Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Letter to the Editor—An Experimental Investigation and Comparative Evaluation of Flow-Shop Scheduling Techniques

Said Ashour,

To cite this article:
Said Ashour, (1970) Letter to the Editor—An Experimental Investigation and Comparative Evaluation of Flow-Shop Scheduling Techniques. Operations Research 18(3):541-549. https://doi.org/10.1287/opre.18.3.541

Published in Operations Research on June 01, 1970 as DOI: 10.1287/opre.18.3.541.
This article has not been copyedited or formatted. The final version may differ from this version.

Letters to the Editor

AN EXPERIMENTAL INVESTIGATION AND COMPARATIVE EVALUATION OF FLOW-SHOP SCHEDULING TECHNIQUES

Said Ashour

Kansas State University, Manhattan, Kansas

(Received January 2, 1969)

This note is concerned with the solution of the flow-shop scheduling problem where all jobs have the same machine ordering. Because of the combinatorial nature of this problem, most practical situations remain unsolved. Various techniques such as switch and check, branch and bound with and without backtracking, modified decomposition, and rounded linear programming have been proposed by several investigators. However, no comparative evaluation of these procedures has been previously made. This note investigates the solutions obtained by these procedures considering both the quality of the solutions and the computational efficiency. Extensive experimentation has been conducted and significant results are reported. The effects of changes in the size of problems on the above criteria are also included.

THE FLOW-SHOP scheduling problem consists of a number of jobs to be performed through various machines. Each job has a number of operations to be processed on the machines in a specified machine ordering that is the same for all jobs. The processing time for each of these operations is known. The objective is to schedule the jobs on the machines so that the sequence is optimal with respect to the minimum schedule time. The usual assumptions made are (1) all jobs are known and ready for processing; (2) all jobs are processed once and only once on each machine; and (3) in-process inventory is allowed. A complete set of assumptions is provided in references 1, 6, and 7.

In view of the combinatorial nature of the flow-shop problem, an exclusive enumeration of all possible sequences for most practical problems is impossible, even though computational abilities are increasing concomitantly with the development of faster and more powerful computers. However, following Theorems 5-1 and 5-2 in CONWAY ET AL,[6] it is sufficient in flow-shop problems having up to three machines to consider only the permutation sequences, that is, the sequences in which all jobs are processed in identical order on all the machines.

The general flow-shop problem has attracted the attention of many researchers over the last decade. The purpose has been to develop efficient algorithms for arriving at optimal solutions. In this paper, the so called optimal-producing methods explicitly assume that the same sequence of jobs will be followed on all machines. This assumption is restrictive for problems having more than three

Published in Operations Research on June 01, 1970 as DOI: 10.1287/opre.18.3.541.
This article has not been copyedited or formatted. The final version may differ from this version.

machines, since it is not sufficient to consider the $J!$ sequences only, where J is the total number of jobs. It should be pointed out, however, that the optimal permutation solution will be simply referred to as the optimal solution.

Although a number of optimal- and suboptimal-producing techniques has been proposed, comparative evaluation of these procedures has not previously been made. The purpose of this paper is to investigate the solutions obtained by switch and check,[7,11,16] branch and bound with and without backtracking,[2,3,5,9,12] modified decomposition,[1,4] and rounded linear programming.[8] All algorithms except rounded linear programming were programmed in FORTRAN IV for IBM 360/50 computer. Techniques are compared considering both the quality of solutions and the computational efficiency.

REVIEW OF SOLUTION APPROACHES

THE BASIC concepts of the approaches under consideration in this paper are reviewed very briefly. The switch and check proposed by SMITH AND DUDEK[14] relies on changing one partial sequence to another through switching the jobs around. Job- and sequence-dominance checks are considered to eliminate many of the alternative partial sequences at each sequence-position.

The branch-and-bound technique has been developed for the solution of flow-shop problems by IGNALL AND SCHRAGE[9] and LOMNICKI[12] independently. The basic concepts of this technique are the partitioning of the set of possible solutions into necessarily smaller subsets and the application of a lower bound on the schedule time to identify a subset containing an optimal solution. A backtracking process may be embodied in this technique to guarantee optimality; however, a solution that may or may not be optimal can be obtained without backtracking. The branch-and-bound algorithm used in this research employs the lower bound of Lomnicki.[12] This is based on a comparative evaluation of five lower bounds conducted by ASHOUR AND QURAISHI.[2]

The decomposition approach developed by Ashour[1] is one of partitioning the original set of jobs into a series of a smaller, more manageable subsets, each of which consists of half the total number of jobs. Each of these subsets is solved by one of the existing techniques to determine its best sequence. The schedule times of the subsets are then combined to form a schedule time for all jobs of the original set. This approach has been modified by improving the partitioning and combining phases in addition to the use of the branch-and-bound technique without backtracking for obtaining the solution of the subsets.[4]

Finally, the rounded linear programming solutions have been obtained by GIGLIO AND WAGNER[8] employing the simplex method (ignoring the integer constraints). The fractional values obtained are rounded to integers.

