International Journal of Production Research
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

Bicriteria scheduling in a two-machine permutation flowshop

B. M. T. Lin [a] & J. M. Wu [b]

[a] Department of Information and Finance Management , Institute of Information Management National Chiao Tung University Hsinchu 300 , Taiwan

[b] Department of Information Management , National Chi Nan University , Nantou 545, Taiwan

Published online: 22 Feb 2007.

To cite this article: B. M. T. Lin & J. M. Wu (2006) Bicriteria scheduling in a two-machine permutation flowshop, International Journal of Production Research, 44:12, 2299-2312, DOI: 10.1080/00207540500446394

To link to this article: http://dx.doi.org/10.1080/00207540500446394

International Journal of Production Research,
Vol. 44, No. 12, 15 June 2006, 2299–2312

Taylor & Francis
Taylor & Francis Group

Bicriteria scheduling in a two-machine permutation flowshop

B. M. T. LIN*† and J. M. WU‡

†Department of Information and Finance Management, Institute of Information Management
National Chiao Tung University Hsinchu 300, Taiwan
‡Department of Information Management, National Chi Nan University,
Nantou 545, Taiwan

(*Revision received October 2005*)

In this paper we consider a production scheduling problem in a two-machine flowshop. The bicriteria objective is a linear combination or weighted sum of the makespan and total completion time. This problem is computationally hard because the special case concerning the minimization of the total completion time is already known to be strongly NP-hard. To find an optimal schedule, we deploy the Johnson algorithm and a lower bound scheme that was previously developed for total completion time scheduling. Computational experiments are presented to study the relative performance of different lower bounds. While the best known bound for the bicriteria problem can successfully solve test cases of 10 jobs within a time limit of 30 min, under the same setting our branch-and-bound algorithm solely equipped with the new scheme can produce optimal schedules for most instances with 30 or less jobs. The results demonstrate the convincing capability of the lower bound scheme in curtailing unnecessary branching during problem-solving sessions. The computational experience also suggests the practical significance and potential implications of this scheme.

Keywords: Flowshop; Makespan; Total completion time; Lower bound; Branch-and-bound algorithm

1. Introduction

Research on production scheduling seeks to develop systematic ways for allocating limited resources to tasks subject to specified requirements or constraints. In the scheduling literature, due to their practical significance and theoretical challenges, flowshop problems are among the most well-studied topics (Dudek *et al*. 1992, Reisman *et al*. 1997). Flowshops are widely adopted to describe the organizational operations process as well as inter-organizational relationships in industrial networks. Flowshop scheduling research was inspired by Johnson's (1954) seminal work that not only proposed the flowshop model but also provided an efficient algorithm that can produce a schedule with an optimal maximum completion time, or makespan, in a two-machine permutation flowshop. While makespan minimization can be optimally achieved by Johnson's algorithm, to find a schedule that has the smallest total flow time is, however, computationally challenging (Garey *et al*. 1976, Garey and Johnson 1979). In this paper, we consider a two-machine permutation flowshop scheduling

*Corresponding author. Email: bmtlin@mail.nctu.edu.tw

International Journal of Production Research
ISSN 0020–7543 print/ISSN 1366–588X online © 2006 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540500446394

problem in which the objective function is a weighted sum of the makespan and total flow time.

The Makespan is commonly adopted as a measure for machine utilization. Total flow time, defined as the sum of the durations in which jobs stay in the system, is another important measure. This measure can be interpreted from aspects such as the average WIP level within an organization and the average waiting time of customers. Roughly speaking, the two measures are related to efficiency management and customer service, respectively. Because the two measures are crucial to the management of resources and service quality, a general decision practice might bring them into consideration simultaneously to measure the quality of a schedule with different criteria. Multiple criteria considerations provide flexibility to decision makers. Furthermore, Dudek *et al*. (1992) even suggest that the absence of multiple criteria from flowshop scheduling may be one of the reasons for the practical applications of flowshop scheduling problems. For such multiple and bicriteria scheduling problems, the reader is refereed to Nargar *et al*. (1995) for a comprehensive survey. Adopting the established three-field notation (Graham *et al*. 1979), we denote the two-machine flowshop scheduling problem by $F2//\alpha\Sigma C_i + \beta C_{\max}$. The first field indicates a flowshop manufacturing system consisting of two machines. The third field specifies the objective function defined by weights α and β with $\alpha + \beta = 1$ and $0 \leq \alpha, \beta \leq 1$. Note that the flow time of a job is equal to its completion time whenever the job is available for processing from time zero onwards. In this paper, we use the total completion time instead of the total flow time. As a sequel, ΣC_i denotes the sum of completion times in the objective function. It is easy to recognize the computational intractability of $F2//\alpha\Sigma C_i + \beta C_{\max}$ because the special case with $\alpha = 1$ is known to be strongly NP-hard (Garey *et al*. 1976). This negative result indicates that it is very unlikely to be able to devise polynomial or pseudo-polynomial time algorithms.

