text{-}C$
$A\text{-}B\text{-}C \rightarrow A\text{-}B\text{-}C$
$B\text{-}C\text{-}A \rightarrow \emptyset\text{-}B\text{-}C \ll A\text{-}\emptyset\text{-}\emptyset$

$B \rightarrow \emptyset\text{-}B\text{-}\emptyset$
$A\text{-}C \rightarrow A\text{-}\emptyset\text{-}C$
$C\text{-}A \rightarrow \emptyset\text{-}\emptyset\text{-}C \ll A\text{-}\emptyset\text{-}\emptyset$
$A\text{-}C\text{-}B \rightarrow A\text{-}\emptyset\text{-}C \ll \emptyset\text{-}B\text{-}\emptyset$
$C\text{-}A\text{-}B \rightarrow \emptyset\text{-}\emptyset\text{-}C \ll A\text{-}B\text{-}\emptyset$

$C \rightarrow \emptyset\text{-}\emptyset\text{-}C$
$B\text{-}A \rightarrow \emptyset\text{-}B\text{-}\emptyset \ll A\text{-}\emptyset\text{-}\emptyset$
$C\text{-}B \rightarrow \emptyset\text{-}\emptyset\text{-}C \ll \emptyset\text{-}B\text{-}\emptyset$
$B\text{-}A\text{-}C \rightarrow \emptyset\text{-}B\text{-}\emptyset \ll A\text{-}\emptyset\text{-}C$
$C\text{-}B\text{-}A \rightarrow \emptyset\text{-}\emptyset\text{-}C \ll \emptyset\text{-}B\text{-}\emptyset \ll A\text{-}\emptyset\text{-}\emptyset$

The job operation partitioning introduces some precedence constraints among the new jobs and some fictitious operations with processing time equal to zero. The precedence constraints are represented by

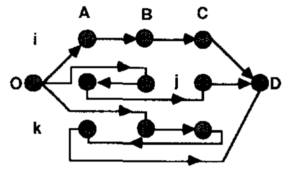

Fig. 1. 3 × 3 job-shop instance.

the symbol ‘ $\ll$ ’ and the fictitious operations by the symbol ‘$\emptyset$’. If two jobs are linked by a precedence constraint, the last operation of the first job (called predecessor) must be completed before beginning the first operation of the second job (called successor).

This idea is simple, however, it has not been published.

The job partitioning does not reduce the solution space. To prove this assertion we model our problem before and after the job transformation with an activity-on-node network. A job-shop problem can be represented by a graph $G = (X,U)$, where:

- X is the set of nodes which represent the job operations and two fictitious operations O (operation ‘beginning’) and D (operation ‘end’).
- U is the set of arcs which model the precedence relations of the job operations. Operation i is connected to operation j if a routing specifies that i precedes j. Operation O is connected to each first job operation and each last job operation is connected to operation D.

Fig. 1 illustrates a 3 × 3 job-shop instance and Fig. 2 shows the resulting 5 × 3 flow-shop instance after job partitioning. The machine order *A-B-C* has been retained. In graph *G*, a job is represented by a path from *O* to *D*. Graph *G* models the entire solution space. To build a schedule, arcs must be introduced between the operations requiring the same

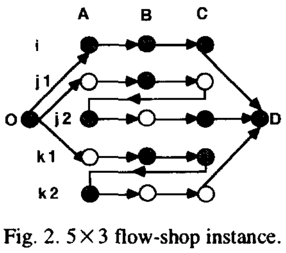

Fig. 2. 5 × 3 flow-shop instance.

machine, i.e., between two paths from O to D. Adding such an arc, decreases the solution space. A job partitioning introduces some new nodes (fictitious operations), it deletes a path from O to D (old job) and it creates a new path from O to D (new jobs). The solution space has not been decreased as no arc is added between two paths (two old paths, or a new path and an old path). Furthermore, a job being processed on each machine once and only once, a new path links only fictitious operations with a real operation on the same machine.

The job partitioning does not extend the solution space because only nodes which represent fictitious operations (i.e., operations with zero processing time) have been added to the graph and they can be neglected in all solutions. The fictitious operations are assumed to be processed just after the completion of the preceding operation. If there are no preceding operations they are assumed to have been completed at the beginning of the schedule horizon.

The job partitioning does not modify the solution space. It can always be achieved. In the worst case, for each operation of an old job a new job is created. For example, job routing *C-B-A* generates 3 new jobs when machine order *A-B-C* is retained.

During the job partitioning, the hypothesis that each new job is processed on each machine (even considering zero processing time) is retained, in order to use existing permutation flow-shop algorithms.

The job partitioning process can be expressed as follows:

```
LET:
OJ: the old job, NJ: the new job,
OP: the operation index of the old job, NP: the operation index of the new job,
OM: the retained machine order, MP the machine rank, $M$ the number of machines.
DO:
STEP 1. Select an old job OJ, Set OP to 1.
STEP 2. Create a new job NJ,
Set NP to 1, Set MP to 1.
STEP 3. IF the machine of rank MP in OM is the machine required for the OP th operation of job OJ
THEN the NP th operation of job NJ is the OP th operation of job OJ, Set OP to OP + 1,
ELSE the NP th operation of job NJ is a fictitious operation with zero processing time.
Set MP to MP + 1, Set NP to NP + 1.
IF MP is lower than or equal to $M$ THEN GOTO STEP 3.
IF OP is lower than or equal to $M$ THEN Create a precedence constraint between the current new job and the next new job and GOTO STEP 2.
IF an old job has not been selected THEN GOTO STEP 1.
```

2.3. The most common route

The first step of the reduction defines the problem of ordering the machines with the objective of minimizing the number of precedence constraints among the new jobs generated by partitioning. This helps to ensure the quality of solutions generated by the flow-shop heuristics and to decrease the size of the resulting flow-shop problem. A precedence constraint is generated among the new jobs when ever a job routing precedence relation between machines is not respected by the proposed machine order.

This formulation leads to a shortest hamiltonian path problem which is NP complete. The machines can be modelled by cities and the occurrences of the job routing precedence relations between machines can be represented by distances between cities. The objective is to maximize the number of job routing precedence relations verified, then to minimize the number of job routing precedence relations which are not verified, i.e., a distance travelled $D(i,j)$. Regarding all job routings, $D(i,j)$ is equal to the number of times the i,j sequence is not followed. Next, an example is presented to illustrate this point. Table 1 presents the job routings. Fig. 3 models the problem and Table 2 defines the distance matrix D. It is important to observe that the distance matrix is non-ordered ($D(4,1) > D(3,1)$ and $D(4,2) < D(3,2)$; $D(1,2) > D(1,3)$ and $D(4,2) < D(4,3)$) and it cannot be reduced to a triangular matrix which respectively defines two particular cases of shortest hamiltonian path (Lawler, 1971; Smith et al., 1975). These two last cases can be solved in polynomial time. The optimal value of the objective function is equal to $D(2,1) + D(1,3) + D(3,4) = 10$ for the machine sequence 2-1-3-4.