COMPUTATIONAL RESULTS

THIS SECTION is devoted to the computational experiments conducted and their findings. The experiments consist of 550 problems that were selected with six to twelve jobs and three to five machines. The entries of the processing-time matrices

Published in Operations Research on June 01, 1970 as DOI: 10.1287/opre.18.3.541.
This article has not been copyedited or formatted. The final version may differ from this version.

were generated at random from a uniform distribution between 1 and 30, inclusive. The quality of solutions and computer times for all experiments are summarized and discussed below. The quality of a solution obtained by one of the suboptimal-producing methods, referred to as efficiency, is defined as the quotient of the optimal schedule time and that solution.

Experiment I consists of 100 problems, each of which has six jobs and three machines. It was mainly designed to compare the efficiencies obtained by switch and check, branch and bound (with and without backtracking), and modified decomposition techniques with those previously obtained by a rounded linear programming procedure from a similar experiment.[8] Although the data of both experiments were produced by different random generators, the sample average

TABLE I

EFFICIENCIES AND CORRESPONDING NUMBER OF PROBLEMS SOLVED BY SUBOPTIMAL-PRODUCING TECHNIQUES

Efficiency	Branch-and-bound without back tracking	Modified decomposition	Rounded linear programming
1.00	45	43	8
0.95	31	49[a]	19
0.90	15	6	22
0.85	7	2	26
0.80	1		11
0.75	1		8
0.70			4
0.65			1
0.60			1
	100	100	100

[a] Illustration: The number of problems solved by decomposition having $0.95 \leqq$ efficiency < 1.00 is 49.

and standard deviation of the optimal schedule times found in our experiment are 126.58 and 18.38, respectively compared to 125.95 and 18.46 obtained in the similar experiment. Thus, it would be permissible to compare the results of both experiments. The efficiencies with their corresponding number of problems for the suboptimal-producing methods are shown in Table I. Clearly, the efficiency of both optimal-producing methods is 1.0. Table I shows a clear inferiority of the rounded-linear-programming procedure. From the computational point of view, it has been stated that the solutions do not seem sufficiently close to the optimal schedule times to warrant the amount of computational times involved.[8] Rounded linear programming solutions are, therefore, not obtained for problems of larger sizes.

The number of optimal sequences generated by switch and check was compared with those found by a complete enumeration. Of the 720 possible sequences in each of the 100 problems, the number of optimal sequences ranged from 1 to 120,

TABLE II

COMPARATIVE RESULTS OBTAINED BY VARIOUS TECHNIQUES

Experiment no.	Problem size	No. of problems	Optimal-producing techniques															
			Switch-and-check								Branch-and-bound with backtracking							
			Computer time[a]				Number of generated sequences				Computer time[a]				Number of explored nodes			
			Min.	Max.	Mean	Std. Dev.	Min.	Max.	Mean	Std. Dev.	Min.	Max.	Mean	Std. Dev.	Min.	Max.	Mean	Std. Dev.
I	6×3	100	0.35	14.52	3.36	3.07	3	44	15.74	8.86	0.64	13.72	2.34	2.48	20	477	77.98	86.89
II	6×4	50	1.47	39.59	9.81	7.94	11	90	33.56	16.59	0.78	17.60	4.57	4.38	20	440	117.00	112.08
III	6×5	50	1.08	23.20	7.43	5.42	16	109	46.16	21.95	0.87	18.58	5.66	4.50	20	425	129.40	103.00
IV	7×3	50	0.53	70.97	19.59	18.59	3	60	28.98	14.43	0.99	53.47	7.49	11.06	27	1465	205.26	302.91
V	7×4	50	0.59	91.61	26.20	22.30	9	129	59.56	28.90	1.16	56.62	10.43	12.75	27	1322	243.44	297.67
VI	7×5	50	4.17	320.85	80.67	66.42	30	223	102.40	47.16	1.36	132.71	16.71	21.58	27	2643	332.76	429.89
VII	8×3	50	1.68	435.81	53.68	67.73	10	94	35.00	17.35	1.53	268.98	19.28	51.32	35	6148	440.71	1172.93
VIII	8×4	50	5.66	514.62	140.62	124.63	15	171	87.74	37.45	1.81	253.04	52.73	77.50	35	4884	1031.56	1495.85
IX	8×5	50	54.42	1880.35	388.74	377.21	57	363	165.24	83.91	2.20	303.04	54.18	62.80	35	4793	862.04	999.23
X	10×3	25	18.80	3376.15	464.84	754.87	23	130	49.32	23.38	3.40	342.33	45.23	101.40	54	6392	845.00	1895.61
XI	12×3	25	49.93[b]	3072.30[b]	1228.50[b]	1063.95[b]	25[b]	58[b]	40.20[b]	11.74[b]	6.50	42.17	12.17	10.58	77	522	150.64	130.93

