JATINDER N. D. GUPTA

A heuristic algorithm for the flowshop scheduling problem

Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle, tome 10, nº V2 (1976), p. 63-73.

<http://www.numdam.org/item?id=RO_1976__10_2_63_0>

R.A.I.R.O. Recherche opérationnelle
(vol. 10, n° 6, juin 1976, p. 63 à 73)

A HEURISTIC ALGORITHM FOR THE FLOWSHOP SCHEDULING PROBLEM (*)

by Jatinder N. D. GUPTA [1]

Abstract. — *This paper describes a simple heuristic algorithm for seeking a quick and approximate solution to the n-job, M-machine flowshop scheduling problem under the assumptions that the process times of the jobs are deterministic and the same order of jobs is followed on all machines. The proposed algorithm is based on the fact that the flowshop scheduling problem may be considered as* $(M-1)$ *quasi equivalent sorting problems and that an approximate solution to the flowshop scheduling problem, then, may be obtained by solving the corresponding sorting problems. The proposed heuristic algorithm can be executed by hand for reasonably large-sized problems and yields solutions that are comparatively closer to optimal solutions than those obtained by Campbell-Dudek-Smith heuristic algorithm. These computational results are discussed and efficiency of the proposed heuristic is compared with that of Campbell-Dudek-Smith algorithm.*

1. INTRODUCTION

The flowshop scheduling problem considered here is one of scheduling a given number of jobs on given number of machines in a shop where the flow of work is unidirectional. The unidirectional flow of work implies that the technological order of all jobs on all machines is identical. This problem was first formulated by Johnson [14] as an n-job, 2-machine production scheduling problem when the objective is to minimize the throughput time (called the make-span) of all jobs. Subsequent developments in scheduling theory have been extensions of Johnson's formulation, in that the number of machines is increased to the general case M $(M \geqq 3)$. Recently, there has been considerable interest in finding suitable mathematical techniques to solve the flowshop scheduling problem and substantial progress has been made in the development of efficient algorithms for obtaining optimal or near-optimal solutions to the flowshop scheduling problems ([1-6], [8-21]). In all the algorithms, several restrictive assumptions are made, the complete statement of these assumptions is provided by Ashour [1] and Dudek and Teuton [6] and hence is not repeated here.

(*) Reçu décembre 1974, version révisée juillet 1975.

[1] U.S. Postal Service Headquarters, Washington, D. C.

The available solution techniques may be divided into two classes; (1) techniques which assure optimality of the solution, and (2) techniques which don't guarantee optimality but yield solutions which are very near-optimal solutions. The exact solution techniques usually require an electronic computer and, because of practical limitations of computer memory and excessive computational effort (cost) of obtaining the solution, the range of their applicability is limited to moderately small sized problems [5, 10, 16, 17, 18]. Thus, while research is being continued to develop efficient optimizing algorithms and/or faster and more economical computational devices with larger memories, companies faced with large-sized flowshop scheduling problems have to resort to heuristic techniques to solve their problems and feel contented by approximate solutions only [5, 10, 18]. For the flowshop scheduling problem, Campbell, Dudek, and Smith [5], Gupta [10], and Palmer [17] have suggested heuristic algorithms which can be applied to large-sized problems even for hand computations. These heuristic approaches to scheduling problems are based on Page's [16] analogy between scheduling and sorting problems. Further, this analogy between scheduling and sorting suggest a strong heuristic algorithm for a *simple and approximate solution* (1) of the M-stage flowshop scheduling problem. This paper describes a heuristic algorithm and compares its effectiveness and efficiency with the Campbell algorithm. The proposed heuristic algorithm decomposes the original M-stage flowshop scheduling problem into a series of 2-machine problems (as in the case of the Campbell algorithm) and uses Page's analogy between scheduling and sorting to solve these decomposed problems. Since the decomposition of the original problem into smaller sub-sets is not exact, the solutions so obtained are not in general optimal.

2. THE FLOWSHOP SCHEDULING PROBLEM

In the flowshop scheduling problem discussed here, as indicated earlier, the flow of work is unidirectional and the order in which jobs are processed on machines is the same and is completely specified. Since the numbering of machines is arbitrary, the machines can be numbered such that jobs are processed on machine 1 first, machine 2 second, ..., and machine M last [11]. With this nomenclature of machines and above discussion of the problem, the flowshop scheduling problem may be stated as:

"Given n jobs to be processed on M machines in the same order, the process time of job i on machine m being t_{im} $(i = 1,2, \ldots, n; m = 1, 2, \ldots, M)$, find that common order in which these n jobs should be processed on the

(1) The term *simple and approximate solution* is used here to mean that the solution procedure can be carried through mannually for reasonably large-sized problems and the make-span of the schedule thus obtained is not very much longer than the minimum (optimal) make-span.