Table 1
Job routings

4-2-3-1
2-1-3-4
3-4-2-1
1-2-3-4
2-3-4-1
2-1-4-3
2-3-1-4
1-2-3-4
1-3-4-2
1-2-3-4
2-1-3-4

The nearest insertion rule has been selected in order to solve this problem (Christofides et al., 1979). This polynomial procedure starts by searching for the smallest distance $D(i,j)$. A path from i to j is defined. The length of the path is equal to $D(i,j)$. Then, the vertices which do not belong to the path are sequentially added. At each stage, the vertex which minimizes the increase in the path length is added into the path. It is inserted in the position which minimizes the increase of the path length. This process is repeated until all vertices have been included in the path.

2.4. The job-shop approach

We now specify an approach which reduces the original job-shop problem to a flow-shop problem with job precedence constraints: First, define the most common route through the machines, and second, partition the jobs into new jobs which respect the selected route.

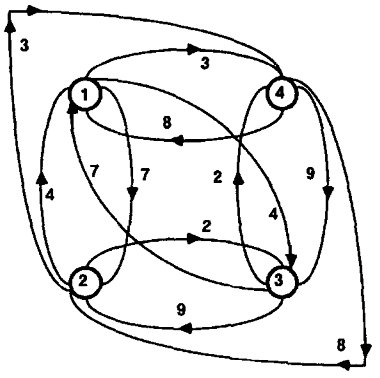

Fig. 3. The routing graph.

Table 2
Distance matrix

TO →	D	1	2	3	4
↑ FROM	1	0	7	4	3
	2	4	0	2	3
	3	7	9	0	2
	4	8	8	9	0

The job routings are respected by adding precedence constraints among jobs.

Having solved a permutation flow-shop problem, a global solution is defined for all the machines. Some new jobs, which have been defined by the partitioning step, have operations with zero processing times. Several new jobs represent an old job. According to these characteristics, some new jobs may disappear for a particular machine and the job order for each machine is not necessarily the same regarding old jobs. For example, if the schedule selected for the flow-shop problem of Fig. 2 is $\{k_1,j_1,i,j_2,k_2\}$, the job order is: $\{i,j,k\}$ for machine A (due to k_1 and j_1 having fictitious operations on A), $\{k,j,i\}$ for machine B (due to j_2 and k_2 having fictitious operations on B), and $\{k,i,j\}$ for machine C (due to j_1 and k_2 having fictitious operations on C).

2.5. Flow-shop heuristics

The flow-shop problem is one of the most studied problems in scheduling. Numerous reviews have been published (Graves, 1981; Proust, 1992; MacCarthy and Liu, 1993). Johnson (1954) proposed an optimal job ordering rule to solve the flow-shop problem with two machines and some particular cases with three machines. For the general case with M machines ($M > 3$), the problem is NP hard (Rinnooy Kan, 1976). Many heuristics have been offered to solve it and most of them are based on Johnson's result (Proust, 1992).

Palmer (1965), Gupta (1971), Dannenbring (1977), Hundal and Rajgopal (1988) proposed various job ordering rules. Campbell et al. (1970), Townsend (1977), Nawaz et al. (1983) defined some partial enumerative methods which involve some local searches of solutions or sub-solutions. Most of the authors have studied the maximum completion time criterion (Palmer, 1965; Campbell et al., 1970; Gupta, 1971; Dannenbring, 1977; Nawaz et al., 1983; Hundal and Rajgopal, 1988) and other authors considered the maximum lateness criterion (Townsend, 1977; Grabowski, 1980; Brucker, 1981; Grabowski et al., 1983). The maximum completion time criterion is a special case of the maximum lateness criterion where all due-dates are equal to zero. Some algorithms which have been defined for this second criterion can also be good algorithms for the first criterion (Guinet and Solomon, 1996).

Some authors have studied cases with additive characteristics such as: setup and removal times (Sule and Huang, 1983; Gupta and Darrow, 1986; Proust et al., 1991a), conveyance times (Szwarc, 1983) and no-waiting times (Panwalkar and Woollam, 1979). Sidney (1979) and Monma (1979) studied the case with two machines and series-parallel precedence constraints. Optimal algorithms have been proposed, but these tools have been defined for the case of precedence constraints among job operations. If jobs i and j are linked by a precedence constraint then each operation of job i must precede each operation of job j on every machine. For the present problem, if jobs i and j are linked by a precedence constraint between entire jobs, the last job i operation must precede the first job j operation. This defines another case of precedence constraints (Rinnooy Kan, 1976) and these algorithms are not relevant.

3. Solution

In this section, three heuristics to schedule N jobs with precedence constraints between entire jobs in a flow-shop environment (permutation flow-shop), are proposed. Based on the fact that not every job permutation is a solution to the problem studied, the most known constructive heuristics (Campbell et al., 1970; Townsend, 1977; Nawaz et al., 1983) have been adapted for the case of M machines with job precedence constraints among jobs (Park et al., 1984).

3.1. Extension of the Johnson's rule

The Johnson's rule determines the optimal solution to the two machine flow-shop problem.

It states that job i must be ordered before job j if:

$$\text{Min}[A(i),B(j)] \le \text{Min}[A(j),B(i)]. \tag{1}$$

The processing time of job i on the first machine is called $A(i)$ and the processing time of job i on the second machine is called $B(i)$. They are either positive or zero.

The job completion times have been modified in order to verify the job precedence constraints using the Johnson's rule:

$$A'(j) := A(j) \quad \text{if } j \text{ has no predecessor,} \tag{2}$$

$$A'(j) := A'(i) + B(i) + A(j)$$
$$\text{if } i \text{ is the immediate predecessor of } j, \tag{3}$$

$$B'(i) := B(i) \quad \text{if } i \text{ has no successor,} \tag{4}$$

$$B'(i) := B'(j) + A(j) + B(i)$$
$$\text{if } i \text{ is the immediate predecessor of } j. \tag{5}$$

The predecessor job processing times have been added to the job j processing time on machine A. The successor job processing times have been added to the job i processing time on machine B.

It is stated that job i must be ordered before job j if:

$$\text{Min}[A'(i),B'(j)] \le \text{Min}[A'(j),B'(i)]. \tag{6}$$

According to these new job completion times, if job i precedes job j:

$$A'(i) \le A'(j) = A'(i) + B(i) + A(j), \tag{7}$$

$$B'(j) \le B'(i) = B'(j) + A(j) + B(i) \tag{8}$$

and i is scheduled before j because $\min[A'(i),B'(j)] \le \text{Min}[A'(j),B'(i)]$.

The extension of the Johnson's rule does not guarantee optimality to the two machine problem due to the structure of the precedence graph of jobs (parallel precedence graph then a forest) and the zero processing times of fictitious operations which corrupt the rule. Rinnooy Kan (1976) proves that scheduling N jobs on a 2 machine flow-shop with tree precedence constraints between entire jobs is an NP hard problem for the makespan criterion.

