*J. Opl Res. Soc.* Vol. 39, No. 4, pp. 359–364, 1988
Printed in Great Britain. All rights reserved

0160-5682/88 \$3.00 + 0.00
Copyright © 1988 Operational Research Society Ltd

# Two-Stage, Hybrid Flowshop Scheduling Problem

JATINDER N. D. GUPTA
Department of Management Science, Ball State University, Indiana, USA

This paper describes the two-stage flowshop problem when there are identical multiple machines at each stage, and shows that the problem is NP-complete. An efficient heuristic algorithm is developed for finding an approximate solution of a special case when there is only one machine at stage 2. The effectiveness of the proposed heuristic algorithm in finding a minimum makespan schedule is empirically evaluated and found to increase with the increase in the number of jobs.

*Key words*: heuristics, production scheduling, sequencing

## INTRODUCTION

Consider the following two-stage, hybrid scheduling problem: there are $n$ jobs to be processed on two stages, each job being processed first on stage 1 and then on stage 2. At stage $s$, there are $m_s$ number of identical machines. It is desired to find a minimum makespan schedule for this problem. As a secondary objective, it is desired to minimize the number of machines used at each stage. If $m_s = 1$ for both stages, this problem can be solved efficiently by using Johnson's algorithm.[1] If $m_s > 1$ for any $s$, the problem becomes rather difficult to solve.

When there is only one machine at stage 2, the branch-and-bound approach of Arthanary and Ramaswamy[2] for any value of $m_1$ and that of Mittal and Bagga[3] for $m_1 = 2$ can be used to solve the problem. However, as shown by Murty,[4] Mittal and Bagga's procedure does not always generate an optimal schedule and hence must be considered a heuristic procedure. Both procedures require excessive computational time to solve the problem. Rao[5] describes a logical approach to solve this problem when only one of the two stages contains multiple machines. However, his approach is difficult to implement on a computer, and requires too many computations. In addition, his procedure generates schedules that are too far away from minimum makespan schedules for the case when there is only one machine at stage 2.

This paper discusses the complexity of the two-stage, hybrid flowshop scheduling problem, and proposes an efficient heuristic algorithm to solve a special case when there is only one machine at stage 2. The effectiveness of the proposed heuristic algorithm in finding a minimum makespan schedule is empirically evaluated and found to increase with the increase in the number of jobs.

## PROBLEM COMPLEXITY

The following theorem shows the NP-completeness of the problem.

*Theorem 1*

For max $(m_1, m_2) > 1$, the two-stage, hybrid flowshop problem is NP-complete even if the number of machines at one of the two stages is one.

*Proof*

Assume that $m_1 \leqslant m_2$ and that processing times for stage 1 are all zero. Then the problem is one of scheduling $n$ jobs on $m_2$ number of identical parallel machines if no pre-emptions are allowed. This problem is equivalent to solving an $m_2$-partition problem which is NP-complete.[6,7] A similar conclusion can be reached if $m_1 > m_2$. Thus, the restricted problem with zero processing times of all jobs on any one of the two stages is NP-complete. Therefore, following Garey and Johnson,[6] the original problem is also NP-complete. The above result is equally true even if one of the two stages contains only one machine, provided the other machine contains at least two machines—hence the proof of theorem 1.

In addition to the problem being NP-complete, there is some evidence that the problem is NP-complete in the strong sense. This comes from the arguments presented by Garey and Johnson,[6] as shown by the following theorem.

*Theorem 2*

The two-machine, hybrid flowshop problem considered above is NP-complete in the strong sense for any arbitrary values of $m_1$ and $m_2$ provided $\max(m_1, m_2) > 1$.

*Proof*

This can easily be derived by extending the analysis in theorem 1 above, coupled with the developments on pp. 96, 106–107 and 238 in Garey and Johnson.[6]

The results of theorem 2 above, however, do not show that the problem is strongly NP-complete for the case where $m_1$ and $m_2$ are both fixed. While the structure of the problem suggests that this might be the case, no mathematical proof could be found. The fact that the multiprocessor scheduling problem with resource constraints is NP-complete in the strong sense, even if only partial orders (see Garey and Johnson,[6] p. 239) are included, tends to confirm the above assertion since the serial processing requirements of the jobs do create partial orders. In addition, several attempts by the author to develop a pseudo-polynomial algorithm for the case with fixed number of machines at either stage failed. Therefore, it is *conjectured* that the two-machine, hybrid flowshop problem considered in this paper is NP-complete in the strong sense.

In view of the NP-completeness (perhaps in the strong sense) of the problem, it is desirable that efficient and effective heuristic algorithms be developed for the problem.

