An Algorithm for Solving Job Shop Problem

Article *in* Management Science · February 1989

DOI: 10.1287/mnsc.35.2.164 · Source: RePEc

2 authors:

Jacques Carlier
Université de Technologie de Compiègne
145 PUBLICATIONS **4,303** CITATIONS

SEE PROFILE

Pinson Eric
Catholic University of the West
59 PUBLICATIONS **1,882** CITATIONS

SEE PROFILE

MANAGEMENT SCIENCE
Vol. 35, No 2, February 1989
Printed in U S A

AN ALGORITHM FOR SOLVING THE JOB-SHOP PROBLEM*

J. CARLIER AND E. PINSON
Université de Technologie de Compiègne
Institut de Mathématiques Appliquées d'Angers

In this paper, we propose a branch and bound method for solving the job-shop problem. It is based on one-machine scheduling problems and is made more efficient by several propositions which limit the search tree by using immediate selections.

It solved for the first time the famous 10 × 10 job-shop problem proposed by Muth and Thompson in 1963.

(JOB SHOP SCHEDULING; BRANCH-AND-BOUND; COMPUTATIONAL EXPERIMENTS)

Introduction

The significant literature (Balas 1969, Baker 1974, Rinnooy Kan 1976, Bouma 1982, Barker and McMahon 1985) on the job-shop problem begins with the book by Muth and Thompson *Industrial Scheduling*, (1963), in which three examples were proposed. So, for 20 years, authors have tested their algorithms on these examples. Two of them were solved in the 70s (Carlier 1978, McMahon and Florian 1975). Lageweg improved the best known solution to the third problem, but nobody has yet proved its optimality. In this paper, we describe a method demonstrating this.

Let us recall that, in a job-shop problem, n jobs have to be processed on m machines in order to minimize makespan. This problem is NP-hard in the strong sense (Rinnooy Kan 1976, Garey and Johnson 1979). So, authors propose branch and bound methods to solve it. Our method is based on one-machine problems and reconciles two conflicting objectives: optimizing complexity of local algorithms and minimizing memory space. The former is satisfied by information redundancy and the latter by a good choice of data structures. In this paper, we describe the method and its results.

1. Generalities

Job-Shop

In a job-shop problem, n jobs have to be processed on m machines assuming the following facts:

— A machine can process only one job at a time.
— The processing of a job on a machine is called an operation.
— An operation cannot be interrupted.
— A job consists of at most n operations.
— The processing order of a job is given according to this job.
— The operation sequence on the machines are unknown and have to be determined in order to minimize makespan.

Disjunctive Graph

Two operations i and j, executed by the same machine, cannot be simultaneously processed. So we associate with them a pair of disjunctive arcs $[i, j] = \{(i, j), (j, i)\}$.

* Accepted by Alexander H. G. Rinnooy Kan; received February 1987. This paper has been with the authors 6 months for 2 revisions.

0025-1909/89/3502/0164$01.25

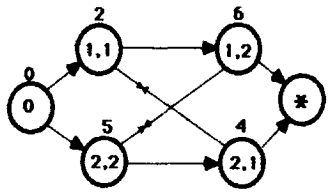

FIGURE 1. A 2 × 2 Job-Shop. Operation (a, b) is the processing of job a on machine b.

The problem is then modelled by a disjunctive graph $\mathcal{G} = (G, D)$, where $G = (X, U)$ is a conjunctive graph and D a set of disjunctions. Figures 1 and 2 show two examples.

Notations

In this paper, p_i denotes the processing time of operation i, r_i its release date, and q_i its tail. Denoting the value of one of the longest paths from i to j in G by $l(i, j)$, we have: $r_i = l(o, i)$ and $q_i = l(i, *) - p_i$ where o and $*$ are the source and the sink in G.

Schedule

A schedule on a disjunctive graph $\mathcal{G} = (G, D)$ is a set of starting times $T = \{t_i \mid i \in X\}$ such that:

— The conjunctive constraints are satisfied: $t_j - t_i \geq p_i \; \forall (i, j) \in U$;

— The disjunctive constraints are satisfied: $t_j - t_i \geq p_i$ or $t_i - t_j \geq p_j \; \forall [i, j] \in D$.

Selection

To build a schedule, we have to select the disjunctive constraints, and thus to choose an operating sequence for each machine.