An example with 6 jobs is presented in order to illustrate this fact. Table 3 presents the real and modified processing times of jobs. Job 1 and job 3 are respectively the predecessors of job 2 and job 4 due to the processing on machine B before machine A. The machine order A-B is retained.

Table 3
Processing times

	1	2	3	4	5	6
A	0	1	0	2	2	3
B	1	0	5	0	1	1
A'	0	2	0	7	2	3
B'	2	0	7	0	1	1

The job order obtained with the Johnson's rule is {1,3,6,5,4,2} with the correspondent makespan of 9. The optimal makespan is 8 as it can be observed in Fig. 4. This optimal solution is obtained by exchanging jobs 4 and 2 in Johnson's order.

3.2. The Campbell et al. heuristic

The heuristic of Campbell et al. (1970) is based on the Johnson's rule (Johnson, 1954). This heuristic determines a good solution to the M-machine flow-shop problem ($M > 3$) with the maximum completion time criterion. It solves M two-machine problems and it retains the best of the M solutions to solve the M-machine problem. To obtain a two-machine flow-shop problem, it calculates processing times $(A(i), B(i))$ for two fictitious machines A and B according to real processing times $(p(i,k))$:

$$A(i) = \sum_{k=1}^{h} p(i,k), \quad B(i) = \sum_{k=M-h+1}^{M} p(i,k). \tag{9}$$

The job order obtained with the Johnson's rule is a solution to the M-machine problem. Thus, the heuristic solves the fictitious problem with this rule and calculates the makespan of Johnson's schedule regarding real processing times. It repeats it for h for the range 1 to M. For the Mth two-machine problem, $A(i)$ is equal to $B(i)$ for each job i. In case of equality between processing times on machine A

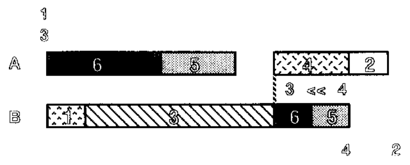

Fig. 4. Johnson's order.

and B, if the implementation of the Johnson's rule retains only the processing times on machine B ($B(j) \le B(i)$), the Johnson's rule is equivalent to MWKR rule. The Campbell et al. heuristic is used with the extension of the Johnson's rule.

This heuristic is also used with the Dannenbring's weighting scheme (Dannenbring, 1977). The processing times $A(i)$ and $B(i)$ are calculated here as follows:

$$A(i) = \sum_{k=1}^{h} (M-k+1) * p(i,k),$$

$$B(i) = \sum_{k=M-h+1}^{M} k * p(i,k). \tag{10}$$

3.3. The Nawaz et al. heuristic

The Nawaz et al. (1983) heuristic finds a good quality solution of the M-machine flow-shop problem ($M > 3$) with the maximum completion time criterion. It uses a scheme that a greater attention is given to jobs with large processing times (i.e. the total processing time of job operations). First, it orders the two larger processing time jobs in order to minimize the sequence maximum completion time. Next, it selects an unscheduled job with the largest processing time. It inserts this job into the scheduled jobs assuming the preceding job order, in the position which minimizes the sequence maximum completion time. The insertion step is repeated until all the jobs are scheduled. This algorithm has been adapted in order to respect the precedence constraints during the two steps. A job is only selected if its predecessors are scheduled and then it is inserted afterwards.

This heuristic is also used with various job priority schemes. In the previous version, the jobs are selected according to the rule of the largest total processing time of job operations, i.e., MWKR rule. Others rules can be also considered, in particular a partial sum of job operation processing times. Varying h from 1 to M, the jobs are selected in decreasing order of the sum of job operation processing times from machine 1 to h, and the Nawaz et al. heuristic is repeated for each order. The best of the

M calculated schedules is retained. In our extension of the Nawaz et al. algorithm, the Dannenbring's weighting scheme has been integrated. Operation processing times are multiplied by $h - k + 1$.

3.4. The Townsend's heuristic

Townsend (1977) proposes a branch and bound algorithm to solve the M-machine flow-shop problem with the maximum lateness criterion. This method does not guarantee optimality as it stops once the first feasible solution is found. The branching rule is based on job pre-sequence (a part of a scheduling sequence). The algorithm constructs the job sequence from first to last. For each branch a lower bound is calculated. It is equal to the maximum of the lateness of the sequenced jobs and an estimation of the maximum lateness of the unsequenced jobs. The estimated value is determined using the Jackson's rule (Earliest Due Date rule) which allows us to calculate the maximum lateness of the unsequenced jobs for each machine. The Jackson's rule minimizes the maximum lateness of the single machine scheduling problem (Jackson, 1955).

For our formulation, all the job due-dates are equal to zero. With regard to precedence constraints, only the branches which respect these constraints are considered. This last principle reduces the computation time of the heuristic.

3.5. Job-shop heuristics

Some job-shop priority rules and the Adams et al. method have been also considered to schedule the jobs in order to demonstrate the efficiency of the previously discussed heuristics. They have been applied to the job-shop problem without transforming the jobs. The MWKR rule, the LWKR rule, the SPT rule and the LPT (Longest Processing Time) rule have logically been selected (Baker, 1974; Gonzalez and Sahni, 1978; French, 1982; Carlier, 1982). For a given job-shop problem, these four rules are applied and the best solution is retained. The Adams et al. method has been programmed bounding the Carlier's Branch and Bound algorithm to 100 nodes.

4. Computational experiments

The performance of the heuristics was examined by solving 3040 different problems. Problems of 40 different sizes have been studied for two series of processing times and two groups of job routings. For each problem size, each processing time series and each job routing group, 32 test data sets were generated. Each problem instance has randomly generated routings and processing times.

4.1. Data

The number of jobs to be processed were: 6, 10, 20, 30, 40 and 50. The numbers of machines were: 3, 4, 5, 6, and 7. The job processing times are integer and positive. To generate the processing times, a log-normal distribution was employed (Bernad, 1973). The first series was generated with a mean of 30 and a variance of 225. The second series was generated with a mean of 20 and a variance of 25.

In a first set of problems, the job routings, i.e., the job operation precedence relations through machines were generated randomly. In a second set of problems, each job had to pass through a first set of machines and next through a second set of machines (Storer et al., 1992). This case can model two workshops with job-shop organization, in series. The precedence relations for each of these two sets were independently and randomly generated. These last problems have proven to be more difficult for heuristic procedures test to date. The Locho software (Proust et al., 1991b) was used to compare the heuristics. It allows us to generate data, to run the heuristics and to calculate statistics about the results.

4.2. Lower bounds

To evaluate the performance of heuristics, two sets of lower bounds have been defined. The first set of lower bounds is based on resource requirements:

$$\mathrm{LBJ} = \max_{\forall i=1,\dots,N} \left[\sum_{j=1}^{M} p(i,j) \right], \tag{11}$$

$$\mathrm{LBM} = \max_{\forall j=1,\dots,M} \left[\sum_{i=1}^{N} p(i,j) \right]. \tag{12}$$

LBJ is equal to the largest processing time of the jobs, LBM is equal to the largest machine requirement.

