J. Opl Res. Soc. Vol. 40, No. 4, pp. 395–399, 1989
Printed in Great Britain. All rights reserved

0160-5682/89 $3.00 + 0.10
Copyright © 1989 Operational Research Society Ltd

Alternative Formulations of a Flow-shop Scheduling Problem

J. M. WILSON
Department of Management Studies, Loughborough University of Technology

This paper presents an alternative approach to an earlier model developed by other authors to formulate a problem of sequencing N jobs on M machines in a standard flow-shop. The objectives of the model are to minimize makespan and flow-time. The new formulation involves substantially fewer variables, at the expense of a rise in the number of constraints. Practical limitations of both approaches are discussed.

Key words: integer programming, multi–objective, production, scheduling

INTRODUCTION

A recent paper by Selen and Hott[1] describes a mixed-integer goal-programming formulation of a flow-shop problem. The problem is one of scheduling N jobs in a shop containing M machines, where each job has to be processed on every machine, and every job follows the same ordering of machines as it is processed. The objectives of the problem are to minimize makespan and flow-time. In what follows, an alternative approach is developed for formulating the problem. The alternative approach has the advantage that it involves substantially fewer variables, at the expense of additional constraints, and conforms to a more natural modelling of precedence. Practical limitations of the goal-programming approach to handle two objectives are also discussed.

THE MODEL

The approach taken by Selen and Hott[1] uses the relationship for jobs in the jth and $(j + 1)$th positions on machines k and $k + 1$ that

$$\begin{array}{c}\text{idle time of machine } k\\ \text{before job } j+1\end{array} + \begin{array}{c}\text{processing time of job}\\ j+1 \text{ on machine } k\end{array} + \begin{array}{c}\text{idle time of job } j+1\\ \text{after machine } k\end{array}$$

is equal to

$$\begin{array}{c}\text{idle time of machine } k+1\\ \text{before job } j+1\end{array} + \begin{array}{c}\text{processing time of job } j\\ \text{on machine } k+1\end{array} + \begin{array}{c}\text{idle time of job } j\\ \text{after machine } k\end{array}$$

This relationship can be checked in Figure 1.

Figure 1 shows the scheduling of the first three jobs to machines $k - 1$, k and $k + 1$ in a sample problem. Job times are proportional to the length of the vertical lines in the figure, and connections between job times and idle times can be seen from the position of the dotted horizontal lines.

In what follows, the approach taken will be based on the relationship that the commencement of job $j + 1$ on machine $k + 1$ is later than both the finish time of job $j + 1$ on machine k and the finish time of job j on machine $k + 1$. For this model the notation used by Selen and Hott[1] is continued where appropriate.

NOTATION

i = job number, $i = 1, 2 \ldots, N$; N is the total number of jobs;
j = schedule position for processing, $j = 1, 2, \ldots N$; N possible positions;
k = operation (machine) number $k = 1, 2, \ldots M$; M is the total number of machines in the shop;
z_{ij} = 1 if job i is scheduled in position j for processing, 0 otherwise;

t_{ik} = the processing time of job i on machine k;
s_{jk} = time at which the job in schedule position j commences processing on machine k.

Selen and Hott[1] explicitly model all idle times, but that will not be done here.

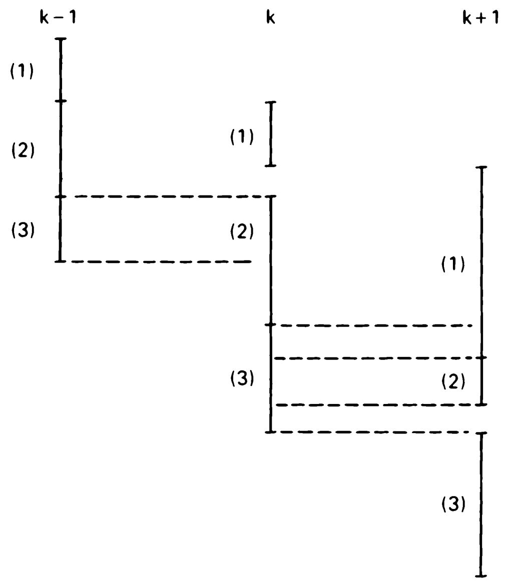

FIG. 1. *Scheduling of three jobs.*

CONSTRAINTS

Jobs must be uniquely positioned for scheduling:

$$\sum_{i=1}^{N} z_{ij} = 1 \qquad j = 1, 2, \ldots, N \tag{1}$$

$$\sum_{j=1}^{N} z_{ij} = 1 \qquad i = 1, 2, \ldots, N. \tag{2}$$

There is no idle time on machine 1 nor on job 1, and so

$$s_{j+1,1} = s_{j1} + \sum_{i=1}^{N} t_{i1} z_{ij} \qquad j = 1, 2, \ldots, N - 1 \tag{3}$$

$$s_{11} = 0 \tag{4}$$

$$s_{1,\,k+1} = s_{1k} + \sum_{i=1}^{N} t_{ik} z_{i1} \qquad k = 1, 2, \ldots, M - 1. \tag{5}$$