In spite of the strong NP-hardness of $F2//\Sigma C_i$, in the scheduling literature several researchers still center on the development of exact algorithms that can solve the problem to a certain scale. To design implicit enumeration algorithms for deriving exact optimal schedules, several lower bounds and dominance properties have been proposed in research work such as Ahmadi and Bagchi (1990), Della Croce *et al*. (1996, 2002), Hoogeveen and Kawaguchi (1999), Hoogeveen and van de Velde (1995), Ignall and Schrage (1965), Lin and Wu (2005), and van de Velde (1990). Most of the proposed bounds are based upon Lagrangian relaxation techniques (Fisher 1981, 1985), which have been widely recognized and adopted to successfully cope with hard combinatorial optimization problems. Lin and Wu (2005) proposed a simple lower bound scheme for the total completion time problem. Their computational experiments show that this new scheme could solve optimally most of the test instances with 50 jobs and some instances with 65 jobs. For the bicriteria two-machine flowshop scheduling problem, Nagar *et al*. (1995a) first considered the weighted sum measure. Nagar *et al*. (1995b) proposed a lower bound for the development of branch-and-bound algorithms. Yeh (1999) developed some optimality properties and improved the lower bound by considering the inevitable idle time for the remaining unscheduled jobs. In a computational study, Yeh (1999) claimed the superiority of his algorithm over previous works. Later, Yeh (2001) improved the branch-and-bound algorithm by incorporating new properties and implementation skills. Their approaches, including initial incumbent values, lower bounds, dominance rules and reinforced implementations, can solve instances with up to 20 jobs. In this paper, our goal is to apply our results of the case $\alpha = 1$ to the general $F2//\alpha\Sigma C_i + \beta C_{\max}$ problem. Our technique will not only lead to better results but also provide an easy-to-implement approach to exactly solve the hard problem.

The rest of this paper is organized as follows. In section 2, we shall introduce the notation that will be used throughout this paper. We shall also present some preliminary results. Section 3 is dedicated to the new scheme for establishing lower bounds. Examples will be given for illustration. The computational study and analysis are given in section 4. Section 5 contains some concluding remarks.

2. Notation and previous results

This section first defines the notation that will be used throughout this paper. Then, some previous results from the literature will be introduced. With the exception that the weights α and β are real numbers, all other variables are integers.

Notation

$N = \{1, 2, \ldots, n\}$	job set to be processed
p_i	processing time of machine-1 operation of job i
q_i	processing time of machine-2 operation of job i
$p_{(i)}$	the ith smallest processing times in $\{p_1, p_2, \ldots, p_n\}$
$q_{(i)}$	the ith smallest processing times in $\{q_1, q_2, \ldots, q_n\}$
S	schedule of job set N
α, β	weights associated with total completion time and makespan, $0 \leq \alpha, \beta \leq 1$ and $\alpha + \beta = 1$
C_i^m	completion time of job i on machine m, $m \in \{1, 2\}$, in some schedule
$Z(S)$	objective value of schedule S
$Z^*(N)$	optimal objective value of job set N

To cope with hard combinatorial optimization problems, one may apply several approaches, such as heuristics for deriving the initial incumbent value and dominance rules or lower bounds for curtailing unnecessary branching, to boost the efficiency of the solution algorithms. Because this study is centered around the lower bounds, dominance properties and heuristic approaches are not included. The reader is referred to Yeh (1999, 2001) for details. The first lower bound, called the *I*-bound and denoted LB_I, was proposed by Nagar *et al.* (1995b) for a given partial schedule S_i for the first i jobs. This bound directly calculates the objective value of the partial schedule and the weighted sum of the machine-2 processing times, which are arranged in the shortest processing time (SPT) order:

$$\begin{aligned} LB_I(S_i) &= Z(S_i) + \alpha \sum_{j=i+1}^{n} (n - j + 1) \times q_{(j)} + \beta \sum_{j=i+1}^{n} q_{(j)} \\ &= Z(S_i) + \alpha \sum_{j=i+1}^{n} (n - j) \times q_{(j)} + (\alpha + \beta) \sum_{j=i+1}^{n} q_{(j)} \\ &= Z(S_i) + \sum_{j=i+1}^{n} ((n - j)\alpha + 1) \times q_{(j)}. \end{aligned}$$