[a] Computation time on IBM 360/50 computer, in seconds.

[b] Indicates solutions not obtained for 9 problems. The average results correspond to the remaining 16 problems.

Experiment no.	Problem size	No. of problems	Suboptimal-producing techniques															
			Branch-and-bound without backtracking								Modified Decomposition							
			Computer time[a]				Efficiency				Computer time[a]				Efficiency			
			Min.	Max.	Mean	Std. Dev.	Min.	Max.	Mean	Std. Dev.	Min.	Max.	Mean	Std. Dev.	Min.	Max.	Mean	Std. Dev.
I	6×3	100	0.68	0.76	0.70	0.02	0.808	1.0	0.963	0.046	3.11	3.92	3.16	0.11	0.884	1.000	0.958	0.031
II	6×4	50	0.85	1.00	0.90	0.04	0.844	1.0	0.952	0.048	3.25	3.74	3.52	0.03	0.800	1.000	0.937	0.043
III	6×5	50	0.93	1.02	0.97	0.02	0.742	1.0	0.946	0.053	4.13	4.68	4.17	0.05	0.805	0.989	0.929	0.040
IV	7×3	50	1.00	1.43	1.12	0.09	0.825	1.0	0.950	0.050	15.19	17.25	15.62	0.48	0.850	1.000	0.937	0.038
V	7×4	50	1.23	1.72	1.28	0.09	0.833	1.0	0.936	0.048	28.18	29.79	28.30	0.08	0.837	1.000	0.923	0.039
VI	7×5	50	1.46	4.52	1.60	0.43	0.803	1.0	0.934	0.054	23.83	24.92	24.21	0.08	0.716	1.000	0.919	0.480
VII	8×3	50	1.60	2.00	1.70	0.07	0.755	1.0	0.964	0.051	16.63	17.18	16.77	0.39	0.883	1.000	0.961	0.035
VIII	8×4	50	1.84	2.61	1.98	0.14	0.794	1.0	0.939	0.057	20.19	21.85	20.37	0.11	0.870	0.994	0.932	0.036
IX	8×5	50	2.21	2.30	2.26	0.02	0.848	1.0	0.929	0.041	27.15	27.90	27.62	0.02	0.838	0.972	0.918	0.032
X	10×3	25	3.32	3.59	3.44	0.07	0.857	1.0	0.960	0.041	20.85	21.79	21.38	0.07	0.782	1.000	0.939	0.054
XI	12×3	25	6.52	7.64	6.93	0.24	0.849	1.0	0.953	0.050	37.82	38.89	38.16	0.12	0.893	1.000	0.957	0.042

[a] Computation time on IBM 360/50 computer, in seconds.

Published in Operations Research on June 01, 1970 as DOI: 10.1287/opre.18.3.541.
This article has not been copyedited or formatted. The final version may differ from this version.

with an average of 15.3. Switch and check evaluated 16.3 sequences on the average, finding from 1 to 10 optimal sequences with an average of 2.6: a rather small fraction of the total, especially in cases where many optima exist.

For a complete assessment of the various techniques, comparative results obtained from Experiments I through XI are displayed in Table II. All four techniques were run on all 550 problems. Detailed pairwise comparisons of techniques are discussed below.

Optimal-Producing Techniques

As mentioned earlier, switch and check and branch and bound produce a permutation optimal schedule. This solution is necessarily global optimal only for flow-shop problems with up to three machines. Table II shows the average computer time and number of sequences generated by switch and check and the average computer time and number of nodes explored by branch and bound.

As can be seen from Table II, the computation time increases rapidly in both techniques because of the combinatorial nature of the problem. Based on the data of all experiments one would be inclined to assume that the passage from J to $J+1$ jobs increases the amount of computation time in switch and check and branch and bound by a factor of about 6 and 2.5, respectively. With the same number of jobs but increasing the number of machines from M to $M+1$, the amount of computational time spent in the former increases with a higher rate than that in the latter. The reason is that in filling each sequence position by switch and check, $M-1$ conditions must be computed for each of the job- and sequence-dominance checks. Similarly, in branch and bound M bounds must be computed for each node to find the corresponding lower bound.