The second set of lower bounds is based on preemptive scheduling. The Jackson's algorithm finds the optimal schedule of N preemptive jobs on a single machine with release and due-dates (Jackson, 1955). The objective is to minimize the maximum lateness. Considering the fact that a common job due-date equals zero and the precedence relations of job operations, a preemptive scheduling problem is modelled for each machine. For a given machine k, job i release date is equal to the total processing time of the previous job operations, as job i cannot be processed on machine k before the processing on the previous machines has been completed. Job i due-date is equal to the total processing time of the following job operation multiplied by minus one. It models an adjusted due-date regarding the common due-date which is equal to zero. In our case, minimizing of the maximum job lateness is equivalent to minimizing of the makespan, because a job lateness is equal to the difference of the job completion time and the job due-date, i.e., the sum of the job completion time and the processing times of the following job operations. The Jackson's algorithm defines an extension of the MWKR rule. More details can be found in Carlier and Pinson (1994). The maximum makespan LBL is selected after the study of each machine k, for the range 1 to M.

The maximum lower bound LB is equal to the greatest of the three lower bounds: LBJ, LBM and LBL.

4.3. Computational results

The heuristics were programmed in C language and were run on the test problems described earlier on a 75 Mhz micro-computer pentium. Tables 4 and 5 present the results of the experiments with all precedence relations randomly generated and Table 6 shows the results of the experiments with part of precedence relations randomly generated.

The first column represents the problem number P, the second column contains the number of jobs N, the third column represents the number of machines M, the fourth column defines the series S of processing times used. Columns five to 18 provide the average (mean), and the maximum (max) of the ratios of the difference between the heuristic solution value (HEU) and the lower bound (LB) for the 32 test data sets. This ratio is equal to (HEU − LB)/LB and:

- for the Campbell et al. algorithm, the column is labeled CDS1,
- for the Campbell et al. algorithm with Dannenbring's weighting scheme, the column is labeled CDS2,
- for the Nawaz et al. algorithm, the column is labeled NEH1,
- for the Nawaz et al. algorithm with multiple job priority rules, the column is labeled NEH2,
- for the Townsend's algorithm, the column is labeled TOW,
- for Job-shop priority rules, the column is labeled PRI,
- for the Adams et al. method, the column is labeled ABZ.

In Table 6, the third column is replaced with two columns which specify the number of machines for each of the two machine sets.

Examining the behavior of the heuristics for the experiments with all precedence relations randomly generated, the Adams et al. method performed best for most of the problem instances. The Nawaz et al. algorithm with multiple job priority rules had results close to Adams et al. results, particularly for the first series. It could be a very interesting alternative heuristic to take into account additive constraints based on machine relations such as limited storage. The Nawaz et al. algorithm without multiple job priority rules provide better results than the Campbell et al. algorithms and Job-shop priority rules. The Campbell et al. algorithm with the Dannenbring's weighting scheme and Job-shop priority rules provided good results, respectively for the first series and the second series.

For the cases with part of the precedence relations generated randomly, the Nawaz et al. algorithms with or without multiple job priority rules performed best for each problem instance, and clearly outperformed the others heuristics. The Nawaz et al. method with multiple job priority rules results were very compact. The maximums of the ratios of the Nawaz et al. algorithm with multiple job priority rules, were usually lower than the means of the ratios of the

Table 4
Maximum completion time results (series 1)

P	*N*	*M*	*S*	CDS1 mean	CDS2 mean	NEH1 mean	NEH2 mean	TOW mean	ABZ mean	PRI mean	CDS1 max	CDS2 max	NEH1 max	NEH2 max	TOW max	ABZ max	PRI max
1	6	3	1	6.67%	7.68%	6.43%	3.58%	17.08%	5.34%	20.41%	25.68%	35.12%	20.69%	14.01%	44.81%	21.03%	47.15%
2	6	4	1	19.03%	17.95%	17.26%	11.47%	24.03%	13.79%	26.57%	42.73%	42.31%	55.93%	41.84%	67.55%	31.41%	63.64%
3	6	5	1	19.80%	20.35%	26.19%	16.63%	37.49%	13.61%	30.03%	41.63%	41.77%	72.77%	49.22%	98.30%	29.32%	76.17%
4	6	6	1	36.69%	36.17%	32.16%	21.90%	50.63%	27.10%	35.41%	80.62%	80.62%	57.25%	35.09%	115.44%	79.61%	64.64%
5	6	7	1	37.23%	34.23%	31.64%	19.62%	51.60%	24.75%	31.01%	111.46%	111.46%	68.77%	41.75%	88.48%	52.46%	57.03%
6	10	3	1	2.62%	2.23%	3.91%	0.94%	10.55%	0.56%	14.22%	22.29%	22.29%	20.88%	10.33%	24.15%	5.67%	36.76%
7	10	4	1	14.08%	11.80%	11.01%	5.28%	21.56%	4.45%	23.06%	26.01%	25.00%	24.52%	20.43%	41.80%	17.81%	60.37%
8	10	5	1	25.64%	22.83%	20.28%	12.18%	39.89%	11.60%	30.01%	58.82%	48.66%	49.50%	24.00%	73.37%	30.48%	61.61%
9	10	6	1	32.86%	31.92%	31.72%	21.45%	56.73%	20.04%	40.79%	55.56%	57.14%	62.46%	43.05%	104.01%	47.34%	71.12%
10	10	7	1	39.01%	37.37%	37.25%	27.30%	61.00%	23.98%	39.91%	68.07%	64.13%	60.44%	42.57%	133.42%	47.04%	77.07%
11	20	3	1	2.84%	2.31%	2.69%	0.89%	8.15%	0.04%	11.70%	16.48%	16.48%	18.48%	5.11%	16.95%	1.15%	24.87%
12	20	4	1	9.09%	7.96%	4.34%	2.02%	15.04%	0.17%	17.39%	33.71%	24.50%	15.59%	8.99%	33.00%	1.84%	34.31%
13	20	5	1	18.30%	17.10%	9.65%	5.08%	23.13%	2.08%	19.12%	36.82%	34.46%	18.39%	16.89%	41.58%	14.52%	39.97%
14	20	6	1	27.05%	24.05%	14.38%	9.72%	37.00%	5.18%	25.97%	42.90%	50.99%	25.99%	16.54%	58.21%	24.14%	49.26%
15	20	7	1	31.16%	30.24%	19.91%	15.18%	44.03%	8.58%	31.60%	51.33%	47.25%	37.65%	30.99%	72.74%	29.67%	52.25%
16	30	3	1	1.26%	0.54%	1.57%	0.65%	5.65%	0.00%	5.99%	10.16%	3.66%	4.96%	4.18%	14.27%	0.00%	22.29%
17	30	4	1	4.70%	4.89%	3.65%	1.48%	11.35%	0.03%	13.26%	12.56%	12.56%	9.40%	6.23%	26.26%	1.12%	32.05%
18	30	5	1	12.02%	10.14%	7.38%	3.35%	15.86%	0.08%	14.30%	25.95%	23.20%	17.94%	8.25%	26.58%	1.86%	31.96%
19	30	6	1	20.37%	20.30%	9.42%	5.85%	28.07%	1.43%	22.68%	35.29%	41.78%	19.77%	13.09%	57.77%	9.69%	36.30%
20	30	7	1	25.54%	23.17%	12.80%	8.93%	33.28%	3.08%	24.50%	37.77%	37.86%	24.33%	15.12%	58.63%	15.59%	44.19%
21	40	3	1	1.26%	1.05%	1.54%	0.15%	4.29%	0.00%	6.65%	11.02%	11.02%	8.94%	2.03%	13.29%	0.00%	18.63%
22	40	4	1	4.15%	3.84%	3.02%	1.15%	9.05%	0.00%	10.54%	15.28%	15.55%	7.97%	5.71%	25.48%	0.00%	22.56%
23	40	5	1	9.83%	8.98%	5.31%	2.59%	12.37%	0.02%	14.74%	24.75%	29.42%	15.04%	8.17%	30.14%	0.51%	36.46%
24	40	6	1	15.19%	13.74%	6.13%	3.67%	22.26%	0.44%	16.78%	31.73%	30.23%	12.43%	8.04%	37.27%	6.76%	36.42%
25	40	7	1	20.93%	18.89%	8.79%	6.77%	31.11%	1.23%	20.68%	44.82%	29.57%	18.93%	13.73%	54.37%	7.12%	42.31%
26	50	3	1	0.83%	0.57%	1.41%	0.55%	3.94%	0.00%	5.30%	13.09%	13.09%	4.86%	4.86%	11.58%	0.00%	15.87%
27	50	4	1	2.11%	1.90%	1.87%	0.67%	6.98%	0.00%	9.09%	7.80%	6.04%	6.53%	3.04%	15.69%	0.00%	26.82%
28	50	5	1	8.12%	6.98%	3.12%	1.43%	11.84%	0.00%	13.95%	18.42%	18.42%	7.93%	4.05%	30.53%	0.00%	20.62%
29	50	6	1	16.30%	14.86%	4.56%	2.97%	19.60%	0.01%	15.96%	31.15%	28.60%	9.93%	8.92%	52.14%	0.36%	36.79%
30	50	7	1	19.14%	17.53%	6.49%	5.07%	23.52%	0.00%	16.32%	34.48%	40.36%	14.19%	10.81%	50.49%	0.00%	30.30%
Averages:				16.13%	15.05%	11.53%	7.28%	24.57%	5.59%	20.26%	35.61%	34.78%	26.42%	17.23%	50.61%	15.88%	42.33%