M machines which minimizes the throughput time (called the make-span) of all jobs."

In order to make this problem tractable, consider a partial sequence σ containing k, $(k < n)$, jobs and the augmentation of job a to σ represented by the concatenation of σ and a (written as σ a). Then, following the physical constraints of the problem (non-simultaneous processing of a job on two or more machines and non-simultaneous processing of two or more jobs by the same machine) and the assumptions outlined by Dudek and Teuton [6], the recursive relation for the completion time of the partial sequence $\sigma\, a$ of length $(k+1)$ at machine m, $T(\sigma\, a, m)$, is as follows [4, 8, 9]:

$$\left.\begin{array}{l} T(\sigma\, a,\, m) = \max\left[T(\sigma,\, m);\; T(\sigma\, a,\, m-1)\right] + t_{am}, \\ \text{where} \\ T(\varphi,\, m) = T(\sigma,\, 0) = 0 \quad \text{for all } \sigma \text{ and } m. \end{array}\right\} \qquad (1)$$

Then, the flowshop scheduling problem stated above, is one of minimizing $T(\sigma\, a, M)$ where a ranges over all the n jobs and σ ranges over all possible sequences of $(n-1)$ jobs not containing job a [9].

3. THE HEURISTIC ALGORITHM

The flowshop scheduling problem, as formulated above, is a typical quantitative combinatorial search problem and permits of a finite number of feasible solutions, in fact equal to $n!$. Theoretically, therefore, the optimal solution can be obtained by a direct enumeration of all the feasible schedules. However, the practical difficulty in such a direct enumerational approach to the scheduling problem is caused by the fact that number of feasible schedules approaches a limit outside the range of practical computational facilities even for problems as small as the ones containing 10 jobs. Thus, if some functional representation and quasi-equivalent sorting problems can be developed, at least an approximate solution to the problem under consideration can be obtained.

The quantitative search problem associated with the scheduling of n jobs on M machines may be considered as a case of sorting n items as to minimize a function, $f(t_{11}, \ldots, t_{nM})$, of the process times, called the make-span. The equivalent sorting problem so defined is quite complex because the corresponding function is not separable in terms of the individual items in the sense that for any specific job, such function cannot be determined without the knowledge of the preceeding and following sequences of other jobs. Because of this complexity, it seems unlikely that a function, independent of other jobs which reflects the merit or demerits of its occupying a specific sequence position, can be assigned to each job. However, as shown by Page [16] and Bakshi and Arora [3], for Johnson's two machine case, such an equivalent function can be obtained. Thus for Johnsons' two machine case, the function associated with job i, $f(i)$, takes the following form:

$$\left.\begin{array}{l} f(i) = \dfrac{A_i}{\min(t_{i1}, t_{i2})}, \\ \text{where} \\ A_i \begin{cases} = 1 & \text{if } t_{i1} \geqq t_{i2} \\ -1 & \text{otherwise.} \end{cases} \end{array}\right\} \quad (2)$$

Following the above functional representation, the two machine scheduling problem is equivalent to the problem of sorting n numbers in such a way that $f(a_1) \leqq f(a_2) \leqq \ldots \leqq f(a_n)$ where a_i represents the job at the ith sequence position. This sorting problem can now be easily solved by ordering the jobs in ascending order of $f(i)$.

It seems very much unlikely that simple functions of the form (2) can be developed for the general M-stage flowshop scheduling problem [16]. However, it is possible to take advantage of this analogy between scheduling and sorting and obtain a quick and approximate (near optimal) solution to the problem. Thus, the given M-stage scheduling problem is broken down to K auxiliary 2-machine problems, each of which is solved by using relation (2) above. The process times of the kth auxiliary 2-machine problem are obtained from the original problem as follows: the sum of the process times for the *first k* machines is treated as the process time of a job on machine 1 of the auxiliary problem, while the sum of the process times for the *last k* machines forms the process time of a job on machine 2 of the auxiliary problem.