Upon examining the results obtained from individual problems with each experiment, it was apparent that the number of sequences generated by switch and check and the number of nodes explored by branch and bound vary greatly from one problem to another. Consequently, the computation times vary greatly within each experiment—*see* the ranges for both techniques in Table II. The reason is that the criterion used in this research is the schedule time, which is a function of the processing times. Therefore, the partial sequences to be checked for dominance and the nodes to be explored depend on the processing times of the jobs on the machines.

It is of interest to note that the computer time is increased regardless of the number of sequences generated by switch and check. The obvious reason would be the increase in the number of partial sequences to be checked for dominance at each sequence position. Hence, the technique is not efficient, especially as the number of jobs increases. It should be pointed out that in Experiment XI—(12×3) problems—only 16 out of 25 problems were solved. We set a time limit of 3600 seconds in the computer program for solving each of the remaining problems by switch and check. The computer stopped earlier than the time limit in five cases without obtaining any solutions. The reason for this is that the dimensions of the FORTRAN variables corresponding to the partial sequences resulting from the job- and sequence-dominance checks exceeded the computer core. No solution was obtained for each of the remaining four problems within the time limit.

Published in Operations Research on June 01, 1970 as DOI: 10.1287/opre.18.3.541.
This article has not been copyedited or formatted. The final version may differ from this version.

It is quite surprising that, for the same experiment, branch and bound produced the optimal solutions in far fewer numbers of nodes, and hence less computer time than those obtained for smaller-size problems. One logical justification for this observation could be that, in large problems, many optimal solutions may exist, and, in general, a solution is obtained with less searching of the scheduling tree than might be expected.[13] In general, however, in terms of computer time, the data obtained from all experiments shows the clear superiority of branch and bound.

Suboptimal-Producing Techniques

In order to make branch and bound more attractive from the computational feasibility point of view, only the least number of nodes, which is approximately $[J(J+1)]/2$, was explored. Thus the solution obtained, which may or may not be optimal, is referred to as a branch-and-bound solution without backtracking. The modified decomposition method, which also does not guarantee optimality, was run on the same sets of problems. A complete set of arrangements was evaluated to obtain the modified-decomposition solutions for problems having up to 8 jobs. However, for problems with 10 and 12 jobs the modified decomposition solutions are obtained by generating and evaluating only a subset from the set of all arrangements that are $10!/(5!)^2$ and $12!(6!)^2$ or 252 and 924, respectively. The numbers of arrangements evaluated in both experiments were 220 and 108. In problems having 7 jobs, a job is added with zero processing times on all machines. Therefore, the resulting set of 8 jobs can be partitioned into two subsets each having 4 jobs.

As can be seen in Table II, the solutions obtained by these two suboptimal-producing techniques yield high efficiency. The average efficiency ranges between 0.92 and 1.00. In terms of quality of solutions, there is no significant difference between the two procedures. However, modified decomposition is inferior to branch and bound without backtracking in terms of computational time. In general, the computer times spent to obtain the solutions by modified decomposition were less than those obtained by the optimal-producing techniques. In comparing branch and bound without backtracking with the optimal-producing techniques, the former is computationally feasible for larger problems but at the expense of the quality of solution.

Observe that the computer times required to obtain the modified decomposition solutions have small variations from one problem to another, because of the fixed number of arrangements evaluated for each of the problems with the same number of jobs. Similarly, in branch and bound without backtracking only one node is explored at each level of the scheduling tree. There is, therefore, no significant variation in the number of explored nodes, and, thus in the computer time spent.

SUMMARY AND CONCLUSIONS

THE BASIC objective of this research was to compare various scheduling procedures for the solution of the flow-shop problem. A brief review of the techniques subject to comparison in this paper was given. Extensive experimentation was performed

to carry out the investigation. The techniques were compared on the basis of two criteria: quality of solution, and computer efficiency.

Switch and check and branch and bound with backtracking produce an optimal solution but at the expense of excessive amounts of computational effort. Thus the optimal-producing techniques are computationally feasible for only small-size problems. Branch and bound without backtracking produces quite good solutions for quite modest computational times. Modified decomposition produces equally good solutions, but requires more computer time, though still considerably less than that required by the optimal-producing techniques. Modified decomposition, however, increases the capability of the optimal-producing techniques for solving larger or more practical problems. Although linear programming has a potentiality in the future, the resulting solutions do not seem sufficiently close to the optimal values to warrant the computational effort involved. Thus, this procedure is not comparable with others, at least for the time being.