Table 5
Maximum completion time results (series 2)

				CDS1	CDS2	NEH1	NEH2	TOW	ABZ	PRI	CDS1	CDS2	NEH1	NEH2	TOW	ABZ	PRI
P	*N*	*M*	*S*	mean	mean	mean	mean	mean	mean	mean	max	max	max	max	max	max	max
31	6	3	2	8.82%	9.90%	9.30%	5.86%	16.31%	4.56%	16.37%	35.90%	35.90%	30.50%	30.50%	47.01%	19.86%	41.56%
32	6	4	2	22.05%	22.45%	18.57%	11.21%	27.29%	13.79%	22.55%	63.08%	63.08%	42.03%	27.21%	53.08%	28.57%	54.07%
33	6	5	2	25.52%	27.39%	30.46%	17.48%	42.00%	16.19%	24.35%	60.34%	78.05%	62.91%	44.72%	88.62%	47.40%	45.58%
34	6	6	2	40.64%	40.27%	36.59%	23.37%	63.22%	27.17%	31.63%	96.05%	96.05%	59.87%	50.30%	101.97%	66.86%	54.84%
35	6	7	2	43.91%	42.47%	34.18%	24.00%	54.06%	27.32%	27.74%	110.44%	110.44%	62.11%	41.54%	89.01%	59.90%	42.86%
36	10	3	2	3.07%	3.32%	7.31%	2.59%	11.45%	0.40%	8.68%	25.70%	25.70%	16.59%	11.82%	25.47%	6.54%	36.62%
37	10	4	2	17.56%	16.06%	13.81%	7.65%	23.75%	4.32%	21.60%	35.45%	29.33%	29.27%	18.18%	41.51%	16.03%	55.27%
38	10	5	2	27.07%	26.36%	22.74%	15.56%	42.48%	10.45%	23.62%	47.89%	45.73%	51.17%	31.92%	69.70%	28.25%	52.56%
39	10	6	2	36.97%	35.08%	32.89%	24.78%	59.79%	21.76%	33.12%	52.09%	52.84%	48.47%	39.47%	98.22%	45.33%	55.88%
40	10	7	2	47.97%	45.72%	40.79%	29.30%	74.81%	26.95%	36.16%	118.66%	118.66%	69.74%	46.15%	132.63%	50.67%	60.39%
41	20	3	2	2.88%	2.98%	6.27%	2.64%	7.24%	0.03%	5.70%	14.61%	22.30%	13.85%	8.64%	16.41%	0.94%	24.01%
42	20	4	2	11.21%	10.39%	7.83%	4.57%	16.19%	0.05%	12.85%	27.50%	27.82%	25.12%	12.00%	32.07%	0.92%	31.83%
43	20	5	2	22.01%	21.50%	12.47%	8.38%	26.13%	1.47%	16.90%	35.48%	39.07%	23.61%	16.02%	44.09%	10.24%	39.09%
44	20	6	2	30.20%	29.72%	17.93%	12.94%	40.11%	5.58%	23.90%	46.27%	42.30%	31.31%	19.90%	81.96%	22.00%	41.78%
45	20	7	2	36.59%	35.25%	22.94%	18.06%	54.28%	10.95%	26.80%	52.57%	59.76%	36.64%	27.41%	86.74%	38.23%	47.08%
46	30	3	2	0.59%	0.91%	3.81%	1.92%	5.84%	0.00%	4.66%	6.97%	11.11%	11.94%	6.63%	13.11%	0.00%	21.85%
47	30	4	2	5.28%	5.63%	6.04%	3.50%	9.78%	0.00%	8.48%	14.68%	14.68%	14.11%	8.64%	25.86%	0.00%	24.72%
48	30	5	2	14.46%	13.34%	8.63%	5.15%	16.21%	0.01%	10.98%	28.03%	26.11%	15.07%	10.95%	28.48%	0.31%	31.92%
49	30	6	2	24.87%	24.57%	13.26%	8.78%	29.92%	1.37%	20.01%	39.63%	37.11%	24.02%	15.29%	55.12%	16.07%	45.53%
50	30	7	2	30.03%	25.93%	17.29%	12.64%	43.37%	3.23%	19.77%	44.48%	39.63%	25.92%	21.32%	91.92%	21.63%	33.28%
51	40	3	2	1.14%	1.01%	4.63%	0.85%	4.26%	0.00%	3.01%	9.45%	8.33%	9.64%	3.63%	9.72%	0.00%	17.63%
52	40	4	2	5.24%	5.17%	5.76%	3.00%	8.90%	0.00%	6.28%	16.92%	17.31%	12.31%	8.59%	23.31%	0.00%	23.95%
53	40	5	2	10.51%	10.60%	8.10%	4.20%	14.20%	0.01%	10.15%	21.40%	19.17%	16.86%	9.44%	29.25%	0.26%	18.94%
54	40	6	2	18.20%	16.67%	9.11%	6.26%	22.53%	0.18%	12.81%	37.86%	38.56%	16.02%	12.83%	39.86%	4.07%	26.67%
55	40	7	2	24.81%	24.76%	13.00%	9.50%	37.08%	1.32%	16.80%	36.56%	37.04%	19.88%	17.02%	63.49%	13.29%	31.34%
56	50	3	2	1.24%	0.61%	3.82%	1.29%	3.69%	0.00%	2.46%	15.21%	8.27%	7.86%	5.62%	8.45%	0.00%	19.02%
57	50	4	2	2.78%	3.32%	4.67%	2.15%	6.92%	0.00%	3.00%	8.49%	9.46%	10.80%	8.76%	12.17%	0.00%	13.51%
58	50	5	2	9.60%	9.63%	5.99%	3.40%	12.20%	0.00%	8.01%	20.28%	20.28%	15.46%	6.54%	27.95%	0.00%	22.63%
59	50	6	2	17.74%	16.93%	7.57%	5.54%	20.77%	0.05%	13.36%	25.90%	26.39%	13.05%	11.11%	45.96%	1.63%	23.17%
60	50	7	2	22.10%	20.23%	10.04%	8.20%	25.77%	0.07%	12.81%	40.46%	41.91%	17.93%	15.46%	50.14%	2.12%	24.98%
Averages:				18.84%	18.27%	14.53%	9.49%	27.35%	5.91%	16.15%	39.61%	40.08%	27.80%	19.59%	51.11%	16.70%	35.42%