Let $p_k(i, 1)$ and $p_k(i, 2)$ be the process times of job i for the k th auxiliary problem. Then $p_k(i, 1)$ and $p_k(i, 2)$ are determined by the following relation [1]:

$$\left.\begin{array}{ll} p_k(i, 1) = \sum_{m=1}^{k} t_{im}, & i = 1, 2, \ldots, n \\ \text{and} & \\ p_k(i, 2) = \sum_{m=M+1-k}^{M} t_{im}, & k = 1, 2, \ldots, K \end{array}\right\} \quad (3)$$

The corresponding function $R(k, i)$ for the kth auxiliary problem, then, takes the following form:

$$\left.\begin{array}{l} R(k, i) = A_{ki}/\min(p_k(i, 1), p_k(i, 2)), \\ \text{where} \\ A_{ki} \begin{cases} = 1 & \text{if } p_k(i, 1) \geqq p_k(i, 2) \\ -1 & \text{otherwise.} \end{cases} \end{array}\right\} \quad (4)$$

[1] Use of relations (3) above limits the maximum value of K to $(M-1)$.

Then the proposed heuristic algorithm may be described by the following step-by-step procedure:

STEP 1: For each k and i, using relation (3) and (4), calculate $R(k, i)$.

STEP 2: For each k, arrange the jobs in ascending order of $R(k, i)$ breaking the tie in favor of a job with lesser value of $R(\alpha, i)$, $\alpha = k + 1, \ldots, M - 1$ or $\alpha = k - 1, k - 2, \ldots, 1$. If the tie cannot be broken, choose any one job arbitrarily.

STEP 3: For each of the k schedules so generated, using the recursive relation (1) above, calculate the make-span.

STEP 4: Select the schedule with minimum make-span. This is the approximate solution to the problem.

4. A NUMERICAL ILLUSTRATION

The working of the above heuristic algorithm is explained by solving the 4-job, 5-machine problem of Table I:

i \ m	1	2	3	4	5
1	4	3	7	2	8
2	3	7	2	8	5
3	1	2	4	3	7
4	3	4	3	7	2

TABLE I

Process Time Matrix

STEP 1: This step involves the calculation of $R(k, i)$. Using relations (3) and (4), the $R(k, i)$ obtained are shown in Table II.

STEP 2: For each value of k, the jobs are to be arranged in ascending order of $R(k, i)$, breaking the ties. Observe that ties exist for $k = 2$ and $k = 4$. For $k = 2$, tie exists between jobs 1 and 4. Since $k = 2$, we set $\alpha = k + 1 = 3$ and observe that $R(3, 4) < R(3, 1)$. Therefore, tie is broken in favor of job 4. Proceeding in this manner. four complete schedules are generated.

i	R(k, i)			
	k = 1	k = 2	k = 3	k = 4
1	-1/4	-1/7	-1/14	-1/16
2	-1/3	-1/10	-1/12	-1/20
3	-1	-1/3	-1/7	-1/10
4	1/2	-1/7	-1/10	+1/16

TABLE II

Functional Values for Example Problem

STEP 3: Using recursive relation (1), the make-spans of the four schedules are obtained and are shown in Table III.

Schedule	Make-Span
3214	36
3412	33
3421	39
3124	34

TABLE III

Make-Span of Schedules

STEP 4: Among the four schedules of Table III, schedule 3412 has a minimum make-span of 33 time-units. Hence it is accepted as the solution of the problem.

For the above problem, the proposed heuristic algorithm results in an optimal schedule as can easily be confirmed by using the lexicographic search technique [8]. However, it will be wrong to leave the reader with an impression that the proposed algorithm yields optimal results in all cases (see the next section).

5. COMPUTATIONAL EXPERIENCE

In order to investigate the effectiveness and efficiency of the proposed heuristic algorithm, considerable experimentation was conducted. Since the Campbell algorithm seems to be the most effective heuristic algorithm available in the literature, the performance of the proposed algorithm was was compared with that of the Campbell heuristic algorithm. For this purpose the proposed and the Campbell heuristic algorithms were programmed in FORTRAN language for an UNIVAC 1108 computer. While writing the program for the proposed heuristic algorithm, no special routine was developed for solving the corresponding sorting problems. Instead, the standard pair comparison method of digital simulation [7] was used to obtain the schedules as per step 2 of the algorithm with additional feature of breaking the ties. In order to carry out these experimental investigations, 1,055 problems with number of jobs varying from 4 to 60 and the number of machines varying from 3 to 60 were generated and solved by the proposed and the Campbell algorithms. The process times of the jobs in the above problems were randomly generated from a rectangular distribution and ranged from 00-99. For purposes of comparing the two algorithms, two factors, viz: the quality of solution (algorithm's effectiveness in finding a better solution) by each algorithm and the computational time required to obtain the solution, are important. Both of these are discussed below.