In the above formula, $Z(S_i)$ is the cost already incurred by the first i jobs, and $\alpha \sum_{j=i+1}^{n} (n-j+1) \times q_{(j)}$ and $\beta \sum_{j=i+1}^{n} q_{(j)}$ are the lower bounds of the weighted total completion time and the weighted makespan, respectively. Yeh (2001) later improved the I-bound by including the potential idle times that might be incurred for unscheduled jobs. That is, we assume the remaining jobs are to be scheduled by Johnson's algorithm and then add the total idle time in this schedule to the I-bound. The second bound, called the J-bound, is defined as

$$LB_J(S_i) = Z(S_i) + \sum_{j=i+1}^{n} ((n-j)\alpha + 1) \times q_{(j)} + \vartheta_i(S_i),$$

where

$$\vartheta_i(S_i) = \alpha \sum_{j=i+1}^{n} (n-j) \times \max(0, C_j^1 - t_{j-1}^2) + \sum_{j=i+1}^{n} \max\left\{0, C_j^1 - t_{j-1}^2\right\}$$

is the total idle time in the schedule derived by applying Johnson's algorithm to the remaining unscheduled jobs. For implementation details and skills of the J-bound, the reader is referred to Yeh (2001).

3. Our approach

In this section, we introduce Lin and Wu's lower bound scheme for the total completion time problem. The new approach is based upon a data rearrangement mechanism, developed by Cheng *et al.* (2000), that transforms an instance of a strongly NP-hard problem into an ideal form that exhibits polynomial solvability and provides a lower bound for the original hard problem. In Lin and Wu's computational study, under the same settings, branch-and-bound algorithms equipped with this bound can solve most of the $F2//\Sigma C_i$ problems with 55 jobs, whereas the best lower bound (Della Croce *et al.* 2002) known in the literature can solve problems with data set of 25 or 30 jobs.

To underestimate the bicriteria $\alpha\Sigma C_i + \beta C_{\max}$, one may find a schedule whose objective value is smaller than or equal to the optimum value. It is also viable to instead find lower bounds for $C_{\max}$ and ΣC_i, respectively. The weighted sum of the two bounds will also be a lower bound. In our study, the latter approach is employed. Because an optimal solution to $F2//C_{\max}$ is attainable in $O(n \log n)$ time using Johnson's algorithm, we use the optimal makespan as a lower bound for the $C_{\max}$ part of $\alpha\Sigma C_i + \beta C_{\max}$. As for the derivation of the lower bounds of ΣC_i, the data rearrangement methodology newly developed by Lin and Wu (2005) is applied. In the following, we introduce this method and give an example for illustration.

Given job set $N = \{1, 2, \ldots, n\}$, we create a new job set $N' = \{1', 2', \ldots, n'\}$ in which the processing times of job i' are defined as p'_i, the ith smallest element in $\{p_1, p_2, \ldots, p_n\}$, and q'_i, the ith smallest element in $\{q_1, q_2, \ldots, q_n\}$. That is, each job i' of N' is defined by $p_{(i)}$ and $q_{(i)}$. Tables 1 and 2 contain the original data set and the data set derived through the rearrangement process, respectively. Although set N' has two ideal SPT sequences on both machines, it remains NP-hard to find an optimal schedule for it (Hoogeveen and Kawaguchi 1999). Furthermore, it is not guaranteed that an optimal solution value of N' would be smaller than that of the

Table 1. Original data set N.

Job	1	2	3	4	5
p_i	8	20	18	10	8
q_i	5	16	11	20	17

Table 2. Data set N' derived from rearrangement of the processing times.

Job	1	2	3	4	5
p'_i	8	8	10	18	20
q'_i	5	11	16	17	20

original set N. Therefore, a second phase for further refinement is needed to find a lower bound in polynomial time.

With the derived job set N', we exploit the following procedure, called Truncation that schedules the jobs of N' in ascending order of their indices and truncates some machine-2 processing times when necessary.

Procedure Truncation
Input: Derived job set N';
Output: Lower bound for the total completion time of N';
Step 1: $t_1 = t_2 = 0$; $TCT = 0$; /* Initialize the completion times and the total completion time;
Step 2: $flag = 0$; $i = 1$;
Step 3: Do loop
{
$t_1 = t_1 + q'_i$;
If $(t_1 \leq t_2)$ then $t_2 \leq t_2\, q'_i$; $TCT = TCT + t_2$;
Else $flag = 1$; $t_2 = t_1 + \min\{q'_i,\, p'_{i+1}\}$; $TCT = TCT + t_2$;
$i = i + 1$
}
While $(i < n - 1)$ and $(flag = 0)$;
Step 4: For $j = i + 1$ to $n - 1$ do
{
$t_1 = t_1 + p'_j$;
$t_2 = \min\{\max\{t_1, t_2\} + q'_j,\; t_1 + p'_{j+1}\}$;
$TCT = TCT + t_2$;
}
Step 5: $TCT = TCT + t_2$;
Step 6: Return TCT.