A selection A is a set of disjunctive arcs such that if $(i, j) \in A$, then $(j, i) \notin A$. The membership of (i, j) in A makes it necessary to process operation i before operation j. We associate the conjunctive graph $G_A = (X, U \cup A)$ with the selection A.

By definition, a selection is complete if all the disjunctions of D are selected. It is consistent if the associated conjunctive graph is acyclic. A schedule corresponds to a consistent complete selection. Its makespan is the value of one of the longest paths in G_A.

For the first example, $A = \{((1, 1), (2, 1)), ((2, 2), (1, 2))\}$ is complete and consistent; the makespan of the associated schedule is 11 (Figure 3).

2. The One-Machine Problem

Introduction

One-machine problems are associated with a job-shop by choosing a machine and relaxing the constraints concerning the other machines. In this section, we unify and generalize results presented in two previous papers (Carlier 1978, 1982).

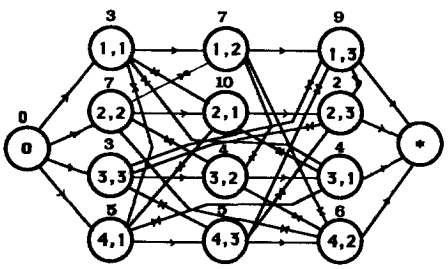

FIGURE 2. A 3 × 4 Job-Shop.

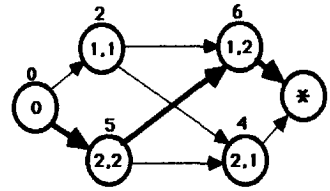

FIGURE 3. A Consistent Complete Selection.

2.1. *Definitions*

In the one-machine problem, we have to sequence a set I of dependent operations on a single machine in order to minimize makespan (McMahon and Florian 1975, Lageweg et al. 1976). An operation i has a release date r_i, a processing time p_i, and a tail q_i.

2.2. *Lower Bound*

Let K be a subset of I and $H(K)$ defined by:

$$H(K) = \text{Min}\,\{r_i \mid i \in K\} + \sum\{p_i \mid i \in K\} + \text{Min}\,\{q_i \mid i \in K\}.$$

PROPOSITION 1. *$H(K)$ is a lower bound of the optimal makespan of the one-machine problem.*

PROPOSITION 2. *Let V_k be the optimal value of the preemptive one-machine problem associated with machine k. $LB = \text{Max}\,\{V_k \mid k = 1, m\}$ is a lower bound of the makespan of the job-shop problem.*

In Carlier (1982), it is proved that V is equal to $\text{Max}\,\{H(K) \mid K \subseteq I\}$ and can be computed in $o(n \log n)$ steps.

2.3. *Jackson's Schedule*

Jackson's schedule is the list schedule associated with the MWR (Most Work Remaining) priority dispatching rule. To build it, we schedule, at the first moment t, where the machine and at least one operation are available, the available operation i with maximal q_i; then we set $t := t + p_i$ and iterate until all the operations are scheduled (Jackson 1955).

PROPOSITION 3. *Let f_o be the makespan of Jackson's schedule. There exists a subset J of I such that:*

— either $H(J) = f_o$ and Jackson's schedule is optimal,

— or $f_o - H(J) < p_c$ for some operation c of $I \backslash J$. J is called a critical set and c a critical operation for J.

For the proof, see Carlier (1982).

2.4. *Solution*

Let $\hat{f}$ be a given integer.[1] By definition, a solution of the one-machine problem is a schedule with a makespan smaller or equal than $\hat{f}$.

2.5. *Input and Output of a Clique*

A clique C is a subset of I that contains at least two operations. Let T be a solution. $e \in C$ (resp. $s \in C$) is called the input (resp. output) of clique C if e is sequenced in T

[1] Initially, $\hat{f}$ is either the best known solution or the lower bound of the makespan augmented by 5% or 10%. Later, $\hat{f} + 1$ is equal to the makespan of the best known schedule of the job-shop problem.

before (resp. after) all the other operations of C. $\hat{E}$ (resp. $\hat{S}$) is the subset of operations $i \in C$ such that there exists a solution T with i as input (resp. output) of C. E (resp. S) is a subset of I that contains $\hat{E}$ (resp. $\hat{S}$).