5.1. Effectiveness of the Algorithms

The 1,055 problems were divided into two sets: one consisting of problems for which the optimal solutions were known since these problems could be solved by using lexicographic search algorithm [8] without excessive computer time and the other for which the optimal solutions were unknown. The number of jobs in the first set varied from 4 to 7 and the number of machines varied from 3 to 7, whereas, in the second set, the numbers of jobs and machines varied from 10 to 60.

a) Problems with known optimal solutions

For the first set of problems, the effectiveness of the algorithm was measured by obtainning the percentage error of the algorithm schedule makespan from the optimal make-span. Thus, the percentage error is given by:

$$\alpha_p = \frac{T_p - T_0}{T_0} \times 100$$

Problem	Size	Number of Problems	Proposed Algorithm			Campbell Algorithm		
n	M		Average Error (α_p)	Number of Times Optimal Observed	Range of α_p	Average Error (α_c)	Number of Times Optimal Observed	Range of α_c
4	3	40	0.616	30	0- 8.657	0.896	30	0- 8.657
5	3	40	2.984	18	0-13.953	3.507	17	0-15.017
6	3	40	1.711	26	0-10.602	2.465	23	0-16.619
7	3	40	3.032	14	0-17.371	3.715	14	0-17.371
4	4	40	2.601	19	0-23.908	4.859	18	0-27.622
5	4	40	2.370	18	0-10.949	3.144	16	0-23.636
6	4	40	4.583	9	0-15.814	6.673	8	0-33.708
4	5	40	2.386	23	0-22.989	7.936	15	0-39.143
5	5	40	3.710	13	0-15.258	5.750	10	0-20.707
6	5	40	4.595	11	0-17.647	5.593	9	0-31.808
7	5	40	6.122	5	0-26.163	8.740	5	0.29.032
4	6	40	2.226	19	0- 8.116	3.879	19	0-20.652
5	6	40	3.682	10	0-14.549	6.063	10	0-25.820
6	6	40	4.274	8	0-14.537	5.721	5	0-17.761
7	6	40	5.464	5	0-16849	11.250	2	0.39.356
4	7	40	2.739	19	0-10.229	3.840	13	0.15.537
5	7	40	4.202	11	0-19.504	8.433	7	0-36.546
6	7	40	5.385	5	0-15.485	10.067	1	0-24.849
7	7	40	5.872	3	0-19.381	9.290	0	0.876-19.565

TABLE IV

Percentage Error Comparison on small Problems and

$$\alpha_c = \frac{T_c - T_0}{T_0} \times 100,$$

where α_p and α_c are the percentage errors for the proposed and the Campbell heuristic algorithms respectively and T_p, T_c, and T_0 are the make-spans obtained by proposed, Campbell and lexicographic search algorithms respectively.

Problem Size		# Problems	Average Value of β	Range of β
n	M			
10	10	5	0.971	0.936-0.998
10	20	5	0.974	0.951-0.991
20	20	5	0.964	0.924-0.992
40	20	5	0.977	0.947-0.996
60	20	5	0.972	0.969-0.996
20	40	5	0.978	0.962-0.994
40	40	5	0.939	0.952-1.045(a)
60	40	5	0.985	0.977-0.991
20	60	5	0.984	0.969-0.997
40	60	5	0.991	0.953-1.009(a)
60	60	5	0.981	0.979-0.984
(a) observed in one case only				

TABLE V

Comparison of Algorithms on Large Problems

For the 1,000 problems belonging to this set, Table IV contains the data concerning the average errors and their ranges by the proposed and the Campbell algorithms.

b) Problems with unknown optimal solutions

Since the optimal solutions to these problems were unknown, the ratio, $\beta = T_p/T_c$, was used to judge the comparative effectiveness of the two algorithms. If the proposed algorithm solution is better than the Campbell Solution, $\beta < 1$ and the lower the value of β, the better is the quality of the proposed algorithm solution as compared to the Campbell Solution. Further, because of larger size of problems, the total number of schedules for each problems were limited to six (i. e. $K = 6$). Table V shows the average value of β and its ranges for 55 problems belonging to this set.

5.2. Efficiency of the Algorithm

The average computational time required to solve a problem may be considered as the measure of computational efficiency of the algorithm. For the above, 1,055 problems, the proposed and the Campbell algorithms require about the same computation time. The variations in computation times times between the two algorithms were in micro-seconds and hence are not reported here.

A review of the results in Tables IV and V shows that the proposed heuristic algorithm generally yields better results than the Campbell algorithm especially for large-sized problems. Further, the computation time required to solve a problem by the proposed algorithm is the same as that by the Campbell algorithm. Based on this computational experience, therefore, it may be said that the proposed heuristic algorithm is comparatively more effective and efficient that the Campbell algorithm.