The time complexity of this procedure is $O(n \log n)$for sorting the jobs. It could be reduced to $O(n)$ if we deploy a preparatory procedure that arranges the processing times on machine 1 and machine 2 in SPT order before the solution procedure is activated. Lin and Wu have shown that the derived total completion time is no greater than the optimal solution value for the original data set N. The key operation

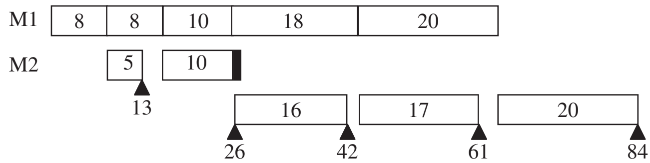

Figure 1. Example of Procedure Truncation.

in PROCEDURE TRUNCATION is locating the first job, in front of which an idle time is incurred on machine 2. Such a non-zero idle time is likely to drive the remaining jobs to have longer completion times and consequently results in a total completion time larger than the optimal one. Therefore, to ensure the existence of a lower bound, truncation is required whenever the machine-2 completion time of a job is strictly later than the machine-1 completion time of its immediate successor in the sequence. Such a mechanism corresponds to the Else part of *Step* 3. When some non-trivial idle time exists to trigger the truncation mechanism, for any remaining job with a machine-2 completion time longer than the machine-1 completion time of its immediate successor, we trim its machine-2 operation such that its machine-2 completion time is the same as the machine-1 completion time of its immediate successor. Statement $t_2 = \min\{\max\{t_1, t_2\} + q'_j, t_1 + p'_{j+1}\}$ of *Step* 4 specifies when and how the truncation operation is performed.

We consider job set N' derived above as an illustration for the procedure. The Gantt chart of the running session is shown in figure 1. Idle time occurs when job $2'$ is scheduled. From this position on, the processing of any machine-2 operation that is completed later than the processing of its successor on machine 1 must be trimmed. As a sequel, the completion time of job $2'$ is 26 instead of 27. Such a trimming operation is conducted when the machine-1 completion time of the newly inserted job is greater than the machine-2 completion time of the current job, i.e. idle time occurs on machine 2 for the newly inserted job. The total completion time reported when the procedure stops is $(13+26+42+61+84)=226$. In this example, only one machine-2 operation is trimmed. Combining the optimal makespan 77 produced by Johnson's algorithm, we have the objective value $0.3\times 226+0.7\times 77=121.7$. The optimal schedule has an objective value of 127.8. Given the same data set, the two previous bounds can be derived as shown in the following:

$$\begin{aligned} LB_I &= \sum_{j=i+1}^{n} [(n-j)\alpha + 1] \times q_{[j]} \\ &= (0.3 \times 4 + 1) \times 5 + (0.3 \times 3 + 1) \times 11 + (0.3 \times 2 + 1) \times 16 \\ &\quad + (0.3 \times 1 + 1) \times 17 + (0.3 \times 0 + 1) \times 2 \\ &= 99.6, \end{aligned}$$

and

$$\begin{aligned} LB_J &= LB_I + \vartheta(S_i) \\ &= 99.6 + (0.3 \times 4 + 1) \times 8 \\ &= 117.2. \end{aligned}$$

In the derivation of $\vartheta(S_i)$, an idle time of 8 units is incurred for the first job in Johnson's sequence for the unscheduled jobs. Therefore, an increment of 17.6 is incorporated. Considering the three lower bounds for the data set given in table 1, we readily realize that the bound derived by Johnson's algorithm and the data rearrangement scheme is tighter than the other two.

In addition to the capability of composing optimal schedules for middle-scale problems, the lower bound based upon data rearrangement also demonstrates several advantages. First of all, the fundamentals of the scheme are free from mathematical skills such as Lagrangian relaxation. Second, the implementation is straightforward and simple. Furthermore, verification of the correctness of the scheme can be conducted through combinatorial arguments instead of complicated mathematical derivations.

Although Lin and Wu's computational study has indicated that the data rearrangement scheme is effective in reducing the effort demanded for probing the solution space of the total completion time problem, whether or not it works well for the bicriteria problem is still unknown. In the next section, we shall design and conduct computational experiments to study the joint effectiveness of Johnson's algorithm and the data rearrangement scheme for the $F2\|\alpha\Sigma C_i + \beta C_{\max}$ problem.