ACKNOWLEDGMENT

THIS RESEARCH was supported in part by the Research Coordinating Council, Kansas State University, Manhattan, Kansas. This paper is a modified and revised version of ASHOUR.[3]

REFERENCES

1. S. ASHOUR, "A Decomposition Approach for the Machine Scheduling Problem," *International J. of Production Res.* **6**, 109–122 (1967).
2. ——— AND M. N. QURAISHI, "Investigation of Various Bounding Procedures for Production Scheduling Problems," *International J. of Production Res.* **7**, 249–252 (1969).
3. ———, "Comparison of Different Approaches for Solving Sequencing Problems," Internal Report, Kansas Engineering Experiment Station, Kansas State University, Manhattan, Kansas, January 1969.
4. ———, "A Modified Decomposition Algorithm for Scheduling Problems," submitted to *International J. of Production Res.* (1969).
5. A. BROWN AND Z. LOMNICKI, "Some Applications of the Branch-and-Bound Algorithm to the Machine Scheduling Problem," *Opnal. Res. Quart.* **17**, 173–186 (1966).
6. R. CONWAY, W. MAXWELL, AND L. MILLER, *Theory of Scheduling*, Addison-Wesley, Reading, Mass., 1967.
7. R. DUDEK AND O. TEUTON, JR., "Development of M-Stage Decision Rule for Scheduling n Jobs Through m Machines," *Opns. Res.* **12**, 471–497 (1964).
8. R. GIGLIO AND H. WAGNER, "Approximate Solutions to the Three-Machine Scheduling Problem," *Opns. Res.* **12**, 305–324 (1964).
9. E. IGNALL AND L. SCHRAGE, "Application of the Branch-and-Bound Technique to Some Flow-Shop Scheduling Problems," *Opns. Res.* **13**, 400–412 (1965).
10. S. JOHNSON, "Optimal Two- and Three-Stage Production Schedules with Setup Times Included," *Nav. Res. Log. Quart.* **1**, 61–68 (1954).

11. W. KARUSH, "A Counter Example to a Proposed Algorithm for Optimal Sequencing of Jobs," *Opns. Res.* **13,** 323–325 (1965).
12. Z. LOMNICKI, "A Branch-and-Bound Algorithm for the Exact Solution of the Three Machine Scheduling Problem," *Opnal. Res. Quart.* **16,** 89–107 (1965).
13. G. MCMAHON AND P. BURTON, "Flowshop Scheduling with Branch-and-Bound Method," *Opns. Res.* **15,** 473–481 (1967).
14. R. SMITH AND R. DUDEK, "A General Algorithm for Solution of the n-Job, M-Machine Sequencing Problem of the Flow Shop," *Opns. Res.* **15,** 71–82 (1967).

ON THE MERIT OF THE GENERALIZED ORIGIN AND RESTARTS IN IMPLICIT ENUMERATION

Harvey M. Salkin

Case Western Reserve University, Cleveland, Ohio

(Received September 12, 1968)

This note discusses the concept of starting a (zero-one) tree search at an integer solution obtained from the associated linear program and restarting it at improved feasible solutions. Computational experience indicates that the length of the enumeration is inversely proportional to the closeness of the origin to the minimal integer vector.

IN THIS note we are concerned with the zero-one integer programming problem:

Minimize	cy,	
constrained by	$Ay \leqq b,\ 0 \leqq y \leqq e,$	(1)
and	y integer.	(2)

Here A is an $m \times n$ matrix, and e is an n-column vector of ones.

Search algorithms (*see* e.g., references 1, 3, 5, or 8) may be described as enumerative methods that endeavor to explore nonredundantly the 2^n values of y. Each contains a bookkeeping scheme to keep track of the enumeration, and criteria that seek to eliminate large portions of the total search (to enumerate implicitly) by verifying that it would be useless to consider certain vectors as candidates for improved solutions.

Often, to gain nonnegativity in the cost row, the search is initiated from the point y^0, the origin, where $y_j^0 = 1$ if $c_j < 0$, otherwise $y_j^0 = 0$. (The bookkeeping schemes usually require that the enumeration be started from the point 0, which is accomplished by introducing the complementing variables $\bar{y}_j = 1 - y_j$ for the j with $y_j^0 = 1$.) Rather than commencing the search at the point determined by the negative costs (referred to as the 'natural' origin), one may first solve the integer