2.6. *Evaluation of $\hat{E}$ and $\hat{S}$*

Let $C \subseteq I$ and $k \in C$. We introduce conditions (1) and (2):

$$r_k + \sum \{p_j \mid j \in C\} + \text{Min}\{q_j \mid j \in S \setminus \{k\}\} > \hat{f}; \tag{1}$$

$$\text{Min}\{r_j \mid j \in E \setminus \{k\}\} + \sum \{p_j \mid j \in C\} + q_k > \hat{f}. \tag{2}$$

PROPOSITION 4. *If condition* (1) (*resp.* (2)) *is satisfied, then* $k \notin \hat{E}$ (*resp.* $\hat{S}$).

PROOF. Let us suppose that (1) is satisfied and $k \in \hat{E}$. The first member of the inequality (1) is a lower bound of the makespan when k is the input of C. So, this makespan is strictly larger than $\hat{f}$, and we obtain a contradiction. □

2.7. *Immediate Selection of a Disjunctive Constraint*

When C contains exactly two operations i and j, we obtain:

PROPOSITION 5. *If* $r_i + p_i + p_j + q_j > \hat{f}$, *then the disjunctive arc* (j, i) *will be selected in any solution.*

2.8. *Input and Output Determination*

Let $C \subseteq I$ and $k \in C$. We introduce condition (3):

$$H(C \setminus \{k\}) + p_k > \hat{f}. \tag{3}$$

LEMMA. *If* (3) *is satisfied, then, in any solution* T, *operation* k *is sequenced either before all the other operations of* C *or after them.*

Proof. Let T be a solution such that k is neither input nor output of C. Then, the operations of C are sequenced in some order $i_1, i_2, \ldots, i_h$ with $i_1 \neq k$ and $i_h \neq k$. So, the makespan of this solution is larger than: $r_{i_1} + \sum_{i \in C} p_i + q_{i_h}$. We have:

$$r_{i_1} + \sum_{i \in C} p_i + q_{i_h} \geqslant \text{Min}\{r_i \mid i \in C \setminus \{k\}\} + \sum \{p_i \mid i \in C \setminus \{k\}\} + p_k + \text{Min}\{q_i \mid i \in C \setminus \{k\}\} = H(C \setminus \{k\}) + p_k.$$

But this contradicts (3). □

PROPOSITION 6. *If conditions* (1) *and* (3) *are satisfied, then* k *is the output of clique* C *in any solution* T.

PROPOSITION 7. *If conditions* (2) *and* (3) *are satisfied, then* k *is the input of clique* C *in any solution* T.

PROPOSITION 8. *If* e *is the input of clique* C, *then the disjunctive arcs* (e, k) $(k \in C \setminus \{e\})$ *will be selected in any solution. If* s *is the output of clique* C, *then the disjunctive arcs* (k, s) $(k \in C \setminus \{s\})$ *will be selected in any solution.*

2.9. *Increase of Release Dates and Tails*

Let us recall that a conjunctive constraint associated with a release date r_i (resp. a tail q_i) is: $t_i - t_o \geqslant r_i$ (resp. $t_* - t_i \geqslant p_i + q_i$).

PROPOSITION 9. *If* s *is the output of clique* C, *then*:

$$t_s - t_o \geqslant \text{Min}\{r_j \mid j \in E\} + \sum \{p_j \mid j \in C \setminus \{s\}\}.$$

If e is the input of clique C, then:

$$t_* - t_e \geqslant \sum \{p_j \mid j \in C\} + \text{Min}\{q_j \mid j \in S\}.$$

PROPOSITION 10. *Let the operations of* $C \setminus \{s\}$ *be arranged in an ascending order of* r_i:

$$C \setminus \{s\} = \{i_1, i_2, \ldots, i_h\}; \qquad r_{i_1} \leqslant r_{i_2} \leqslant \cdots \leqslant r_{i_h}.$$

If s is the output of clique C, then:

$$t_s - t_o \geqslant \operatorname*{Max}_{P=1,h} [r_{i_P} + \sum \{p_{ij} \mid j = P, h\}].$$

PROOF. s is the output of clique $C_P = \{i_P, i_{P+1}, \ldots, i_h, s\}$. □

PROPOSITION 11. *Let the operations of* $C \setminus \{e\}$ *be arranged in an ascending order of* q_j:

$$C \setminus \{e\} = \{j_1, j_2, \ldots, j_h\}; \qquad q_{j_1} \leqslant q_{j_2} \leqslant \cdots \leqslant q_{j_h}.$$

If e is the input of clique C, then:

$$t_* - t_e \geqslant \operatorname*{Max}_{P=1,h} [\sum \{p_{j_i} \mid i = P, h\} + p_e + q_{j_P}].$$

PROOF. e is the input of clique $C'_P = \{e, j_P, j_{P+1}, \ldots, j_h\}$. □

REMARK. Propositions 10 and 11 are more general than Proposition 9, but the corresponding test is more expensive. Our experience is that these two propositions do not significantly improve the method.