4. Computational experiments

Because there is no theoretical analysis concerning the relative performances of the existing bounds and the new scheme, we circumvent this problem and use computational experiments to investigate the efficiency issue. We wrote the codes in *C* language and performed the experiments under the Linux Red Hat 7.0 operating system running on a personal computer with a Pentium III 1.6 GHz CPU, 256 MB RAM and a 40 GB hard disk. In the experiments, two branch-and-bound algorithms were implemented. The first algorithm is based upon the data rearrangement approach. The second algorithm first uses the *I*-bound for each partial schedule. If a partial solution is not pruned by the *I*-bound, the second part of the *J*-bound will be calculated to derive the value of the *J*-bound. The two algorithms are denoted $LB_{TR\text{-}J}$ and $LB_{I\text{-}J}$. The solution trees of both algorithms are explored in a depth-first fashion. Such a choice was made to keep the implementations simple in order to avoid excess memory requirement and programming skills that might be needed by strategies such as breadth-first search or best-first search. Such simplicity also avoids potential bias that might result from the implementation details, such as sophisticated data structures. Also note that we did not use any heuristic to derive initial upper bounds or incumbent values. Initially, the incumbent is a large number that would gradually decrease as better solutions are encountered.

The job instances were randomly generated in three different modes: (1) $p_i \in [0,100], q_i \in [0,100]$; (2) $p_i \in [0,100], q_i \in [0,50]$; and (3) $p_i \in [0,50], q_i \in [0,100]$. Each interval is discrete and uniformly distributed. The arrangement will depict different situations concerning the relative length of processing for each individual job on different machines, and will provide more extensive observations on the behavior of the implemented algorithms. For each different problem size (n), 10 job sets were generated and run by the branch-and-bound algorithms equipped with different bounds. For each job set, a limit of 30 min was given to confine the execution of the algorithms. That is, if an algorithm cannot optimally solve a job set in 30 min, it will abort and report a failure. The statistics of the experiments shown were averaged over successful running sessions of all 10 instances. Throughout the experiments, we recorded (1) *#_opt*: the number of instances successfully solved; (2) *avg_time*: the average run time for the successfully solved instances; (3) *max_time*: the longest time elapsed of the successfully solved instances; (4) *avg_node*: the average number of nodes visited for the successfully solved instances; and (5) *max_node*: the largest number of nodes visited of the successfully solved instances.

Table 3 contains the results for two algorithms solving instances with 10 jobs. The statistics clearly demonstrate the superiority of our new approach over the existing one. The elapsed computation time and the number of visited nodes of our algorithm are almost negligible in comparison with those of the algorithm with LB_{I-J}. The ratio is around 1000. Our test continued with an increment of five jobs. Because the algorithm with LB_{I-J} could not solve any instance with 15 or more jobs, the results are not shown. Table 4 lists the results of our algorithm for solving instances of 15, 20, 30 and 35 jobs. For 15 job problems, the number of candidate sequences is 15!, which is on the order of 10^{12}. On average, our algorithm visited only about 10^5 or 10^6 nodes, which is 10^7 or 10^6 times less than the size of the solution space. A very important observation to address is the relationship between the performance of our algorithm and the modes by which the data were generated. The most difficult cases were encountered when the processing times on both machines were taken from the same interval [0,100]. The algorithm performed well for the other two data modes, $p_i \in [0,100], q_i \in [0,50]$ and $p_i \in [0,50], q_i \in [0,100]$. This is reflected in all terms including time, nodes and number of instances solved. Figure 2 compares the total number of instances solved for different problem sizes and data modes. Further examination suggests that our algorithm demonstrates the best problem-solving capability when the data size is relatively large and generated using the mode $p_i \in [0,50], q_i \in [0,100]$. This might be due to the fact that when p_i is relatively smaller than q_i, the potential idle time on machine 2 could be reduced to a certain degree and the lower bounds were much closer to the optimal solution values. We further scrutinized the dimensions of the weights α and β. When the value of α is large, the objective value is anticipated to be more dependent on the total completion time. This might then imply that the role the lower bound of the total completion time plays will become more crucial. However, the statistics do not reflect this.

As a general summary, our approach demonstrates its capability in dealing with middle-scale instances. Most of the test cases with 30 or less jobs were solved successfully. Compared with the existing method, our algorithm represents a significant improvement. Furthermore, the simplicity of implementation makes its practical use much more viable.

5. Conclusions

In this paper, we have considered a two-machine flowshop scheduling problem to minimize the weighted sum of the makespan and the total completion time. To optimally solve the problem, a branch-and-bound algorithm equipped with two lower bounds has been addressed. The lower bound of the total completion time

Table 3. Numerical results for bounds $LB_{TR\text{–}J}$ and $LB_{I\text{–}J}$.

