RAIRO. RECHERCHE OPÉRATIONNELLE

MOHAMED HAOUARI
THOURAYA DAOUAS

Optimal scheduling of the 3-machine assembly-type flow shop

RAIRO. Recherche opérationnelle, tome 33, n° 4 (1999), p. 439-445

<http://www.numdam.org/item?id=RO_1999__33_4_439_0>

RAIRO Operations Research
RAIRO Oper. Res. **33** (1999) 439-445

OPTIMAL SCHEDULING OF THE 3-MACHINE ASSEMBLY-TYPE FLOW SHOP (*)

by Mohamed HAOUARI ([1]) and Thouraya DAOUAS ([2])

Communicated by Bernard LEMAIRE

Abstract. – *We address the 3-Machine Assembly-Type Flowshop Scheduling Problem (3MAF). This problem is known to be NP-complete in the strong sense. We propose an exact branch and bound method based on a recursive enumeration of potential inputs and outputs of the machines. Using this algorithm, several large size instances have been solved to optimality.*

Keywords: Scheduling, flow shop, branch and bound, assembly-type production.

Résumé. – *Nous nous intéressons au problème d'ordonnancement de 3 machines dans un atelier d'assemblage de type flowshop. Ce problème est connu comme étant NP-complet au sens fort. Nous proposons un algorithme exact de séparation et évaluation progressives, basé sur une énumération récursive des entrées et des sorties potentielles des machines. Des problèmes de grande taille ont été résolus d'une manière optimale. À notre connaissance, il s'agit de la première fois, où ce problème est résolu exactement.*

Mots clés : Ordonnancement, flow shop, séparation et évaluation progressives, production de type assemblage.

1. INTRODUCTION

In this note, we consider the following scheduling problem. There is a set of three machines $M = \{M_1, M_2, M_3\}$ that are continuously available for processing a set of n independent jobs $J = \{J_1, J_2, ..., J_n\}$. Each job J_j consists of three operations (or tasks) O_{j1}, O_{j2}, O_{j3} which must be processed on machines M_1, M_2, and M_3 respectively. The processing of job J_j $(j = 1, 2, ..., n)$ on machine M_m $(m = 1, 2, 3)$ requires p_{jm} time units. For any job J_j $(j = 1, 2, ..., n)$ operation O_{j3} can not begin before complete processing of both operations O_{j1} and O_{j2}. Each machine can process no

(*) Received June 1996.
([1]) École Polytechnique de Tunisie, BP. 743, 2078 La Marsa, Tunisia.
([2]) Institut des Hautes Études Commerciales, 2016 CarthagePresidence, Tunisia.

Recherche opérationnelle/Operations Research, 0399-0559/99/04/$ 7.00

more than one operation at a time. Furthermore, once an operation has begun processing it can not be interrupted. Such shops are common in manufacturing settings where component parts and fabrication parts are fed into a common assembly line for final assembly operation.

Let $C_{\max}(\sigma)$ denote the maximum length of the schedule σ which is defined as the total time to complete all jobs (*i.e.* the makespan) under σ. The objective is to find an optimal schedule σ^* among the set of feasible schedules that minimizes the makespan, *i.e.* $C_{\max}(\sigma^*) \leq C_{\max}(\sigma)$ for all σ. This problem is referred to as the 3-machine assembly-type flow shop problem (3MAF).

Whereas almost all past research, dealing with deterministic multi-operation scheduling models, has been focused on serial-type manufacturing systems, assembly-type systems received scant attention (see for example Lawler *et al.* 1993, Gotha 1993). The assembly-type flow shop problem has been first introduced by Lee *et al.* (1993). They proved that the 3MAF is NP-complete in the strong sense. They also suggested a branch and bound solution scheme, and proposed three heuristics and analyzed their error bounds. In a recent paper, Potts *et al.* (1995) generalized the 3MAF problem by considering m machines at the first stage, and one assembly machine in the second stage. Heuristics with ratio and absolute performance guarantees have been presented. Other recent papers dealing with similar two-stage flow shop scheduling problems include the two- stage flow shop scheduling with a common second stage machine (Oguz and Cheng 1995), and the two-stage hybrid flow shop problem (Haouari and M'hallah 1997).

2. PRELIMINARY RESULTS

We define the inverse of 3MAF, as a two-stage scheduling problem with one machine in the first stage, and two machines in the second one. The jobs are processed first on the unique first stage machine, and then on stage 2 machines. This latter problem is referred to as the 3-machine dismantling problem (3MDF). The following result shows that the 3MAF and the 3MDF are equivalent.

Observation 1

An optimal schedule for 3MAF, can be reversed to obtain an optimal schedule for 3MDF, with the same makespan.

Proof: If 0 and $C_{\max}$ are the ends of an optimal feasible schedule for 3MAF, then we can perform a "mirror transformation" of this schedule with

respect to the vertical line $1/2C_{\max}$; we obtain a feasible schedule for the 3MDF with the same makespan.

Thus, 3MAF and 3MDF are equivalent problems. We restrict ourselves from here on to the description of an exact branch and bound method for solving the 3MDF problem.