PROPOSITION 12. *If* $i \in C \setminus E$, *then*: $t_i - t_o \geqslant \text{Min}\{r_e + p_e \mid e \in E\}$.
If $i \in C \setminus S$, *then*: $t_* - t_i \geqslant \text{Min}\{p_s + q_s \mid s \in S\} + p_i$.

3. A Branch and Bound Method for the Job-Shop

3.1. *Branching Scheme*

A selection A is associated with each node N of the search tree. To branch, we choose a disjunction $[i, j]$ and we introduce two nodes with selections $A \cup \{(i, j)\}$ and $A \cup \{(j, i)\}$.

3.2. *Bounds*

The upper bound is equal to the value of the best known schedule of the job-shop. The lower bound $F(N)$ for node N is obtained by application of Proposition 2.

3.3. *Computing of R and Q*

With node N is associated a conjunctive graph $G = (X, U \cup A)$. The release dates vector R (resp. tails vector Q) is computed by solving a longest path problem by applying Bellman's algorithm on this graph: $r_i = l(o, i)$ (resp. $q_i = l(i, *) - p_i$).

3.4. *The Boolean Active*

With the tree is associated a boolean Active, which is initialized to False. This boolean takes the value True every time a proposition is efficient. For instance, Propositions 5 and 8 are efficient if they permit to select a disjunctive constraint. Proposition 4 is efficient if it improves the evaluation E of $\hat{E}$ (resp. S of $\hat{S}$). Finally, Propositions 9, 10, 11 and 12 are efficient if they modify one release date or one tail.

3.5. *Treatments before Branching*

We successively consider the m one-machine problems associated with the job-shop problem. For each one, we compute the optimal preemptive schedule, a clique C and evaluations E (resp. S) of $\hat{E}$ (resp. $\hat{S}$) (Proposition 4).

Propositions 6 and 7 were only applied for the critical set J and the critical operation c given by Jackson's schedule. Propositions 5 and 8 permit the selection of disjunctive constraints. Finally, the evaluations of $t_i - t_o$ and $t_* - t_i$ by application of Propositions 9, 10, 11, and 12 allow the modification of vectors R and Q.

REMARK. Initially, the clique C, associated with a one-machine problem, is the whole set I. Then, progressively with the determination of inputs and outputs of C, we adjust this clique by setting up:

$C := C \setminus \{e\}$ when an input is found;

$C := C \setminus \{s\}$ when an output is found.

3.6. *Choice of a Disjunction*

The branching scheme is based on the following remarks. When the cardinality of E (resp. S) becomes 0 or 1, we can truncate the search tree and apply very efficiently the previous propositions. Moreover, if we select a disjunction involving two operations of a set E (resp. S), the cardinality of E (resp. S) will strictly decrease for the nodes created in the search tree.

We found it better to select as a priority the disjunctions of the machine with the largest initial lower bound, called the critical machine and indexed 1.

So, we propose the following heuristic method:

1—If the critical machine is not completely selected, choose a disjunction in the set E_1 or S_1 with minimal cardinality.

2—Otherwise, choose a disjunction belonging to a set E_r or S_r ($r \neq 1$) with minimal cardinality.

In case of several candidates, we use a penalty function and compute the quantities (Bertier and Roy 1965):

— $d_{ij} = \text{Max}\,(0, r_i + p_i + p_j + q_j - LB)$;

— $d_{ji} = \text{Max}\,(0, r_j + p_j + p_i + q_i - LB)$;

— $a_{ij} = \text{Min}\,(d_{ij}, d_{ji})$;

— $v_{ij} = |\, d_{ij} - d_{ji} \,|$.

Then we choose the disjunction with maximal v_{ij}, or, in case of ties, the one with maximal a_{ij}.