## THE CASE WITH ONE MACHINE AT THE SECOND STAGE

In various practical situations, it may be important to eliminate (or decrease to a minimum amount) the total idle time at the second stage since only one expensive machine is available to complete the work of the second stage. This could be true even if the distribution of job-processing times for the two stages is identical. When there is only one machine at each stage, the idle time on the second stage is minimized by using Johnson's[1] algorithm. However, by increasing the number of machines at the first stage, it is possible to reduce this second-stage idle time even further.

### *Heuristic algorithm for this special case*

The solution to this problem requires two aspects: sequencing of jobs on both stages and assignment of jobs to various machines at each stage. For the case with only one machine at stage 2, assume that the sequencing and assignment aspects of the problem can be considered independently. This assumption may produce suboptimal results but saves considerable computational effort in finding an approximately optimal solution to the problem. The first part is easily solved, using Johnson's procedure, by assuming that there is only one machine at each stage. The second part, one of assigning jobs to multiple machines, is done by attempting to minimize the idle time on the second stage. Let $t(a, s)$ represent the processing time of job $a$ on stage $s$.

For each job $a$, define:

$$f(a) = \operatorname{sign}[t(a,1) - t(a,2)]/\min[t(a,1), t(a,2)],$$

where

$$\operatorname{sign}[t(a,1) - t(a,2)] = 1 \quad \text{if} \quad t(a,1) \geqslant t(a,2) \quad \text{and} \quad -1 \quad \text{otherwise.}$$

Then the schedule obtained by arranging jobs in ascending order of $f(a)$ values will minimize makespan if there is only one machine at each stage.[8]

With the above definition, the proposed heuristic algorithm can be stated as follows:

*Step 1: sequence identification.* Let $S = (a_1, a_2, \ldots, a_n)$ be the schedule obtained by arranging jobs in ascending order of $f(a)$, breaking ties in favour of a job with least processing time on stage 1 or greatest processing time on stage 2. If $t(a_1, 1) \leqslant t(a_i, 1)$ for all $i \geqslant 2$, go to step 2; otherwise modify $S$ by bringing the job with minimum processing time to the first sequence position.

*Step 2: assignment of jobs.* Assign job $a_i$ to the latest available machine at stage 1 such that no *additional* idle time is incurred at stage 2. If it is not possible, then assign job $a_i$ to the machine such that minimum *additional* idle time is incurred at stage 2.

*Step 3: approximate solution.* Accept the final assignment and schedule thus obtained as an approximate solution to the problem.

The above algorithm attempts to minimize the number of machines required at stage 1 as well. Because of the assignment of jobs to the latest available machine at stage 1, it will use an additional machine only if the makespan cannot be minimized by the use of a fewer number of machines.

## A NUMERICAL ILLUSTRATION

As an example of the above procedure, consider the eight-job problem with processing times and $f(a)$ values given in Table 1. There are two machines at stage 1 and one machine at stage 2.

*Step 1:* Arranging jobs in ascending order of $f(a)$ yields the schedule $S = (7, 4, 5, 6, 3, 8, 2, 1)$. Since job 1 has the minimum processing time at stage 1, the revised schedule is $S = (1, 7, 4, 5, 6, 3\ 8, 2)$.

*Step 2:* The assignment of jobs is now done and is shown in Table 2. To start with, job 1 is assigned to machine 1 at stage 1. This job completes its processing at stage 1 at four time-units and at five time-units at stage 2. The next job in the sequence is job 7. Among the two machines at stage 1, this job is assigned to machine 2 since such an assignment minimizes the idle time at stage 2. Similarly, job 4 is assigned to machine 2 at stage 1 since neither machine leads to *additional* idleness at stage 2, and machine 2 (at stage 1) is the later to become available. Proceeding this way, jobs are assigned to two machines at stage 1, and their completion times calculated as shown in Table 2.

*Step 3:* From Table 2, it is seen that the schedule $S = (1, 7, 4, 5, 6, 3, 8, 2)$ minimizes makespan when machine 1 at stage 1 processes jobs (1,5, 6, 3, 8, 2), and machine 2 at stage 1 processes jobs (7, 4), and the processing schedule is as in $S$. This solution is optimal since there is no intermediate idleness at stage 2 after a minimum unavoidable initial idleness of one time-unit.

The above example also illustrates that the proposed heuristic algorithm aims at minimizing the number of machines at stage 1. If the number of available machines at stage 1 were more than 2, the proposed algorithm would still use only two of these machines since it minimizes makespan.