Before going into the description of the algorithm we make the following second observation.

Observation 2

There is an optimal schedule in which all machines have the same sequence.

Proof: Let σ_3 be the job sequence of machine M_3. Clearly, an optimal sequencing of machine $M_1(M_2)$ can be obtained by minimizing the makespan on a single machine with release dates $(1/r_j/C_{\max})$. The release date r_j of job j is equal to the completion time of job j on machine M_3. An optimal sequencing of machine $M_1(M_2)$ can be found by scheduling the jobs in order of non-decreasing release dates. The optimality of this rule can be easily checked by a simple interchange argument. Thus, the job sequences of machines M_3 and $M_1(M_2)$ are identical.

The consequence of observation 2, is that we need only consider permutation schedules.

3. THE BRANCH AND BOUND ALGORITHM

3.1. Definitions

3MDF being NP-hard, we propose in this section a branch and bound method for finding an optimal permutation schedule. Our algorithm is an adaptation of the branch and bound algorithm developped by Carlier and Rebaï (1996) for the permutation flow shop problem. For that purpose, we define three sets of jobs at each node of the branch and bound search tree:

- a set "I" of jobs fixed at the beginning of the sequence. We call them *inputs of machines* or *inputs* since the sequence is identical for all machines;
- a set "OUT" of jobs fixed jobs at the end of the sequence. These jobs are called *outputs of machines* or *outputs*.
- Finally, a set "DIS" of jobs not yet fixed and thus still in disjunction.

We define also for each machine three vectors:

- R: a 3-vector, whose m^{th} component gives for machine m, the date when it becomes idle and can process one of the jobs of DIS.
- Q: a 3-vector, whose m^{th} component gives for machine m, the time required for processing all the jobs of OUT.
- SUM_P: a 3-vector, whose m^{th} component gives for each machine m, the sum of processing times of the unscheduled jobs (*i.e.* those belonging to DIS).

At any node of the search tree, jobs are partitioned into three sets: IN, OUT, and DIS.

3.2. Branching and bounding scheme

Two blocks of jobs are constructed: one at the beginning of the sequence and one at its end. Branching consists of alternatively adding a job to IN and OUT, until only one job remains in DIS. If at a given level of the search, a new input (output) has been fixed, then at the next level an output (input) will be fixed. We now explain how to adjust vectors R, Q, and SUM_P.

For convenience, machine M_1 denotes the first stage machine and $\{M_2, M_3\}$ denote stage 2 machines.

- Suppose a job J_j is fixed as a new input. Thus, a new node is created in the search tree. Consequently, both R and SUM_P are adjusted, and Q remains unchanged. We have:
  $R(1) = R(1) + p_{j1}$ and $SUM\_P(1) = SUM\_P(1) - p_{j1}$
  $R(m) = \max(R(1), R(m)) + p_{jm}$ and $SUM\_P(m) = SUM\_P(m) - p_{jm}$ $(m = 2$ and $3)$.
- Suppose a job J_j is fixed as a new output. Both Q and SUM_P are adjusted, and R remains unchanged. This yields:
  $Q(m) = Q(m) + p_{jm}$ and $SUM\_P(m) = SUM\_P(m) - p_{jm}$ $(m = 2$ and $3)$
  $Q(1) = \max\{Q(1), Q(2), Q(3)\}$ and $SUM\_P(1) = SUM\_P(1) - p_{j1}$.

The lower bound is given by:

$$LB = \max_{m=1,2,3} \{R(m) + SUM\_P(m) + Q(m)\}.$$

It is noteworthy that at any node of the search tree (except at the root node), computing R, Q, and SUM_P can be done in $O(1)$ time. Thus, computing LB can be done in $O(1)$ as well.

3.3. Upper bounds

Similarly to Lee *et al.* (1993), we propose to obtain an approximate solution to 3MDF by applying Johnson's algorithm (1954) to an instance of a two-machine flowshop problem defined on machine M_1 and a dummy machine M^*. Indeed, three different instances of $F2//C_{\max}$ are solved. The processing times p_{j^*} $(j = 1, 2, ..., n)$ on machine M^* are, respectively, set equal to:

- $p_{j^*} = \max(p_{j2}, p_{j3})$, for $j = 1, 2, ..., n$
- $p_{j^*} = 1/2(p_{j2} + p_{j3})$, for $j = 1, 2, ..., n$
- $p_{j^*} = p_{j2}$, for $j = 1, 2, ..., n$ if $\Sigma_j p_{j2} \geq \Sigma_j p_{j3}$, otherwise $p_{j^*} = p_{j3}$, for $j = 1, 2, ..., n$.

Each optimal solution to $F2//C_{\max}$ is an approximate schedule for 3MDF. We propose to take as an upper bound the value of the least 3MDF makespan.

3.4. The algorithm

We define a node k of the search tree as a data structure containing mainly the current lower bound $LB(k)$, the current upper bound UB(k), sets IN, DIS, and OUT, vectors R, SUM_P, and Q, and finally a boolean variable $last\_fixed$ defining whether in the node k, from which we move to generate node k', an input $(last\_fixed = 1)$ or an output $(last\_fixed) = 0$ will be fixed.