REMARK. A second branching scheme was also tested with no great success up to now. We computed Jackson's solution for the critical machine, and determined the critical set J and operation c (whenever the latter exists). If $H(J) + p_c > \hat{f}$, we considered, according to the previous lemma, both the following problems:

— $c \rightarrow J$ where operation c is processed before J;

— $J \rightarrow c$ where operation c is processed after J.

3.7. *General Algorithm*

```
Begin
  Read the problem data ; initialize data structures ;
  N:=1 ; f̂:=+∞ ; F(1):=0 ;
  While the search tree is not empty
    Begin
      While F(N)> f̂ Do
        Begin
          Suppress node N ;
```

```
        N:=N-1
      End
      Restore node N ;
    Active:=True
    While Active=True Do
      Begin
        Active:=False ;
        Execute Treatments before branching ;
        If Active=True Compute the vectors R and Q ;
      End
    Compute the new lower bound F(N) ;
    If the selection is complete then
      Begin
        Print the solution ;
        Compute the value of this solution : f1 ;
        f̂ := Min(f̂,f1-1) ;
        N:=N-1
      End
    Else
      Begin
        Apply the branching heuristic ;
        Create the new nodes corresponding to the
        selected disjunction ;
        N:=N+1
      End
    Endif
  End
End
```

3.8. *Discussion*

We expected Propositions 6 and 7 to be very powerful if applied to every subset C of I and to every operation k of C. So, we looked for an efficient algorithm to reach this goal. We found an $o(n^4)$ algorithm and used it. Generally, the size of the search tree was divided by a factor of 2, but the algorithm was too time consuming. But a very interesting property appeared: when the sequence is given for a machine, the two propositions permit the selection of nearly all the other disjunctions. So, it provides a good method to build heuristic schedules. At present, we are working on a better algorithm that detects in $o(n^2)$ the useful couples (k, C) satisfying the conditions of Propositions 6 and 7.

3.9. *Example*

Let us explain the branch and bound method on the job-shop problem of Figure 2. First we initialize $\hat{f}$ by $\hat{f} = 26$ (the initial lower bound augmented of 10%) and we apply the propositions to the one machine problem of machine 2. If (1, 2) was the input of clique $C_2 = I$, the makespan would be larger than: $r_{12} + \Sigma \{p_i \mid i \in I\} + \text{Min} \{q_i \mid i \in I\} = 3 + 24 = 27$ (Proposition 4). This is impossible because $\hat{f} = 26$. For the same reason, (3, 2) (resp. (4, 2)) cannot be the input of C_2. So, we select the disjunctive arcs: ((2, 2), (1, 2)), ((2, 2), (3, 2)), ((2, 2), (4, 2)) and set: $C_2 := C_2 \setminus \{(2, 2)\}$. By applying Proposition 4, we obtain $S_2 = \{(4, 2)\}$. So, with $C_2 := C_2 \setminus \{(4, 2)\}$ and by applying Proposition 5, machine 2 is entirely selected (Figure 4). Then Proposition 5 permits the selection of the arcs ((1, 3), (2, 3)), ((3, 3), (2, 3)), and ((4, 3), (2, 3)). So, (2, 3) is the output of machine 3. We set: $C_3 := C_3 \setminus \{(2, 3)\}$, apply Proposition 5 to C_3 and get $S_3 = \{(1, 3)\}$. So, we obtain the conjunctive graph of Figure 5.

Next, we compute the new values of R and Q. Consequently, (3, 1) is the output of machine 1 and (2, 1) is the output of $C_1 \setminus \{(3, 1)\}$. Propositions are no longer useful.

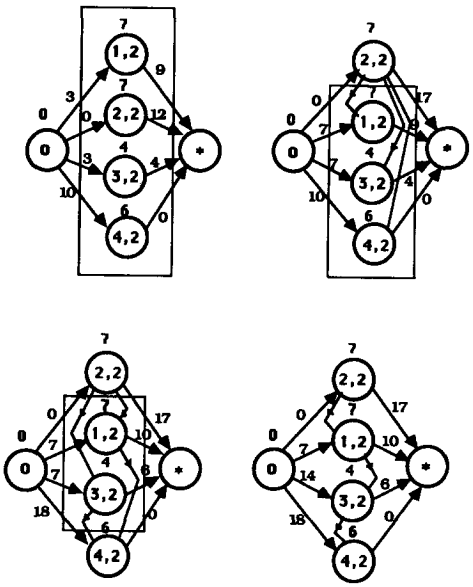

FIGURE 4. Selection of Machine 2. (For clarity, disjunctions are omitted.)

