MANAGEMENT SCIENCE
Vol. 16, No. 10, June, 1970
Printed in U.S.A.

A HEURISTIC ALGORITHM FOR THE n JOB, m MACHINE SEQUENCING PROBLEM* †

HERBERT G. CAMPBELL, RICHARD A. DUDEK AND MILTON L. SMITH

Texas Tech University

This paper describes a simple algorithm for the solution of very large sequence problems without the use of a computer. It produces approximate solutions to the n job, m machine sequencing problem where no passing is considered and the criterion is minimum total elapsed time. Up to m-1 sequences may be found.

Introduction

The search for a solution to the problem of finding an optimal or near optimal sequence of jobs being scheduled in a flow shop type situation has given consideration to both exact and approximate techniques of solution. Exact techniques, which usually require an electronic computer, have been developed to minimize some well defined criterion on problems involving a limited number of jobs [2, 5, 6, 10].

Sisson [9] has pointed out that the researcher must be concerned not only with obtaining an optimal solution but also with the practical and economical application of the solution technique. It is the second aspect of the problem which has led to the consideration of approximate methods. At this point in time companies with sequencing problems involving large numbers of jobs and machines must use approximate methods while awaiting further development of exact techniques or faster and more economical computers. Another reason to investigate approximate methods is that the procedural steps can be kept simple enough so that the problem solver does not lose sight of the overall view of the problem, thus enabling man to make the best use of his intuition and judgment. See [4].

The procedure to be developed in this paper can be calculated entirely by hand and will give approximate solutions to any size of n job, m machine problem. If a computer is used, computation time even for large problems is short. Not only does it permit solution of problems far beyond the present capability of exact techniques, but its effectiveness is such that some companies may find the approximate solution to be more economical even for those problems small enough to be handled by exact methods.

The algorithm is applied to the processing of n jobs through m machines with each job following the same technological order of machines. The passing of jobs is not considered by the algorithm; however a passing procedure specified by Palmer [8] may be employed to further refine the sequence generated by this algorithm. In this development, machine order will be A, B, C, . . . , M. The usual assumptions which have been given in earlier literature apply [2, 3, 4, 9, 11].

During our study two approaches were used. The first involved a procedure building a sequence by viewing total processing time as being made up of three parts, a fixed part in the center with a variable part on each end. The procedure then minimized

* Received October 1967; revised April 1968 and November 1969.
† Partially supported by National Science Foundation Grant GK-1156.

the two variable portions of the total time. The second approach involved an extension of the S. M. Johnson [7] procedure applied to large problems.

Sequences found utilizing an algorithm developed from the first procedure gave total elapsed times only slightly smaller than those found when applying an extension of the Johnson procedure. Since procedural simplicity was an objective of the research, we used an extension of the simpler Johnson procedure modified to handle more than two jobs.

Statement of the Algorithm

The algorithm presented here provides for the generation of up to $m-1$ sequences. Feasible sequence generation is accomplished in the following manner: let t_{ji} where $j = 1, 2, \cdots, n$, and $i = 1, 2, \cdots, m$ represent the time for processing the j^{th} job on the i^{th} machine in an n-job, m-machine problem. Then p, where $p \leqq m-1$, auxiliary n-job, 2-machine problems can be defined as follows. In the k^{th} auxiliary problem:

$\theta_{j1}^{k} = \sum_{i=1}^{k} t_{ji}$ = the processing time for the j^{th} job on machine 1 (M1),

$\theta_{j2}^{k} = \sum_{i=m+1-k}^{m} t_{ji}$ = the processing time for the j^{th} job on machine 2 (M2).