## LOWER BOUNDS ON MAKESPAN

Arthanary and Ramaswamy describe lower bounds on the makespan of the problem when there is only one machine at stage 2. Their proposed bounds require too much computational effort. As such, the following simple and easily computable bounds were developed to evaluate the effectiveness of the proposed heuristic algorithm.

TABLE 1. *Processing times for an eight-job problem*

| Job $a$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $t(a, 1)$ | 4 | 5 | 8 | 6 | 8 | 9 | 5 | 10 |
| $t(a, 2)$ | 1 | 4 | 7 | 7 | 10 | 12 | 6 | 6 |
| $f(a)$ | 1 | 1/4 | 1/7 | −1/6 | −1/8 | −1/9 | −1/5 | 1/6 |

TABLE 2. *Assignment of jobs to machines*

| | Stage 1 | | |
|---|---|---|---|
| Job | Machine 1 | Machine 2 | Stage 2 |
| 1 | 4 | — | 5 |
| 7 | — | 5 | 11 |
| 4 | — | 11 | 18 |
| 5 | 12 | — | 28 |
| 6 | 21 | — | 40 |
| 3 | 29 | — | 47 |
| 8 | 39 | — | 53 |
| 2 | 44 | — | 57 |

Let $a_1$ be the job with minimum processing time on stage 1. Then one possible lower bound is:

$$LB_1 = t(a_1, 1) + \sum_{a=1}^{n} t(a, 2)$$

since the machine at stage 2 does remain idle for at least $t(a_1, 1)$ time-units, and this idleness cannot be decreased.

The above lower bound can be improved by considering simultaneous processing of jobs with minimum processing times on the first stage. Let $a_1$ and $a_2$ be the jobs with minimum and second minimum processing times on stage 1. Then the unavoidable idleness, *UI*, on the second stage is determined as follows:

$$UI = \max[t(a_1, 1) \quad t(a_2, 1) - t(a_1, 2)]$$

and, therefore, the lower bound on makespan is

$$LB_2 = UI + \sum_{a=1}^{n} t(a, 2).$$

It is readily seen that $LB_2 \geqslant LB_1$. Further, this lower bound can be improved by considering more than two jobs. Since this increases computational burden, no attempt was made to generalize $LB_2$ beyond consideration of two jobs.

Yet another lower bound is found by examining the case where each of the $m_1$ machines at stage 1 may process all jobs in an equal amount of time but there will be one job that still needs to be processed at stage 2. Thus

$$LB_3 = INT\left[\sum_{a=1}^{n} t(a, 1)/m_1\right] + \min_{1 \leqslant a \leqslant n} [t(a, 2)],$$

where $INT[X]$ indicates the least integer value greater than or equal to $X$.

From the above lower bounds, it follows that the makespan of an optimal schedule cannot be less than $LB = \max\,(LB_2, LB_3)$.

## COMPUTATIONAL EXPERIENCE

The effectiveness of the proposed heuristic algorithm in finding a minimum makespan schedule is worst when the number of machines at stage 1 equals 2, since the availability of more than two machines at stage 1 increases the possibility of no idleness on the second stage. Therefore, empirical evaluation of the proposed heuristic algorithm was restricted to the case where stage 1 has only two machines. For this purpose, the proposed heuristic algorithm was programmed in FORTRAN to solve 380 problems ranging from 5 jobs to 100 jobs. The processing times of these problems were generated from the same, discrete, uniform distribution in the range (1, 99). For each problem, percentage deviation of the makespan of the schedule generated by the heuristic algorithm from its lower bound was calculated. Based on 20 problems of each size ($n$), the following statistics were collected:

$n_1$: number of times heuristic makespan equalled its lower bound.
$n_2$: number of times the deviation of heuristic makespan from its lower bound was greater than zero but less than or equal to 3%.

TABLE 3. *Percentage deviation from lower bound for the heuristic makespan*

| $n$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | *MIN* | *AVR* | *MAX* |
|---|---|---|---|---|---|---|---|
| 5 | 18 | 2 | 0 | 0 | 0.0 | 0.196 | 1.980 |
| 6 | 19 | 0 | 0 | 1 | 0.0 | 0.821 | 16.425 |
| 7 | 18 | 2 | 0 | 0 | 0.0 | 0.050 | 0.709 |
| 8 | 18 | 0 | 2 | 0 | 0.0 | 0.435 | 4.779 |
| 9 | 17 | 1 | 0 | 2 | 0.0 | 1.047 | 11.347 |
| 10 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 15 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 20 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 25 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 30 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 35 | 19 | 1 | 0 | 0 | 0.0 | 0.005 | 0.105 |
| 40 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 45 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 50 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 60 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 70 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 80 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 90 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 100 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |

$n_3$: number of times the deviation of heuristic makespan from its lower bound was greater than 3% but less than or equal to 5%.