Figure 6 shows the resulting conjunctive graph, and Figure 7 the two disjunctions left. The tree is reported in Figure 8.

4. Data Structures

4.1. *The Search Tree*

The current tree is a path $(1, 2, \ldots, N)$. It is coded with a stack in order to quickly restore a node. To a node M corresponds a block of this stack containing the number d_M of the disjunction that creates it and the set D_M of disjunctions locally selected with Propositions 5 and 8. (See Figure 9.) The maximal size of this stack is the number nds of disjunctions. The lower bounds are in a table indexed by M.

4.2. *The Disjunctions*

In order to determine immediate selection, we associate with node N a table DIS defined by:

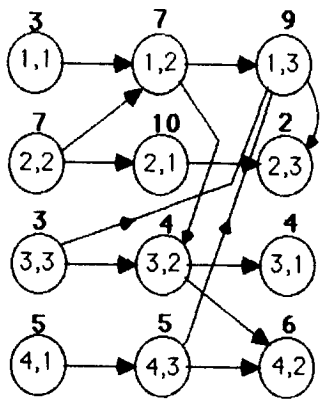

FIGURE 5. Conjunctive Graph after Selection on Machines 2 and 3.

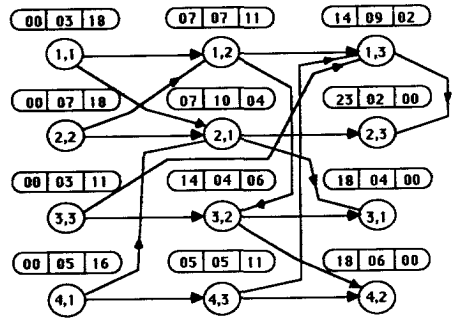

FIGURE 6. Conjunctive Graph.

$DIS(k) = M$ (resp. $-M$) if the disjunction k is selected in the direct (resp. reverse) sense at node M;

$DIS(k) = 0$ if it is not selected.

4.3. *The Conjunctive Graph*

The graph of node N is redundantly coded by the successors file and the predecessors file. These files are treated as a set of stacks: when we suppress (resp. add) a conjunctive arc (i, j), we decrease (resp. increase) by one the number of successors of i and the number of predecessors of j.

4.4. *Release Dates and Tails*

With node N are associated a couple of vectors R and Q. These vectors are computed by Bellman's algorithm.

4.5. *Cliques*

To save memory spaces, the sets C_r, E_r, and S_r $(r = 1, m)$ are only stored for the node N. It may be very costly to compute these sets after a backtracking, by applying the propositions; so, we introduce, for each machine, a table ES defined by (i is an operation of the corresponding machine):

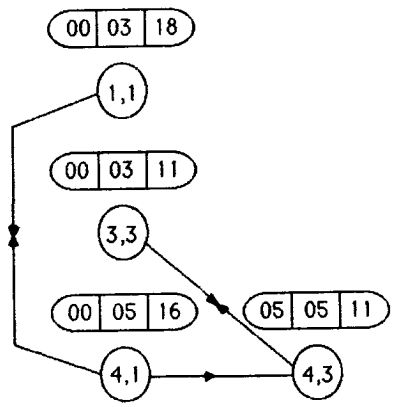

FIGURE 7. Disjunctions Left.

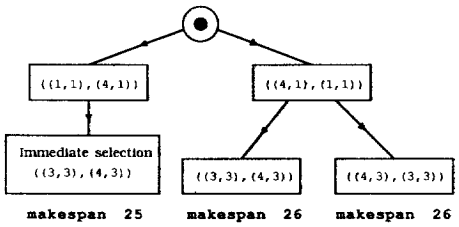

FIGURE 8. Search Tree.

— $ES(i) = 1$ if $i \in E \cap S$;
— $ES(i) = 2$ if $i \in E \setminus S$;
— $ES(i) = 3$ if $i \in S \setminus E$;
— $ES(i) = 0$ if $i \in C \setminus E \cup S$;
— $ES(i) = -1$ if i was determined as input;
— $ES(i) = -2$ if i was determined as output.