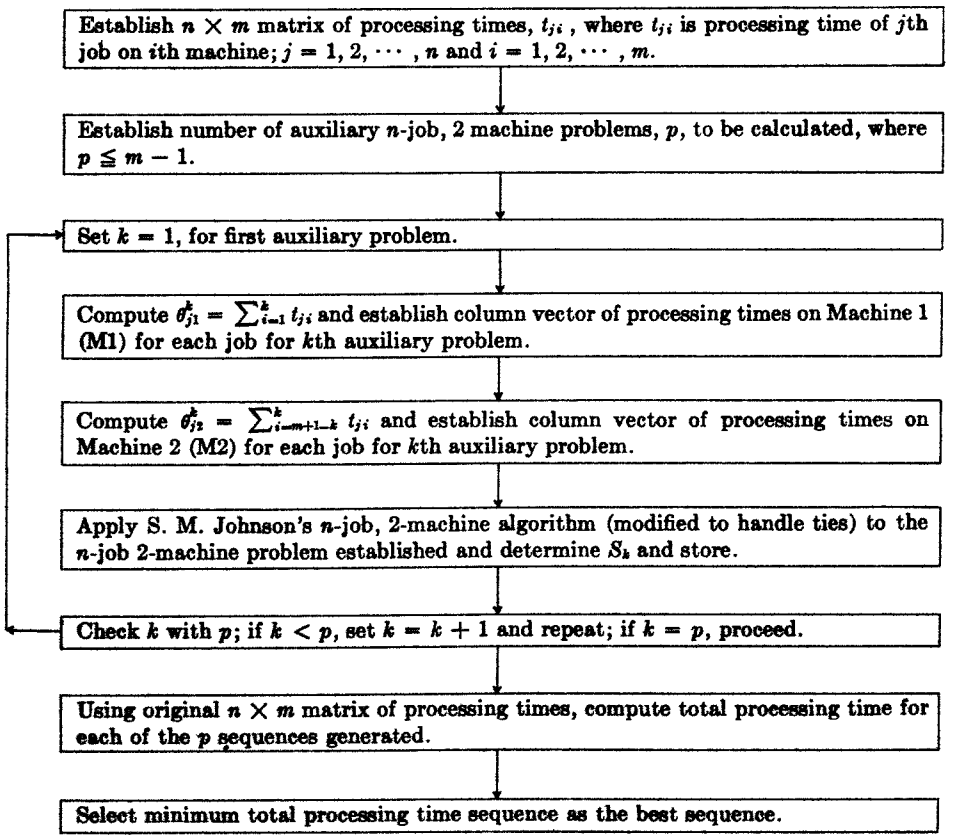

FIGURE 1. Flow Chart for Campbell-Dudek Algorithm

It should be noticed that θ_{j1}^{k} is the sum of processing times of job j on machines A through k; also θ_{j2}^{k} is the sum of processing times of job j on the last k machines in the technological order. Then S. M. Johnson's n-job, 2-machine algorithm modified to handle ties is used to determine the optimal sequences $S_1, S_2, \cdots, S_p$ for the p auxiliary problems. The best sequence among these is chosen on the basis of minimum total processing time on the original m machines. See Figure 1 for a flow chart of the algorithm.

In using the S. M. Johnson [7] n-job, 2-machine algorithm, let the first machine be M1 and the second M2. Select the smallest processing time in the two column processing time matrix; i.e., min $M1_1, \cdots, M1_n, M2_1, \cdots, M2_n$]. If there is a tie, select any within the tie set. (*Note.* This will be modified below). If the minimum processing time is $M1_g$, do the g^{th} job first. If it is $M2_h$ do the h^{th} job last. Remove the processing times of the sequenced job from the processing time matrix. Apply this procedure until all jobs have been assigned to a sequence position. The resulting sequence will minimize total processing time for this specific 2-machine problem.

The breaking of ties is done in a manner that by observation tends to minimize total make-span. Ties as used here are those occurring in a row or column of any auxiliary problem. When a tie occurs, a choice must be made between two positions for each job involved. If the tie occurs in the k^{th} auxiliary problem let the choice between these two positions be determined by the $k - 1^{st}$ auxiliary problem. If the tie cannot be broken there, proceed in order through auxiliary problems $k - 2$, $k - 3$, $\cdots$, 1. If it still cannot be broken, try breaking it with auxiliary problems $k + 1$, $k + 2, \cdots, m - 1$. If the tie remains unbroken, build two sequences from this point.

Example of an 8-Job, 7-Machine Problem

Jobs	A	B	C	D	E	F	G
1	13	79	23	71	60	27	2
2	31	13	14	94	60	61	57
3	17	1	ϵ	23	36	8	86
4	19	28	10	4	58	73	40
5	94	75	ϵ	58	ϵ	68	46
6	8	24	3	32	4	94	89
7	10	57	13	1	92	75	29
8	80	17	38	40	66	25	88

Note. The elements of value ϵ require zero processing time but must pass the machine.