0. Initialization:
   $k = 1$, last-fixed $= 0$, $R(1) = 0$, $Q(1) = \text{Min}_{j=1,n}\{\text{Max}(p_{j2}, p_{j3})\}$, $R(2) = R(3) = \min_j\{P_{j1}\}$, $Q(2) = Q(3) = 0$, $SUMP(m) = \Sigma_j p_{jm}$, DIS $= J$, IN $=$ OUT $= \varnothing$, compute LB.
1. Assign to UB the value of the makespan obtained by the heuristic described in Section 3.3.
2. While $(LB(k) > UB$ and $k > 0)$, $k = k - 1$ (*i.e.* delete node k).
3. If $k = 0$ go to step 11.
4. Find the node k' having the same predecessor than k and with lowest LB. Exchange k and k'.
5. Read vectors R, SUM_P, and Q as well as DIS, IN, OUT, last-fixed and LB corresponding to node k.
6. $k = k - 1$.
7. Last-fixed $= 1 -$ last-fixed.

8. If $|\text{DIS}| = 1$ a complete schedule is obtained, then compute the makespan.
   If the makespan is less than UB, then update UB. Else:
9. Build new nodes: for all jobs J_j of DIS and according to last- fixed
   - Compute the new values of R, SUM_P, and Q.
   - If $LB = \text{Max}_{m=1,2,3}(R(m) + SUMP(m) + Q(m)) < \text{UB}$, then:
     - $k = k + 1$.
     - Copy these vectors and these values in node k and update IN or OUT (according to last-fixed).
     - Apply the heuristic to the set of jobs in DIS, and let S denote the resulting approximate schedule. Compute $\text{UB}(k)$ as the makespan of the schedule IN-S-OUT. $\text{UB}(k) < \text{UB}$, then update UB.
10. Go to step 2.
11. End.

4. COMPUTATIONAL RESULTS

The branch and bound algorithm described above has been coded in C language and implemented on a Pentium 200 MHz PC. For each problem size, 10 different instances have been randomly generated according to the uniform distribution U(1,100). The results are summarized in Table 1. We define:

n: total number of jobs.

NN: average number of developped nodes.

NN_{max}: maximum number of developped nodes.

t: average CPU time (in seconds) to prove the optimality.

Computational results show that relatively large sized two-stage assembly-type flow shop problems can be solved exactly in a reasonable amount of time. To the best of our knowledge, this is the first time that this NP-complete problem has been solved exactly.

There are at least two issues that are worthy of future investigations. First, as pointed out by Lee *et al.* (1993), the challenge would be to extend such approach to solve scheduling problems arising in multistage assembly systems which are more realistic in practice. Second, consideration of other optimization criteria, like minimization of maximum lateness or total flow time, would be useful for practical applications.

TABLE 1
Computational results.

n	NN	NN_{max}	t
20	28	31	~0
40	159	249	1
60	184	252	1
80	316	370	2
100	581	663	4
120	824	908	6
140	1031	1310	6
160	1424	1620	11
180	1686	1713	17
200	1904	2285	21

REFERENCES

J. CARLIER and I. REBAÏ, Two Branch and Bound algorithms for the Permutation Flow Shop Problem, *European J. Oper. Res.*, 1996, *90*, n° 2, p. 238-251.

M. HAOUARI and R. M'HALLAH, A Heuristic for the Two-Stage Hybrid flow Shop Problem, *Oper. Res. Lett.*, 1997, *21*, n° 1, p. 43-53.

S. M. JOHNSON, Optimal Two and Three-Stage Production Schedule with Setup Times Included, *Naval Res. Logist. Quart.*, 1954, *1*, p. 61-68.

GOTHA, Les problèmes d'ordonnancement, *RAIRO-Oper. Res.*, 1993, *27*, p. 77-150.

E. L. LAWLER, J. K. LENSTRA, A. H. G. RINNOOY KAN and D. B. SHMOYS, Sequencing and Scheduling: Algorithms and Complexity, in: *Operations Research and Management Science*, *Vol. 4*: Logistics of Production and Inventory, North-Holland, Amsterdam, 1993, p. 445-522.

C. Y. LEE, T. C. E. CHENG and B. M. T. LIN, Minimizing the makespan in the 3-Machine Assembly-Type Flowshop Scheduling Problem, *Management Sci.*, 1993, *39*, p. 616-625.

C. OGUZ and T. C. E. CHENG, Two-stage Flowshop Scheduling with a Common Second-Stage Machine, *INFORMS Meeting*, Los Angeles, May 1995.

C. N. POTTS, S. V. SEVAST'JANOV, V. A. STRUSEVICH, L. N. VAN WASSENHOVE and C. M. ZWANEVALD, The Two-Stage Assembly Scheduling Problem: Complexity and Approximation, *Oper. Res.*, 1995, *43*, p. 346-355.