The commencement of job j on machine $(k + 1)$ is later than its finish time on machine k:

$$s_{j,k+1} \geqslant s_{jk} + \sum_{i=1}^{N} t_{ik} z_{ij} \qquad \begin{array}{l} j = 2, 3, \ldots, N; \\ k = 1, 2, \ldots, M - 1. \end{array} \tag{6}$$

The commencement of job $(j + 1)$ on a machine is later than the finish time of a job j on the same machine:

$$s_{j+1,k} \geqslant s_{jk} + \sum_{i=1}^{N} t_{ik} z_{ij} \qquad \begin{array}{l} j = 1, 2, \ldots, N - 1 \\ k = 2, 3, \ldots, M. \end{array} \tag{7}$$

The objective of minimizing makespan is

$$\text{minimize} \left(s_{NM} + \sum_{i=1}^{N} t_{iM} z_{iN} \right). \tag{8}$$

The objective of minimizing total flow-time is

$$\text{minimize}\left(\sum_{j=1}^{N} s_{jM}\right). \tag{9}$$

SIMPLIFICATION OF MODEL

Constraints (1)–(5) are essentially definitional, while constraints (6) and (7) model precedence. Simplifications are possible to reduce the number of z_{ij} variables, but these are not particularly advantageous and will not be included as they are likely to adversely affect the density of the coefficient matrix.

EXAMPLE PROBLEM

Selen and Hott[1] suggest the data of Table 1 as a small example. This example can be modelled using (1)–(8), and involves 60 variables, of which 36 are $(0 - 1)$, and 50 constraints, and then objective function (8) is optimized and added as a constraint to allow (9) to be optimized. The model described by Selen and Hott[1] required 73 variables and 34 constraints. Thus the alternative approach involves fewer variables at the expense of more constraints. The model with objective function (8) converged immediately to an integer answer, a fairly common property of a naturally formulated sequencing problem.

TABLE 1. *Problem processing times*

	job					
machine	1	2	3	4	5	6
1	10	12	14	15	13	12
2	9	7	6	4	9	7
3	6	4	3	2	4	5
4	2	2	1	1	3	3

PRACTICAL LIMITATIONS

The size of the proposed model is

$2MN + N - M$ constraints,

N^2 $(0 - 1)$ variables,

MN other variables.

The size of the Selen and Hott[1] model is

$MN + N + M$ constraints,

N^2 $(0 - 1)$ variables,

$2MN - 2N + 1$ other variables.

The major limitation for larger problems will be the N^2 binary variables. Mixed-integer problems are usually difficult to solve, and a model with several hundred binary variables may be impracticable. Recent work on solving difficult integer-programming problems is reported in Crowder *et al.*,[2] and this work gives hope for the future. However, the practitioner, limited by existing commercial software and hardware, would be unlikely to be able successfully to solve problems where N was large. It might be realistic to expect problems to involve up to seven machines and 50 jobs. Such problems would require 2500 $(0 - 1)$ variables and would be unlikely to be solvable in realistic times.

COMPUTATIONAL RESULTS

The formulation of this paper was compared with the formulation of Selen and Hott[1] on a series of test problems run on a Prime 750 computer. The matrix generator system MGG/VM[3] was used to generate the integer-programming problem, and SCICONIC/VM[4] was used as the optimization system. A slight modification was necessary to the model for the Selen and Hott[1] formulation for ease of use of MGG/VM, so that in Table 2 the small problems appear to be a slightly different size from that quoted in the Selen and Hott[1] paper. However, the formulation used was mathematically equivalent to the Selen and Hott[1] formulation to minimize makespan. The results are given in Table 2, which shows statistics on the problems. Each pair of results represents one problem formulated by the Selen and Hott[1] (SH) approach or the alternative formulation (AF) of this paper. The 'special problem' and the 'general problem' refer to the two test problems described in Selen and Hott.[1] Numbers under the heading 'problem' refer to number of machines and number of jobs. The computational results relate to a variety of problems of up to seven machines and 20 jobs in size. It was found that larger problems did not fit into the size configuration of the software as it is currently set up, although it might be expected that some larger problems could be solved using a different configuration. It was also found that CPU times were quite sensitive to the order of input of constraints.