		$LB_{TR\text{-}J}$					$LB_{I\text{-}J}$				
Data mode	α	#_opt	Avg_time	Max_time	Avg_node	Max_node	#_opt	Avg_time	Max_time	Avg_node	Max_node
$p_i \in [0, 100]$, $q_i \in [0, 100]$	0.1	10	0.011	0.02	4342.2	9694	10	5.896	9.57	3 630 226	5 797 565
	0.2	10	0.008	0.01	3585.4	5830	10	8.103	13.08	5 066 051	8 243 532
	0.3	10	0.011	0.03	4981.5	12 860	10	10.265	14.38	6 313 707	8 897 275
	0.4	10	0.007	0.01	2616.3	6327	10	8.214	11.88	5 106 893	7 300 271
	0.5	10	0.007	0.01	3104.6	4644	10	8.642	11.38	5 360 178	7 174 942
	0.6	10	0.007	0.01	3635.9	6495	10	9.870	13.67	6 113 064	8 423 607
	0.7	10	0.009	0.02	3716.0	10 996	10	9.683	11.43	6 033 337	7 186 903
	0.8	10	0.006	0.02	2736.1	5855	10	8.747	11.02	5 472 960	6 918 081
	0.9	10	0.010	0.04	3969.0	16 826	10	10.132	13.94	6 309 212	8 657 193
$p_i \in [0, 100]$, $q_i \in [0, 50]$	0.1	10	0.010	0.03	5685.3	15 041	10	7.407	10.89	4 459 487	6 649 113
	0.2	10	0.011	0.02	5685.6	13 599	10	9.532	13.51	5 834 734	8 413 749
	0.3	10	0.007	0.04	3965.0	19 666	10	8.353	13.39	5 130 353	8 331 656
	0.4	10	0.006	0.01	2398.2	6523	10	9.305	12.79	5 716 079	7 928 900
	0.5	10	0.008	0.02	2772.9	9601	10	10.329	14.15	6 415 446	8 886 679
	0.6	10	0.006	0.01	1589.3	2938	10	8.550	11.04	5 272 271	6 833 493
	0.7	10	0.005	0.01	1940.7	3243	10	8.768	11.85	5 413 319	7 370 102
	0.8	10	0.003	0.01	1302.0	2593	10	9.862	13.92	6 107 623	8 703 403
	0.9	10	0.003	0.01	1411.2	3291	10	10.834	12.81	6 734 367	8 025 638
$p_i \in [0, 50]$, $q_i \in [0, 100]$	0.1	10	0.004	0.01	2091.0	6014	10	6.843	9.91	4 301 645	6 234 171
	0.2	10	0.003	0.01	1581.9	4346	10	7.290	9.09	4 578 888	5 716 712
	0.3	10	0.005	0.01	2141.9	4931	10	9.197	13.54	5 783 879	8 517 798
	0.4	10	0.007	0.04	3092.4	19 125	10	9.436	12.28	5 927 541	7 725 276
	0.5	10	0.008	0.04	3421.2	17 412	10	7.979	9.72	5 008 025	6 105 567
	0.6	10	0.004	0.02	1770.5	7467	10	8.396	11.01	5 277 513	6 905 758
	0.7	10	0.003	0.01	1479.8	3138	10	9.131	13.75	5 746 402	8 670 108
	0.8	10	0.004	0.02	1773.8	5635	10	8.826	11.42	5 549 160	7 194 248
	0.9	10	0.002	0.01	1822.9	3287	10	8.905	11.13	5 602 277	6 997 718

Table 4(a). Numerical results for bound $LB_{TR\text{–}J}$.