Table 6
Maximum completion time results (flow-like problems)

					CDS1	CDS2	NEH1	NEH2	TOW	ABZ	PRI	CDS1	CDS2	NEH1	NEH2	TOW	ABZ	PRI
P	*N*	*M*1	*M*2	*S*	mean	mean	mean	mean	mean	mean	mean	max	max	max	max	max	max	max
60	6	3	3	1	43.59%	44.24%	36.52%	28.08%	47.34%	40.40%	43.93%	95.04%	95.04%	64.92%	60.08%	81.14%	78.48%	74.70%
61	6	3	4	1	46.79%	43.28%	36.94%	30.84%	54.09%	53.57%	48.65%	80.25%	96.10%	64.98%	53.07%	86.64%	83.62%	96.44%
62	6	4	3	1	45.61%	47.18%	39.46%	30.89%	50.42%	45.08%	46.79%	83.04%	87.19%	62.08%	51.53%	96.40%	93.83%	80.95%
63	10	3	3	1	52.11%	49.53%	35.90%	27.99%	52.88%	56.13%	52.58%	94.23%	67.70%	59.58%	44.80%	83.83%	94.69%	77.47%
64	10	3	4	1	57.91%	56.81%	38.54%	31.73%	63.25%	67.45%	64.13%	90.46%	80.93%	59.80%	47.24%	102.19%	99.54%	89.29%
65	10	4	3	1	58.86%	57.45%	37.99%	31.18%	58.92%	65.86%	61.25%	119.57%	103.56%	60.22%	45.00%	85.11%	118.54%	101.11%
66	20	3	3	1	58.60%	53.73%	29.60%	24.90%	49.10%	59.04%	63.67%	74.61%	73.64%	46.50%	39.80%	70.36%	89.16%	81.38%
67	20	3	4	1	62.99%	64.47%	34.64%	30.76%	66.36%	86.49%	69.07%	82.47%	84.16%	46.48%	39.40%	110.77%	125.07%	105.03%
68	20	4	3	1	63.55%	61.09%	33.65%	28.09%	60.48%	72.77%	65.22%	131.47%	81.96%	49.65%	42.48%	79.32%	128.13%	96.54%
69	30	3	3	1	59.86%	59.82%	27.50%	23.64%	47.69%	71.31%	65.71%	90.43%	78.34%	40.40%	34.57%	77.43%	102.18%	96.67%
70	30	3	4	1	66.39%	67.40%	32.48%	28.74%	65.86%	94.55%	67.77%	86.69%	86.69%	44.57%	44.57%	87.36%	130.81%	83.90%
71	30	4	3	1	65.90%	66.33%	31.66%	27.70%	62.73%	86.75%	69.25%	92.91%	92.91%	44.25%	36.75%	77.72%	113.64%	90.95%
72	40	3	3	1	59.60%	60.85%	23.62%	21.21%	43.36%	70.74%	65.76%	80.09%	75.73%	37.33%	30.37%	61.36%	101.77%	101.74%
73	40	3	4	1	67.69%	66.74%	29.33%	26.19%	63.57%	102.29%	69.38%	83.78%	86.27%	43.02%	35.18%	107.12%	137.02%	84.47%
74	40	4	3	1	68.43%	65.70%	31.58%	27.32%	60.03%	82.89%	68.07%	89.10%	84.85%	40.86%	36.01%	91.27%	107.67%	94.84%
75	50	3	3	1	57.24%	58.89%	22.03%	19.56%	46.96%	72.31%	67.85%	75.38%	79.62%	33.94%	27.26%	69.20%	95.41%	81.27%
76	50	3	4	1	73.69%	70.30%	27.99%	25.96%	63.87%	106.40%	72.14%	89.93%	98.40%	42.37%	35.40%	118.88%	143.25%	87.13%
77	50	4	3	1	70.50%	68.52%	29.69%	26.04%	58.78%	89.33%	70.44%	93.44%	85.12%	45.62%	36.88%	78.09%	104.70%	88.49%
Averages:					59.96%	59.02%	32.17%	27.27%	56.43%	73.52%	62.87%	90.72%	85.46%	49.25%	41.13%	86.90%	108.20%	89.58%
78	6	3	3	2	39.86%	40.44%	33.23%	26.01%	47.28%	46.31%	42.99%	80.81%	80.81%	48.77%	45.20%	78.69%	79.33%	59.88%
79	6	3	4	2	45.83%	45.83%	39.77%	31.15%	55.14%	60.39%	43.53%	106.48%	106.48%	74.27%	50.00%	81.37%	101.02%	62.44%
80	6	4	3	2	45.48%	46.49%	40.11%	28.03%	53.27%	48.65%	45.27%	80.77%	86.39%	60.10%	46.32%	81.09%	92.23%	79.00%
81	10	3	3	2	48.84%	47.63%	33.47%	26.81%	52.44%	62.68%	52.95%	65.58%	63.41%	48.02%	35.29%	81.47%	93.26%	74.51%
82	10	3	4	2	57.17%	56.11%	35.52%	30.55%	69.15%	79.45%	61.49%	90.55%	83.15%	54.24%	49.25%	106.27%	112.37%	98.56%
83	10	4	3	2	62.14%	61.30%	36.11%	29.44%	62.63%	75.59%	57.42%	112.88%	109.25%	53.41%	47.22%	90.88%	113.60%	95.83%
84	20	3	3	2	57.53%	57.25%	28.18%	24.20%	50.59%	69.32%	66.13%	74.37%	74.37%	41.36%	35.67%	70.89%	104.34%	93.48%
85	20	3	4	2	64.67%	63.77%	34.52%	30.24%	72.76%	107.92%	65.11%	103.01%	103.01%	51.43%	40.99%	103.52%	147.20%	88.72%
86	20	4	3	2	67.91%	68.04%	35.01%	30.31%	66.70%	88.52%	69.06%	95.26%	134.70%	46.17%	41.85%	87.47%	133.77%	91.30%
87	30	3	3	2	61.48%	62.05%	25.49%	22.99%	51.00%	82.18%	69.44%	76.86%	79.62%	35.53%	29.56%	82.43%	98.68%	91.87%
88	30	3	4	2	67.23%	67.86%	31.88%	27.68%	69.62%	113.05%	70.45%	81.46%	82.25%	45.83%	37.04%	92.87%	155.86%	87.33%
89	30	4	3	2	68.89%	69.78%	33.10%	28.76%	64.57%	97.99%	69.98%	91.68%	102.84%	43.53%	33.97%	84.69%	127.68%	83.56%
90	40	3	3	2	62.44%	62.98%	23.11%	19.87%	48.41%	85.97%	70.57%	76.16%	78.00%	36.34%	26.37%	75.50%	111.92%	91.89%
91	40	3	4	2	68.71%	67.69%	27.60%	25.21%	67.01%	118.87%	71.17%	82.06%	83.56%	36.70%	33.65%	101.95%	147.05%	87.10%
92	40	4	3	2	70.77%	70.24%	31.55%	27.01%	66.95%	98.35%	71.49%	94.06%	82.35%	39.63%	34.88%	84.28%	121.42%	82.03%
93	50	3	3	2	63.74%	63.30%	21.28%	18.59%	48.83%	83.82%	71.90%	79.40%	78.78%	35.68%	28.92%	66.17%	108.57%	87.42%
94	50	3	4	2	73.79%	72.81%	26.55%	23.83%	69.83%	127.92%	72.10%	87.93%	87.30%	32.16%	29.27%	99.52%	163.18%	77.32%
95	50	4	3	2	74.87%	73.31%	28.69%	25.62%	65.58%	103.45%	72.85%	99.62%	102.08%	39.58%	34.70%	89.22%	124.64%	88.49%
Averages:					61.19%	60.94%	31.40%	26.46%	60.10%	86.14%	63.55%	87.72%	89.91%	45.71%	37.79%	86.57%	118.67%	84.49%