"p" auxiliary two-stage problems will be worked. For the first auxiliary ($k = 1$) problem, the k^{th} two column processing time matrix is established as derived earlier. In this case it will be the processing times listed in the problem for machine A and machine G as the processing times on M1 and M2, respectively. Now applying the S. M. Johnson algorithm, $M2_1$ is minimum so sequence job 1 last and remove the job 1 processing time from the processing time matrix. Next, $M1_6$ is minimum so sequence job 6 first and remove the processing times from the matrix. Continuing until all jobs have been assigned a sequence position yields the sequence 6 7 3 4 2 8 5 1 with a total processing time of 618 and the minimal total time sequence for this auxiliary problem. Continuing in this way, setting $k = 2, 3, 4, 5$, and 6 six sequences will be generated. Development of the sequence for $k = 4$ is shown below. The processing times on M1

are found by adding times on machines A, B, C, and D; the times on M2 are the sums of times on machines D, E, F, and G. These times are given below.

	Machine	
Job	M1	M2
1	186	160
2	152	272
3	41	153
4	61	175
5	227	172
6	67	219
7	81	197
8	175	219

The algorithm generates the sequence 3 4 6 7 2 8 5 1 with total processing time of 632. The six sequences generated with k equal to 1 through 6 are listed below:

k^{th} Seq.	Sequence	Total Processing Time
1	6 7 3 4 2 8 5 1	618
2	3 6 2 4 7 8 5 1	628
3	3 6 4 2 7 8 5 1	596
4	3 4 6 7 2 8 5 1	632
5	6 3 4 7 2 8 1 5	605
6	3 6 4 7 8 2 1 5	595

The sixth sequence found has the least total processing time so it is selected. Using the Smith-Dudek algorithm [10] it was determined that there is one optimal sequence to this problem with a total time of 584 hours. It is 3 6 4 7 2 8 1 5.

The error calculation adopted here is based on the percent deviation of the algorithm's best sequence time from the optimal solution time. Thus, in the example just cited the error is

$$\frac{595 - 584}{584} 100 = 1.9\%$$

Verification

The empirical verification of the approximate procedure developed here is based upon the solution of 340 problems ranging in size from 3 jobs and 3 machines to 8 jobs, 5 machines; and 10 problems ranging from 20 jobs, 20 machines to 60 jobs, 30 machines. Processing times were selected randomly from a rectangular distribution ranging from 01 to 99. Only problems of size 8 jobs or less were considered in the statistical determination of effectiveness due to the difficulty in obtaining exact solutions for the large problems. The larger problems were solved, however, to determine the time required for solution and to make comparisons with solutions from another approximate algorithm.

Expected Error

Table 1 gives the distribution of error in terms of percent deviation the algorithm best sequence time was from the optimal sequence time for a sample of 20 problems

TABLE 1

Distribution of Error in Terms of Percent Deviation of Algorithm Best Sequence Time from Optimal Sequence Time

n	m	Sample Size	Numbers of Problems with Percent Deviation of			Range (%)	Average Error (90)	
			0%	0% < to ≤ 5%	>5%			
3	3	20	17	3	0	0–2.9	0.19	
4	3	20	15	3	2	0–22.6	1.80	
5	3	20	13	6	1	0–10.4	1.13	1.65
6	3	20	12	2	6	0–15.9	3.57	
7	3	20	13	5	2	0–8.3	1.17	
8	3	20	12	5	3	0–14.7	2.03	
3	5	20	19	1	0	0–1.4	0.07	
4	5	20	9	7	4	0–41.3	3.95	
5	5	20	11	5	4	0–9.2	2.02	3.77
6	5	20	4	7	9	0–23.2	5.24	
7	5	20	2	9	9	0–14.3	5.18	
8	5	20	3	7	10	0–25.4	6.18	
3	7	20	20	0	0	0	0	
4	7	20	14	5	1	0–5.3	0.68	
5	7	20	7	9	4	0–10.0	2.27	2.13
6	7	20	3	9	8	0–11.4	3.64	
7	7	20	2	12	6	0–12.6	4.07	
Total		340	176	95	69	Average	2.54	

in each category of problems considered. The mean error for each problem group in terms of percentage of optimal time is shown. Also shown are mean errors for problem groups with constant m.