		$n=15$					$n=20$				
Data mode	α	#_opt	Avg_time	Max_time	Avg_node	Max_node	#_opt	Avg_time	Max_time	Avg_node	Max_node
$p_i \in [0, 100]$, $q_i \in [0, 100]$	0.1	10	3.419	20.36	1.45E+06	8.59E+06	6	91.613	239.90	3.48E+07	9.08E+07
	0.2	10	0.717	3.85	3.09E+05	1.66E+06	6	89.083	240.11	3.43E+07	9.49E+07
	0.3	10	0.512	3.06	2.17E+05	1.28E+06	8	255.731	719.18	9.71E+07	2.69E+08
	0.4	10	0.874	3.34	3.77E+05	1.42E+06	10	97.489	356.73	3.66E+07	1.33E+08
	0.5	10	0.831	3.88	3.55E+05	1.64E+06	9	164.307	859.06	6.23E+07	3.22E+08
	0.6	10	1.969	14.25	8.46E+05	6.12E+06	9	269.424	669.52	1.02E+08	2.56E+08
	0.7	10	2.122	15.70	9.09E+05	6.67E+06	10	319.935	775.67	1.21E+08	2.95E+08
	0.8	10	2.078	17.76	8.88E+05	7.57E+06	9	171.083	912.00	6.48E+07	3.47E+08
	0.9	10	0.410	1.00	1.74E+05	4.20E+05	10	105.745	274.95	3.96E+07	1.03E+08
$p_i \in [0, 100]$, $q_i \in [0, 50]$	0.1	10	0.711	2.79	3.30E+05	1.32E+06	10	99.247	688.87	4.17E+07	2.91E+08
	0.2	10	2.039	6.32	9.16E+05	2.85E+06	10	49.186	339.70	1.97E+07	1.33E+08
	0.3	10	0.173	0.87	8.51E+04	4.31E+05	10	11.816	46.23	4.94E+06	2.02E+07
	0.4	10	0.145	0.58	6.72E+04	2.67E+05	10	13.339	93.07	5.22E+06	3.55E+07
	0.5	10	0.096	0.53	4.38E+04	2.27E+05	10	3.720	28.44	1.63E+06	1.26E+07
	0.6	10	0.120	0.57	5.48E+04	2.46E+05	10	10.746	48.02	4.07E+06	1.81E+07
	0.7	10	0.049	0.14	2.36E+04	6.40E+04	10	2.320	12.14	8.98E+05	4.41E+06
	0.8	10	0.172	0.91	7.56E+04	3.82E+05	10	1.864	6.54	7.12E+05	2.47E+06
	0.9	10	0.114	0.76	4.99E+04	3.14E+05	10	5.401	48.87	2.03E+06	1.83E+07
$p_i \in [0, 50]$, $q_i \in [0, 100]$	0.1	10	0.929	5.27	3.92E+05	2.22E+06	8	10.109	74.06	3.77E+06	2.76E+07
	0.2	10	0.860	6.91	3.69E+05	2.96E+06	10	1.269	9.69	4.76E+05	3.61E+06
	0.3	10	5.231	35.38	2.28E+06	1.53E+07	10	12.253	111.14	4.60E+06	4.18E+07
	0.4	10	0.182	1.00	7.85E+04	4.22E+05	10	0.964	6.57	3.59E+05	2.40E+06
	0.5	10	0.045	0.21	2.13E+04	8.99E+04	10	0.295	1.47	1.13E+05	5.52E+05
	0.6	10	2.279	16.36	9.99E+05	7.29E+06	10	1.864	14.35	6.99E+05	5.38E+06
	0.7	10	3.514	18.13	1.52E+06	7.69E+06	8	0.994	2.94	3.68E+05	1.07E+06
	0.8	10	0.084	0.59	3.77E+04	2.52E+05	10	1.515	10.14	5.60E+05	3.73E+06
	0.9	10	35.190	351.47	1.58E+07	1.58E+08	10	0.266	1.12	1.02E+05	4.14E+05

Table 4(b). Numerical results for bound $LB_{TR\text{–}J}$.

		n = 25					n = 40				
Data mode	α	#_opt	Avg_time	Max_time	Avg_node	Max_node	#_opt	Avg_time	Max_time	Avg_node	Max_node
$p_i \in [0, 100]$, $q_i \in [0, 100]$	0.1	3	403.883	1167.90	1.35E+08	3.91E+08	0	–	–	–	–
	0.2	2	189.045	213.48	6.41E+07	7.26E+07	0	–	–	–	–
	0.3	2	363.750	501.65	1.21E+08	1.67E+08	1	598.5	598.5	1.75E+08	1.75E+08
	0.4	1	634.040	634.04	2.11E+08	2.11E+08	1	17.55	17.55	4.95E+06	4.95E+06
	0.5	3	230.687	518.97	7.70E+07	1.73E+08	0	–	–	–	–
	0.6	4	80.073	164.34	2.62E+07	5.33E+07	0	–	–	–	–
	0.7	2	4.585	8.41	1.58E+06	2.89E+06	1	1.37	1.37	4.70E+05	4.70E+05
	0.8	6	635.328	1456.56	2.15E+08	4.85E+08	1	111.22	111.22	3.20E+07	3.20E+07
	0.9	3	599.283	1267.27	2.06E+08	4.27E+08	0	–	–	–	–
$p_i \in [0, 100]$, $q_i \in [0, 50]$	0.1	7	122.869	435.65	4.64E+07	1.59E+08	4	529.228	1323.78	1.63E+08	3.89E+08
	0.2	8	53.781	257.71	2.07E+07	1.01E+08	4	306.526	406.35	1.01E+08	1.39E+08
	0.3	8	508.314	1626.28	1.95E+08	6.33E+08	7	107.053	208.24	3.60E+07	7.26E+07
	0.4	9	177.861	623.56	6.68E+07	2.51E+08	4	59.590	125.44	2.03E+07	4.45E+07
	0.5	10	128.361	535.38	4.86E+07	2.21E+08	6	186.197	636.07	5.75E+07	1.88E+08
	0.6	9	28.629	77.59	9.83E+06	2.59E+07	7	209.546	509.54	7.68E+07	1.90E+08
	0.7	9	255.621	1723.42	8.47E+07	5.67E+08	7	81.684	200.45	2.66E+07	6.21E+07
	0.8	10	330.733	1349.27	1.13E+08	4.51E+08	10	86.389	378.44	2.59E+07	1.11E+08
	0.9	9	37.921	152.62	1.26E+07	5.08E+07	9	72.968	236.88	2.19E+07	6.92E+07
$p_i \in [0, 50]$, $q_i \in [0, 100]$	0.1	9	8.954	58.42	2.91E+06	1.88E+07	9	65.390	367.99	1.86E+07	1.05E+08
	0.2	9	32.761	99.00	1.08E+07	3.26E+07	8	14.948	95.19	4.29E+06	2.69E+07
	0.3	9	2.108	8.70	6.98E+05	2.82E+06	4	64.585	157.03	1.86E+07	4.51E+07
	0.4	10	7.742	56.19	2.57E+06	1.85E+07	8	190.838	1476.57	5.49E+07	4.24E+08
	0.5	8	90.881	717.77	3.02E+07	2.39E+08	6	215.667	1259.46	6.20E+07	3.62E+08
	0.6	9	11.711	67.65	3.84E+06	2.21E+07	8	424.113	1211.43	1.23E+08	3.49E+08
	0.7	8	12.675	35.99	4.11E+06	1.16E+07	8	352.265	1130.42	1.02E+08	3.26E+08
	0.8	6	25.378	144.36	8.35E+06	4.74E+07	9	234.053	850.15	6.73E+07	2.46E+08
	0.9	8	71.446	283.35	2.33E+07	9.22E+07	8	249.576	1447.67	7.23E+07	4.19E+08