6. CONCLUSIONS

The heuristic algorithm described above provides a practical solution to larger sized flowshop scheduling problems which cannot be solved by exact solution techniques. Solutions obtained by the proposed algorithm are optimal or near optimal and are consistently better than those obtained by the Campbell algorithm. The simplicity and ease of computations in the proposed algorithm make it possible to solve reasonably large-sized problems manually, though the use of computers will definitely increase the quality of solution (because of increased number of schedules K) and the size of the problem that can be solved by the algorithm.

REFERENCES

1. S. Ashour, *A Branch and Bound Algorithm for the Flowshop Scheduling Problems*, A.I.I.E. Transactions, 2, No. 2, 1970, pp. 171-178.
2. S. Ashour, *An Experimental Investigation and Comparative Evaluation of Flowshop Scheduling Techniques*, Operations Research, 18, No. 3, 1970, pp. 541-549.
3. M. K. Bakshi and S. R. Arora, *The Sequencing Problem*, Management Science, 16, No. 4, 1969, B247-B263.
4. A. P. G. Brown and Z. A. Lomnicki, *Some Applications of the 'Branch and Bound' Algorithm to the Machine Scheduling Problem*, Operational Research Quarterly, 17, No. 2, 1966, pp. 173-186.

5. H. G. CAMPBELL, R. A. DUDEK and M. L. SMITH, *A Heuristic Algorithm for the n Job, m Machine Sequencing Problem*, Management Science, 16, No. 10, 1970, B630-B637.

6. R. A. DUKEK and O. F., TUETON Jr., *Development of M-Stage Dedision Rule for Scheduling n Jobs through m Machines*, Operations Research, 12, No. 3, 1964, pp. 471-497.

7. R. H. GREGORY and R. L. VAN HORN, *Automatic Data-Processing Systems*, Wadsworth Publishing Company, Inc., Belmont, California, Chapter 13, 1963.

8. J. N. D. GUPTA, *A General Algotithm for the $n \times M$ Flowshop Scheduling Problem*, The International Journal of Production Research, 7, No. 3, 1969, pp. 241-247.

9. J. N. D. GUPTA, *M-Stage Flowshop Scheduling by Branch and Bound*, Opsearch, (India), 7, No. 1, 1970, pp. 37-43.

10. J. N. D. GUPTA, *A Functional Heuristic Algorithm for the Flowshop Scheduling Problem*, Operational Research Quarterly, 22, No. 1, 1971, pp. 39-47.

11. J. N. D. GUPTA, *M-Stage Scheduling Problem—A Critical Appraisal*, The International Journal of Production Research, 8, No. 2, 1971, pp. 276-281.

12. J. N. D. GUPTA, *An Improved Combinatorial Algorithm for the Flowshop Scheduling Problem*, Operations Research, 19, No. 6, 1971, pp. 1753-1758.

13. J. N. D. GUPTA, *Optimal Schedu:ling in a Multi-Stage Flowshop*, A.I.I.E. Transactions, 4, No. 2, 1972, pp. 238-243.

14. S. M. JOHNSON, *Optimal Two-and Three Stage Production Schedules with Set-Up Times Uncluded*, Naval Research Logistics Quarterly, 1, No. 1, 1954, pp. 61-68.

15. G. B. MCMAHON, *Optimal Production Schedules for Flowshops*, Canadian Operations Research Society Journal, 7, 1969, pp. 141-151.

16. E. S. PAGE, *An Approach to the Scheduling of Jobs on Machines*, Journal of Royal Statistical Society, Series B, 23, No. 2, 1961, pp. 484-492.

17. D. S. PALMER, *Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time—A Quick Method of Obtaining a Near Optimum*, Operational Research Quarterly, 16, No. 1, 1965, pp. 101-107.

18. R. L. SISSON, *Sequencing Theory*, Chapter 7 in Progress in Operations Research, R. L. ACKOFF (ed.) 1, John Wiley and Sons, Inc., New York, 1961, pp. 295-325.

19. R. D. SMITH and R. A. DUDEK, *A General Algorithm for the Solution of the n-Job, m-Machine Sequencing Problem of the Flowshop*, Operations Research, 15, No. 1, 1967, 71-82 (*Also see* their Errata, Operations Research, 17, 1969, p. 756).

20. W. SZWARC, *Elimination Methods in the $m \times n$ Sequencing Problem*, Naval Research Logistics Quarterly, 18, No. 3, 1971, pp. 295-305.

21. W. SZWARC, *Optimal Elimination Methods in the $m \times n$ Flowshop Scheduling Problem*, Operations Research, 21, No. 6, 1973, pp. 1250-1259.

juin 1976.