$n_4$: number of times the deviation of heuristic makespan from its lower bound was greater than 5%.

*MIN*: minimum percentage deviation of the makespan from its lower bound.

*AVR*: average percentage deviation of the makespan from its lower bound.

*MAX*: maximum percentage deviation of the makespan from its lower bound.

Table 3 depicts these summary statistics for the 380 test problems.

From Table 3, it is clear that the proposed heuristic finds an optimum schedule in most cases. The percentage deviation of makespan from its lower bound is rarely more than 5%. Further, the number of times heuristic makespan equals its lower bound increases with an increase in the number of jobs. Since the minimum makespan may be more than its lower bound, actual percentage deviations of heuristic makespans from minimum makespans will be no more than those shown in Table 3.

It may, however, be thought that the algorithm performance will decrease when the processing times of various jobs at the second stage are less than those at the first stage. In order to test the effectiveness of the proposed algorithm for such cases, the computational experiments were repeated to solve problems involving 5 to 100 jobs when processing times on the first stage were generated from a uniform discrete distribution in the range (1, 99) for the first case and (1, 49) for the second stage. Based on the solution of 20 problems of each size, Table 4 depicts the results obtained.

TABLE 4. *Percentage deviation from lower bound for the heuristic makespan when second-stage processing times may be less than the first-stage processing times*

| $n$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | *MIN* | *AVR* | *MAX* |
|---|---|---|---|---|---|---|---|
| 5 | 16 | 1 | 1 | 2 | 0.0 | 1.989 | 26.154 |
| 6 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 7 | 17 | 2 | 1 | 0 | 0.0 | 0.346 | 3.030 |
| 8 | 17 | 2 | 0 | 1 | 0.0 | 0.694 | 10.853 |
| 9 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 10 | 19 | 1 | 0 | 0 | 0.0 | 0.0195 | 0.389 |
| 15 | 19 | 1 | 0 | 0 | 0.0 | 0.0848 | 1.695 |
| 20 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 25 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 30 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 35 | 19 | 1 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 40 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 45 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 50 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 60 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 70 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 80 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 90 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |
| 100 | 20 | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 |

Based on the results in Table 4, it may be concluded that the effectiveness of the proposed heuristic algorithm does not significantly change with the change in the processing times. Thus, in view of these computational experiments, it may be concluded that the proposed heuristic algorithm is quite effective in minimizing makespan for the two-stage, hybrid flowshop problem with only one machine at stage 2.

The computational time for the proposed heuristic algorithm was not measured since it is relatively small. In the worst case, the number of computational steps will not be more than $0(n \log n + nm)$, where $m$ is the number of machines at stage 1.

## CONCLUSIONS

This paper has discussed the two-stage, hybrid flowshop scheduling problem and has described a heuristic algorithm for its solution when stage 2 contains only one machine. The effectiveness of the proposed heuristic algorithm was empirically evaluated and found to increase as the problem-size increased. In view of the NP-completeness (perhaps in the strong sense) of the problem, this result is quite encouraging since it provides an efficient procedure for solving large-sized problems. Further, the proposed heuristic algorithm may be used to find an initial upper bound to reduce the size of the decision tree in the branch-and-bound procedure such as the one proposed by Arthanary and Ramaswamy.[1]

*Acknowledgement*—Helpful comments from two anonymous referees improved the clarity of this paper.

## REFERENCES

1. S. M. Johnson (1954) Optimal two- and three-stage production schedules with setup times included. *Nav. Res. Log. Q.* **1,** 61–68.
2. T. S. Arthanary and K. G. Ramaswamy (1971) An extension of two machine sequencing problem. *Opsearch* **8,** 10–22.
3. B. S. Mittal and P. C. Bagga (1973) Two machine sequencing problem with parallel machines. *Opsearch* **10,** 50–61.
4. R. N. Murty (1974) On two machine sequencing problems with parallel machines. *Opsearch* **11,** 42–44.
5. T. B. K. Rao (1970) Sequencing in the order A, B with multiplicity of machines for a single operation. *Opsearch* **7,** 135–144.
6. M. R. Garey and D. S. Johnson (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, Calif.
7. A. H. G. Rinnooy Kan (1977) *Machine Scheduling Problems: Classification, Complexity and Computations.* Martinus Nijhoff, The Hague.
8. J. N. D. Gupta (1975) Optimal schedules for special structure flowshops. *Nav. Res. Log. Q.* **26,** 255–269.