Proposition 13 shows that the number of modifications of ES for a machine with n operations is very small in comparison with the cost of the storage of the sets at every node of the tree. We keep in a stack the modifications of ES in order to restore ES when we backtrack.

PROPOSITION 13. *The maximal number of modifications of ES along a path of the tree for a clique of cardinal n is strictly smaller than*: $(3/2)n(n+1)$.

PROOF. When an input or an output is determined, the cardinality of the clique C decrease by one and before such a determination, the number of modifications is smaller than $3|C|$ (Figure 10). So, along a path of the tree, it is less than:

$$\sum 3\{n - i \mid i = 0, n-2\} < (3/2)n(n+1). \qquad \square$$

5. Computational Experiments

We have implemented the algorithm on a mini-computer PR1ME 2655 (1.3 Mips) in FORTRAN77, and tested it on about 150 bench marks, some of them coming from the literature (Muth and Thompson 1963, Carlier 1975), the others randomly generated.

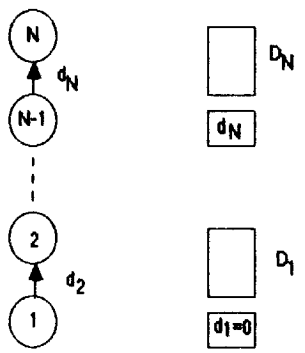

FIGURE 9. Search Tree and Corresponding Stack.

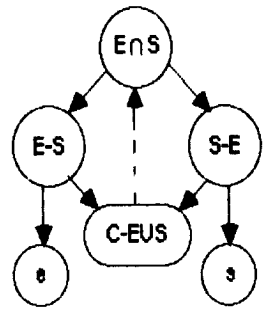

FIGURE 10. States of Vertex i.

We report the following characteristics of some experiments (see Table 1):
— n: number of jobs,
— m: number of machines,
— nds: number of disjunctions of the problem,
— f^*: optimum value,
— LB: lower bound on the search tree root,
— $t1$: CPU time used for finding the optimal schedule,[2]
— $nd1$: number of branching used for the optimum's search,
— $t2$: CPU time used for the optimality control,[3]
— $nd2$: number of branching used for the optimality control,
— tt: total CPU time,
— ndt: total number of branchings.

As a referee pointed out, the practical complexity of the 10×10 instance is probably due to the gap of 15% between the initial lower bound and the value of an optimal schedule.

We also tested this method on problems with 50 jobs and 10 machines. In this case, we always obtain schedules near 5% of the optimum within 20 mn of CPU.

We used it as well for scheduling optimally a flexible work-shop with 4 machines and a daily production of about 50 products.

TABLE 1

	n	m	nds	f^*	LB	$t1$	$nd1$	$t2$	$nd2$	tt	ndt	Up[a]
1	6	6	90	55	52	1	1	0	0	1	1	55
2	7	6	126	63	62	12	17	13	19	25	36	65
3	8	8	224	6309	6213	47	52	51	63	98	115	6523
4	11	5	275	7038	6917	40	47	1	0	41	47	6263
5	9	9	324	4223	4016	211	257	432	503	643	760	4417
6	12	5	330	7312	7103	13	25	1	0	14	25	7458
7	14	4	364	8003	8003	62	55	2	1	64	56	8403
8	10	10	450	930	808	3305	4039	14680	17982	17985	22021	969
9	15	5	525	5422	5223	2315	2719	4023	5012	6338	7731	5484
10	20	5	950	1165	1164	1234	1609	214	253	1448	1862	1222

[a] Up: upper bound used at the beginning of the treatment.

[2] CPU time in seconds on PR1ME 2655.

[3] We separated the search of the optimum from the optimality control, the latter corresponding, for most bench marks, to the largest consumption of CPU time. For instance, for the 10×10 job-shop, the optimum search takes 53 mn and the optimality control 4h05.