Similar trials were performed with processing times selected from normal and exponential distributions. Effectiveness of the algorithm was, on the average, better with these problems than it was with the problems used in forming Table 1.

Comparison to Palmer Algorithm

The Palmer [8] algorithm based upon a ranking of slope indices of jobs has several similarities with the algorithm presented here. One sequence per problem is found by the Palmer algorithm. Since the Palmer algorithm is an approximate method, a comparison of performance of the two algorithms was made after the Palmer algorithm was programmed to run on the IBM 7040. Two measures of comparison, effectiveness and computation time, were used in conjunction with two sets of problems.

The first set of problems contained eight groups of 20 problems per group with known optimal sequences. The eight groups ranged in size from 3 job—4 machine to 6 job—4 machine problems and 3 job—6 machine to 6 job—6 machine problems. All processing times were selected randomly from a rectangular distribution ranging from 001 to 999. The problems were selected from 4 machine and 6 machine problems in order to also give further verification to Table 1 discussed above.

Percent error for both algorithms was computed in the manner discussed earlier. Table 2 contains data on the 160 problems in the first set. The Campbell-Dudek se-

quence times averaged 1.38% greater than optimal sequence times as compared to an average error of 4.58% for the Palmer sequence times.

The second set of problems contained ten problems of large sizes and were solved only with the Campbell-Dudek and the Palmer algorithms; the problem size precluded use of enumeration or optimizing algorithms. Problems in the second set were formed with random numbers drawn from a rectangular distribution with a range of 00 to 99. For each of the problems, the Campbell-Dudek algorithm produced only the first six sequences.

As in these problems the optimal solution was not known, the percent improvement of the results of our algorithm compared with that of Palmer's was calculated, and this is shown in Table 3. In nine out of the ten problems, the Campbell-Dudek sequence time was lower than that of the Palmer solution.

Computer computation time comparisons were made by solving identical sets of problems of various sizes with each algorithm. An IBM 7040 with online printer was used; computation time included data input, data output, and solution output. Computation times for both algorithms were very insensitive to the number of machines;

TABLE 2

Error Comparison on Problems with Known Optimal Sequence Times

Problem Size		No. Problems	Campbell-Dudek				Palmer			
n	m		No. Optimals	Largest Error (%)	Average Error (%)		No. Optimals	Largest Error (%)	Average Error (%)	
3	4	20	17	8.1	.61	1.54	11	17.5	2.68	4.58
4	4	20	15	10.1	1.74		6	19.2	6.08	
5	4	20	10	13.8	2.56		4	15.6	4.95	
6	4	20	12	11.5	1.24		2	15.1	4.61	
3	6	20	18	1.4	.12	1.22	14	18.7	2.77	4.58
4	6	20	15	4.1	.56		8	13.9	3.13	
5	6	20	11	6.9	2.01		2	20.2	5.90	
6	6	20	6	9.0	2.18		3	18.2	6.52	
Totals		160	104		1.38		50		4.58	

TABLE 3

Comparison on Problems with Unknown Optimal Sequence Times

n	m	Sequence Time		% Improvement
		C-D	Palmer	
20	20	2452	2712	10.60
20	20	2496	2542	1.84
20	20	2390	2382	−0.34
20	20	2422	2484	2.56
40	30	4458	4574	2.60
40	30	4597	4634	0.80
40	30	4496	4600	2.31
40	30	4475	4665	4.25
60	30	5747	5841	1.64
60	30	5849	5997	2.53

TABLE 4
Computer Computation Time

Number of Jobs	Average Computation Time (min)	
	C-D	Palmer
8	.055	.029
10	.067	.037
20	.195	.100
40	.752	.223
60	1.806	.347

thus only the number of jobs was considered. Table 4 contains the computation time data. The Palmer algorithm produces one sequence per problem while the Campbell-Dudek algorithm produces six. As a result, Palmer computation times are smaller. However, with only 1.5 minutes difference in time on a 60 job problem, the choice between the two algorithms probably would be made on factors other than computation time.