TABLE 2. *Comparison of two formulations*

Problem	Rows	Columns	CPU seconds to continuous solution	Nodes to first integer solution	CPU seconds to first integer solution	Nodes to complete search	CPU seconds to complete search	Total simplex iterations
General 4 × 6 AF	51	60	2.9	0	3.3	0	3.3	52
General 4 × 6 SH	35	73	4.4	0	5.4	0	5.4	67
Special 4 × 6 AF	51	60	5.6	2	6.5	3	7.0	79
Special 4 × 6 SH	35	73	3.6	3	5.0	31	14.0	132
4 × 10 AF	87	140	10	0	11	0	11	109
4 × 10 SH	55	161	15	0	16	0	16	135
4 × 15 AF	132	285	33	0	34	0	34	214
4 × 15SH	80	316	38	0	39	0	39	215
4 × 20AF	177	480	136	0	138	0	138	520
4 × 20 SH	105	521	73	2	81	3	81	306
5 × 10 AF	106	150	34	5	39	9	39	239
5 × 10 SH	66	181	18	8	26	15	26	166
6 × 10 AF	125	160	62	11	135	48	135	571
6 × 10 SH	77	201	21	13	56	42	56	272
7 × 20 AF	294	540	349	34	1248	55	1248	1945
7 × 20 SH	168	641	165	26	358	46	358	740

AF denotes formulation of this paper. SH denotes Selen and Hott formulation.

The results in Table 2 seem to indicate that

(a) for small problems, with a small number of machines, the formulation of this paper performs best;
(b) for larger problems the Selen and Hott[1] formulaton performs best;
(c) the breakeven point would seem to be about 15 jobs and four machines, and above these sizes (b) becomes dominant;
(d) the computational times required for problems of up to 20 jobs and seven machines seem quite acceptable.

MULTIPLE OBJECTIVES

The use of two objectives in the formulation is obviously a helpful step towards realistic modelling. However, the approach suggested by Selen and Hott[1]—namely to minimize makespan first and then seek solutions which minimize flow-time, while keeping makespan minimized—seems limited.

In practice it might be expected that a time t_{ik} comprises three elements:
(a) a set-up time (possibly manual),
(b) a processing time (machining),
(c) a delay time (possibly a result of breakage).
Elements (a) and (b) might be assumed to arise from normal distributions. Element (c) might be a variable which takes the value zero with high probability, and with low probability takes a non-zero value which is much larger than the sum of the means of (a) and (b).

Thus a more realistic two-stage programming approach might run as follows. First, find a lower limit on makespan. Secondly optimize flow-time, given the restriction

$$\text{makespan} < (1 + x) \text{ (lower limit on makespan)},$$

where x is a factor such as 0.05 to model the fact that makespan may vary within a certain tolerance of a theoretical lower limit without making much difference to the perceived quality of the solution. A series of solutions could be investigated for various values of x to give some idea of the robustness of the sequencing policy.

CONCLUSION

A goal-programming mixed-integer programming approach to flow-shop scheduling has been reconsidered and modified to produce a model with fewer variables than a previous version. The result is an approach to a standard scheduling problem which works well on smaller problems. Some doubt is cast on the basic assumptions of the goal-programming approach, and an attempt has been made to avoid some limitations of the approach. However, limitations remain, given the existing state of the art of integer programming, which ensure that large models are difficult to solve without using very large amounts of computer time. Heuristics suggested from the features of the integer programming solution would then seem to offer the best way forward in practice to solve realistically sized problems.

REFERENCES

1. W. J. Selen and D. D. Hott (1986) A mixed integer goal-programming formulation of a flow-shop scheduling problem. *J. Opl Res. Soc.* **37**, 1121–1128.
2. H. Crowder, E. L. Johnson and M. W. Padberg (1983) Solving large-scale zero-one linear programming problems. *Opns Res.* **31**, 803–834.
3. *MGG User Guide* (1987) Scicon Limited, Milton Keynes.
4. *SCICONIC/VM User Guide* (1986) Scicon Limited, Milton Keynes.