Glossary

i, j	operations
I	set of operations on a given machine
$C \subseteq I$	clique of disjunction
$K \subseteq I$	clique of disjunction
solution	one-machine schedule of makespan less or equal than $\hat{f}$
$\hat{f}$	upper bound
e	input of C
s	output of C
$\hat{E}$	inputs set of C
$\hat{S}$	outputs set of C
E	over set of $\hat{E}$
S	over set of $\hat{S}$
J	critical set (Jackson)
c	critical operation (Jackson)
N	number of a terminal node in the search tree
M	number of a node in the search tree
n	number of jobs
m	number of machines
nds	number of disjunctions in the problem
r_i	release date of operation i
p_i	processing time of operation i
q_i	tail of operation i
$G = (X, U)$	initial conjunctive graph
o	source of G
$*$	sink of G
A	selection
$G = (X, U \cup A)$	conjunctive graph associated with selection A
C_r	clique associated with machine r
$\hat{E}_r, \hat{S}_r$	inputs and outputs sets of C_r
$H(K)$	lower bound for clique K
$F(N)$	lower bound for node N
$l(i, j)$	value of a longest path from i to j in G
T	schedule

Bibliography

ADAMS, J., E. BALAS AND D. ZAWACK, "The Shifting Bottleneck Procedure for Job-Shop Scheduling," Management Science Research Report No. MSRR-525, 1986.

BALAS, E., "Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm," *Oper. Res.*, 17 (1969), 941–957.

BAKER, K. R., *Introduction to Sequencing and Scheduling*, Wiley, New York, 1974.

BARKER, J. R. AND G. B. MCMAHON, "Scheduling the General Job-Shop," *Management Sci.*, 31, 5 (1985).

BELLMAN, R. E., "On a routing problem," *Quart. J. Appl. Math.*, 16 (1958), 87–90.

BERTIER, P. AND B. ROY, "Trois exemples numeriques d'application de la procedure SEP," Note de travail n°32 de la Direction Scientifique de la SEMA, 1965.

BOUMA, R. W., "Job-Shop Scheduling: A Comparison of Three Enumeration Schemes in a Branch and Bound Approach," Master's thesis. Erasmus University Rotterdam, Faculty of Econometrics and Operations Research, 1982.

CARLIER, J., "Ordonnancements à contraintes disjonctives," Thèse de 3ème cycle, 1975.

———, "Ordonnancements à contraintes disjonctives," *RAIRO*, 12 (1978), 333–351.

———, "One Machine Problem." *European J. Oper. Res.*, 11 (1982), 42–47.

———, "Problèmes d'ordonnancements à contraintes de ressources: algorithmes et complexite," Thèse d'état, 1984.

FRENCH, S., *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, Wiley, New York, 1982.

GAREY, M. R. AND D. S. JOHNSON, *Computers and Intractability*, W. H. Freeman and Co., 1979.

GRABOWSKI, J., "A New Algorithm of Solving the Flow-Shop Problem," *Oper. Res.*, (1982), 57–75.

HANSEN, P., Private communication.

JACKSON, J. R., "Scheduling a Production Line to Minimize Maximum Tardiness," Research report 43, Management Science Research Project, University of California, Los Angeles, 1955.

LAGEWEG, B. J., J. K. LENSTRA AND A. H. G. RINNOOY KAN, "Minimizing Maximum Lateness on One Machine: Computational Experiences and Some Applications," *Statist. Neerlandica*, 30 (1976), 25–41.

———, ——— AND ———, "A General Bounding Scheme for the Permutation Flow-Shop Problem," *Oper. Res.*, 26 (1978), 53–67.

LENSTRA, J. K., *Sequencing by Enumerative Methods*, Mathematisch Centrum, Amsterdam, 1976.

———, A. H. G. RINNOOY KAN AND P. BRUCKER, "Complexity of Machine Scheduling Problems," *Ann. Discrete Math.*, 1 (1977), 343–362.

G. MCMAHON AND M. FLORIAN, "On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness," *Oper. Res.*, 23, 3 (1975), 475–482.

MUTH, J. F. AND G. L. THOMPSON, *Industrial Scheduling*, Prentice Hall, Englewood Cliffs, NJ, 1963.

POTTS, C. N., "An Adaptative Branching Rule for the Permutation Flow-Shop Problem," *European J. Oper. Res.*, 5 (1980), 19–25.

RINNOOY KAN, A. H. G., *Machine Scheduling Problems: Classification, Complexity and Computations*, Nijhoff, The Hague, 1976.

ROY, B. AND B. SUSSMAN, "Les problèmes d'ordonnancements avec contraintes disjonctives," Notes DS n°9 bis, SEMA, Paris, 1964.