Calculation Time

Hand-calculation time for the approximate method varies from 2.5 minutes for a 3-job, 3-machine problem to 31.9 minutes for a 10-job, 15-machine problem. These times include finding only two trial sequences and computing their total time. Since the number of individual arithmetic operations in any size problem are known, the hand-calculation time for any size problem can be predicted. The following times are predicted for the larger problems which were not solved by hand:

20-job, 20-machine—130 minutes,
30-job, 20-machine—130 minutes,
60-job, 30-machine—380 minutes, and
60-job, 60-machine—740 minutes.

Economy Considerations

The choice between exact sequencing procedures and approximate procedures exists only for limited problem sizes. Computational experience with the Smith-Dudek [10] algorithm has indicated that problems with greater than 10 jobs require extended computation time precluding finding optimal solutions. Therefore, until improved exact procedures are developed, approximate methods must be used.

If problems are of a size permitting a choice between approximate and exact procedures, several factors must be considered. Approximate methods will give results quickly and with less computation costs than with an exact algorithm. Conversely, the approximate algorithm results will not be as accurate as results from an exact algorithm. The two factors, computational costs and costs of nonoptimal solutions, appear to be of primary importance when choosing between approximate and exact sequencing procedures. Since each situation will be unique, an economic analysis of the costs involved should be made before a procedure is selected.

Conclusions

The approximate sequencing method described provides a practical solution to large sequencing problems that cannot be solved by exact procedures. Solutions produced by this algorithm are optimal or near-optimal and are easily and quickly produced.

Up to $m - 1$ sequences can be found; if a computer is available, the expected error could be reduced by finding most or all of the $m - 1$ sequences. If computations are made by hand, less than $m - 1$ sequences might be considered adequate even though some loss of effectiveness is incurred.

References

1. CAMPBELL, HERBERT G., "A Heuristic Solution Technique for Near Optimal Production Schedules," Unpublished Masters Thesis, Texas Tech College, 1966, Advisor—Richard A. Dudek.
2. DUDEK, RICHARD A. AND TEUTON, OTTIS F., JR., "Development of M-Stage Decision Rule for Scheduling n Jobs Through M Machines," *Operations Research*, Vol. 12, No. 3 (May-June 1965), pp. 471–497.
3. GIFFLER, B., "Mathematical Solution of Explosion and Scheduling Problems," IBM Research Report RC-118 (June 18, 1959), IBM Research Center, Business Systems Research, Yorktown Heights, N. Y., p. 1.
4. GIGLIO, R. J. AND WAGNER, H. M., "Approximate Solutions to the Three-Machine Sequencing Problems," *Operations Research*, Vol. 12 (1964), pp. 305–324.
5. IGNALL, EDWARD AND SCHRAGE, LINUS, "Application of the Branch and Bound Technique to Some Flow-Shop Scheduling Problems," *Operations Research*, Vol. 13, No. 3 (May-June 1965), pp. 400–412.
6. KARUSH, W., "A Counter-Example to a Proposed Algorithm for Optimal Sequencing of Jobs," *Operations Research*, Vol. 13 (1965), pp. 323–325.
7. JOHNSON, S. M., "Optimal Two and Three Stage Production Schedules with Set-Up Times Included," *Nav. Res. Log. Quart.*, Vol. 1, No. 1 (March 1954), pp. 61–68.
8. PALMER, D. S., "Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time —A Quick Method of Obtaining a Near Optimum," *Operational Research Quarterly*, Vol. 16, No. 1 (March 1965), pp. 101–107.
9. SISSON, R. L., "Sequencing Theory," Chapter 7 *Progress in Operations Research*, Russel L. Ackoff (ed.), Vol. 1, John Wiley and Sons, Inc., New York, 1961, pp. 295–325.
10. SMITH, RICHARD AND DUDEK, RICHARD A., "A General Algorithm for Solution of the n-Job, M-Machine Sequencing Problem of the Flow Shop," *Operations Research*, Vol. 15 (1967), pp. 71–82.
11. WAGNER, H. M., "An Integer Linear Programming Model for Machine Scheduling," *Nav. Res. Log. Quart.*, Vol. 6 (1959), pp. 131–140.