Table 7
Computation times

N	*M*	ABZ	NEH1	NEH2	TOW	N	M	ABZ	NEH1	NEH2	TOW
6	3	< 1	< 1	< 1	< 1	30	3	1	1	2	3
6	4	< 1	< 1	< 1	< 1	30	4	2	2	5	10
6	5	< 1	< 1	< 1	< 1	30	5	2	3	10	16
6	6	< 1	< 1	< 1	< 1	30	6	4	3	16	30
6	7	< 1	< 1	< 1	< 1	30	7	7	4	26	53
10	3	< 1	1	1	1	40	3	1	2	4	8
10	4	1	1	1	1	40	4	2	3	11	25
10	5	1	1	1	1	40	5	3	5	22	47
10	6	1	1	1	1	40	6	9	7	39	86
10	7	1	1	2	1	40	7	16	10	69	165
20	3	< 1	1	1	1	50	3	1	3	8	21
20	4	1	1	2	2	50	4	4	6	22	53
20	5	2	1	2	4	50	5	5	9	48	102
20	6	2	1	5	7	50	6	13	14	88	200
20	7	3	2	8	10	50	7	29	22	156	333

other heuristics. The Townsend's algorithm provided good results.

The Campbell et al. and Nawaz et al. heuristics are polynomial computation time algorithms, their time complexity are respectively $O(M * N * \log(N))$ and $O(M * N^2)$. The time complexity of the Nawaz et al. algorithm with multiple job priority rules is $O(M^2 * N^2)$. The Townsend's algorithm resulted in the worst computation times $(O(M * N^3 * \log(N)))$. The time cost of the Adams et al. method is practically $O(M^2 * N^2 * \log(N))$. The time complexity of the nearest insertion rule which is used to find shortest hamiltonian paths, is $O(M^3)$. Table 7 shows the computation times for the most costly heuristics. The Campbell et al. algorithms and Job-shop priority rules never exceed one second.

5. Conclusion

In this paper a reduction of job-shop problems to flow-shop problems with job precedence constraints between entire jobs was presented. Some known algorithms for the flow-shop problem were modified to take into account job precedence constraints and their performance was studied comparing with job-shop heuristics such as the Adam's et al. method. The results of the computational study indicate that the Adams et al. method provided the best quality results for pure job-shop problems, and both versions of the Nawaz et al. algorithm provided the best quality results for 'flow-like' problems which have proven to be more difficult for heuristic procedures test to date. The Nawaz et al. algorithm defines an approach which can be easily modified to take into account additive constraints based on machine relations such as: limited storage, limited conveyance resources (pallets), minimum and maximum operation waiting times. The proposed heuristics can also be used to minimize the maximum lateness. The Townsend's algorithm was first defined to take into account this objective. For the Campbell et al. and Nawaz et al. methods it is possible to define a machine $M + 1$ with job processing times equal to the maximum of the due-dates minus the job due-date. A natural extension of the research presented would be the study of generalized job-shop problems, i.e., the job-shop problem with a multiple use of the same resource.

Acknowledgements

The authors sincerely thank both the referees for their constructive and thoughtful criticisms and suggestions which led to substantial improvement of earlier versions of this paper.

References

Adams, J., Balas, E., Zawack, D., 1988. The shifting bottleneck procedure for job-shop scheduling. Management Science 34, 391–401.

Akers, S.B., 1956. A graphical approach to production scheduling problems. Operations Research 4, 244–255.

Applegate, D., Cook, W., 1991. A computational study of job-shop scheduling. ORSA Journal of Computing 3, 149–156.

Baker, K.R., 1984. Sequencing rules and due-date assignments in a job shop. Management Science 30, 1093–1104.

Baker, K.R., 1974. Introduction to Sequencing and Scheduling. John Wiley and Sons, Inc., New-York.

Balas, E., 1969. Machine sequencing via disjunctive graphs: An implicit enumeration algorithm. Operations Research 17, 941–957.

Barker, J.R., MacMahon, G.B., 1985. Scheduling the general job shop. Management Science 31, 594–598.

Bernad, J., 1973. La distribution log-normale dans les problèmes industriels. Production et Gestion 252, 21–27.

Brucker, P., 1988. An efficient algorithm for the job-shop problem with two jobs. Computing 40, 353–359.

Brucker, P., 1981. Minimizing maximum lateness in a two machine unit-time job shop. Computing 27, 367–370.

Campbell, H.G., Dudek, R.A., Smith, M.L., 1970. A heuristic algorithm for the *n* job *m* machine sequencing problem. Management Science 16, B630–B637.

Carlier, J., Pinson, E., 1994. Adjustment of heads and tails for the job shop problem. European Journal of Operational Research 78, 146–161.

Carlier, J., Pinson, E., 1989. An algorithm for solving the job shop problem. Management Science 35, 164–176.

Carlier, J., 1982. The one machine problem. European Journal of Operational Research 11, 42–47.

Charalambous, O., Hindi, K.S., 1991. A review of artificial intelligence-based job-shop scheduling systems. Information and Decision Technologies 3, 189–202.

Christofides, N., Mingozzi, A., Toth, P., Sandi, C., 1979. Combinatorial Optimization. John Wiley and Sons, Inc., New York.

Dannenbring, D.G., 1977. An evaluation of Flow shop sequencing heuristics. Management Science 23, 1174–1192.

Dell'Amico, M., Trubian, M., 1993. Applying tabu search to the job-shop scheduling problem. Annals of Operation Research 41, 231–252.

Fisher, M.L., Lageweg, B.J., Lenstra, J.K., Rinnooy Kan, A.H.G., 1983. Surrogate duality relaxation for job-shop scheduling. Discrete Applied Mathematics 5, 65–75.

French, S., 1982. Sequencing and Scheduling, An Introduction to the Mathematics of the Job-Shop. Ellis-Horwood Ltd.

Gonzalez, T., Sahni, S., 1978. flow-shop and job-shop schedules: Complexity and approximation. Operations Research 26, 36–52.

Grabowski, J., Skubalska, E., Smutnicki, 1983. On flow-shop scheduling with release and due dates to minimize maximum lateness. Journal of Operational Research Society 34, 615–620.

Grabowski, J., 1980. On two machine scheduling with release and due dates to minimize maximum lateness. Opsearch 17, 133–154.

Graves, S.C., 1981. A review of production scheduling. Operations Research 29, 646–675.

Guinet, A.G.P., Solomon, M.M., 1996. Scheduling hybrid flow-shops to minimize tardiness or maximum completion time. International Journal of Production Research 34, 1643–1654.

Gupta, J.N.D., 1971. A functional heuristic algorithm for the flow-shop scheduling problem. Operation Research Quarterly 22, 39–47.

Gupta, J.N.D., Darrow, W.P., 1986. The two-machine sequence dependent flow-shop scheduling problem. European Journal of Operational Research 24, 439–446.

Hardgrave, W.W., Nemhauser, G.L., 1963. A geometric method and a graphical algorithm for a sequencing problem. Operations Research 12, 655–679.

Hundal, T.S., Rajgopal, J., 1988. An extension of Palmer's heuristic for the flow-shop scheduling problem. International Journal of Production Research 26, 1119–1124.

Jackson, J.R., 1956. An extension of Johnson's results on job lot scheduling. Naval Research Logistics Quarterly 3, 201–203.

Jackson, J.R., 1955. Scheduling a production line to minimize maximum tardiness. Research report 43, University of California, Los Angeles.

Johnson, S.M., 1954. Optimal two and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1, 61–68.

Lageweg, B.J., Lenstra, J.K., Rinnooy Kan, A.H.G., 1977. Job-shop scheduling by implicit enumeration. Management Science 24, 441–450.

Lawler, E.L., 1971. A solvable case of the travelling salesman problem. Mathematics Programming 1, 267–269.

Liao, C.J., You, C.T., 1992. An improved formulation for the job-shop scheduling problem. Journal of Operational Research Society 43, 1047–1054.

MacCarthy, B.L., Liu, J., 1993. Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling. International Journal of Production Research 31, 59–79.

Nakado, R., and Yamada, T., 1991. Conventional genetic algorithm for job-shop problems. In: Proc. Fourth Int. Conf. on Genetic Algorithms. Morgan Kaufmann, San Mateo, California, pp. 474–479.

Monma, C.L., 1979. The two-machine maximum flow-time problem with series-parallel precedence constraints: An algorithm and extensions. Operations Research 27, 792–798.

Nawaz, M., Enscore, E., Ham, I., 1983. A heuristic algorithm for the *m* machine *n* job flow-shop sequencing problem. Omega 11, 91–95.

Palmer, D.S., 1965. Sequencing jobs through a multi stage process in the minimum total time. A quick method of obtaining a near optimum. Operation Research Quarterly 16, 101–107.

Panwalkar, S.S., Woollam, C.R., 1979. Flow-shop scheduling problems with no in-process waiting: A special case. Journal of Operational Research Society 30, 661–664.

Panwalkar, S.S., Iskander, W., 1977. A survey of scheduling rules. Operations Research 25, 45–61.

Park, Y.B., Pegden, C.D., Enscore, E.E., 1984. A survey and evaluation of static flow-shop scheduling heuristics. International Journal of Production Research 22, 127–141.

Proust, C., 1982. Using Johnson's algorithm for solving flow-shop problems. Summer School on scheduling theory and its applications. INRIA, Bonnas, France, pp. 297–342.

Proust, C., Gupta, J.N.D., Deschamps, V., 1991a. Flow-shop scheduling with setup processing and removal times separated. International Journal of Production Research 29, 479–493.

Proust, C., Ferreira, A., Bijaoui, J., 1991b. The interprogramming paradox: The diversity of softwares for an effective integration. International Congress of Industrial Engineering, Tours, pp. 1157–1166.

Rinnooy Kan, A.H.G., 1976. Machine Scheduling Problem: Classification, Complexity and Computations. Nijhoff, The Hague.

Sidney, J.B., 1979. The two-machine maximum flow time problem with series parallel precedence relations. Operations Research 27, 782–791.

Smith, M.L., Panwalkar, S.S., Dudek, R.A., 1975. Flow-shop sequencing problem with ordered time matrices. Management Science 21, 544–549.

Storer, R.H., David, W.S., Vaccari, R., 1992. New search spaces for sequencing problems with application to job shop scheduling. Management Science 38, 1495–1509.

Sule, D.R., Huang, K.Y., 1983. Sequency on two and three machines with setup, processing and removal times separated. International Journal of Production Research 21, 723–732.

Szwarc, W., 1983. Flow shop problems with time lags. Management Science 29, 477–481.

Townsend, D.W., 1977. Sequencing *n* jobs on *m* machines to minimize tardiness: A branch and bound solution. Management Science 23, 1016–1019.

Van Laarhoven, P.J.M., Aarts, E.H.L., Lenstra, J.K., 1992. Job-shop scheduling by simulated annealing. Operation Research 40, 113–125.