Table 4(c). Numerical results for bound LB_{TR-J}.

		$n=35$					$n=30$				
Data mode	α	#_opt	Avg_time	Max_time	Avg_node	Max_node	#_opt	Avg_time	Max_time	Avg_node	Max_node
$p_i \in [0, 100]$, $q_i \in [0, 100]$	0.1	0	–	–	–	–	0	–	–	–	–
	0.2	0	–	–	–	–	0	–	–	–	–
	0.3	0	–	–	–	–	0	–	–	–	–
	0.4	0	–	–	–	–	0	–	–	–	–
	0.5	0	–	–	–	–	0	–	–	–	–
	0.6	0	–	–	–	–	0	–	–	–	–
	0.7	0	–	–	–	–	0	–	–	–	–
	0.8	0	–	–	–	–	0	–	–	–	–
	0.9	0	–	–	–	–	0	–	–	–	–
$p_i \in [0, 100]$, $q_i \in [0, 50]$	0.1	0	–	–	–	–	0	–	–	–	–
	0.2	2	481.705	716.85	1.6E+08	2.42E+08	0	–	–	–	–
	0.3	0	–	–	–	–	4	513.455	1468.48	1.42E+08	4.11E+08
	0.4	2	104.530	195.99	2.78E+07	5.16E+07	1	21.590	21.59	5.93E+06	5.93E+06
	0.5	3	766.973	1776.81	2.80E+08	6.77E+08	1	304.770	304.77	7.49E+07	7.49E+07
	0.6	3	554.173	1294.92	1.88E+08	4.38E+08	2	202.220	230.31	4.94E+07	5.63E+07
	0.7	7	461.350	1440.32	1.46E+08	4.94E+08	2	975.170	1324.85	2.58E+08	3.16E+08
	0.8	5	379.582	158.84	1.04E+08	3.33E+08	4	365.250	653.82	1.16E+08	1.96E+08
	0.9	5	234.832	747.52	6.51E+07	1.94E+08	1	42.020	42.02	1.00E+07	1.00E+07
$p_i \in [0, 50]$, $q_i \in [0, 100]$	0.1	4	18.550	57.25	4.98E+06	1.48E+07	6	681.730	1564.29	1.63E+08	3.70E+08
	0.2	5	97.782	251.36	2.54E+07	6.57E+07	6	191.063	996.82	4.61E+07	2.40E+08
	0.3	3	91.000	263.79	2.35E+07	6.79E+07	6	252.192	1201.54	6.13E+07	2.91E+08
	0.4	3	134.037	298.63	3.53E+07	7.83E+07	3	141.737	411.55	3.38E+07	9.75E+07
	0.5	7	28.894	176.34	7.76E+06	4.65E+07	5	112.400	489.16	2.66E+07	1.15E+08
	0.6	6	251.925	976.01	6.49E+07	2.51E+08	5	118.022	357.19	2.82E+07	8.45E+07
	0.7	5	23.734	52.24	6.28E+06	1.37E+07	5	101.100	372.93	2.45E+07	8.95E+07
	0.8	7	271.600	976.20	6.99E+07	2.49E+08	3	114.773	310.02	2.77E+07	7.45E+07
	0.9	6	458.083	1354.96	1.22E+08	3.63E+08	2	707.140	1405.87	1.68E+08	3.35E+08

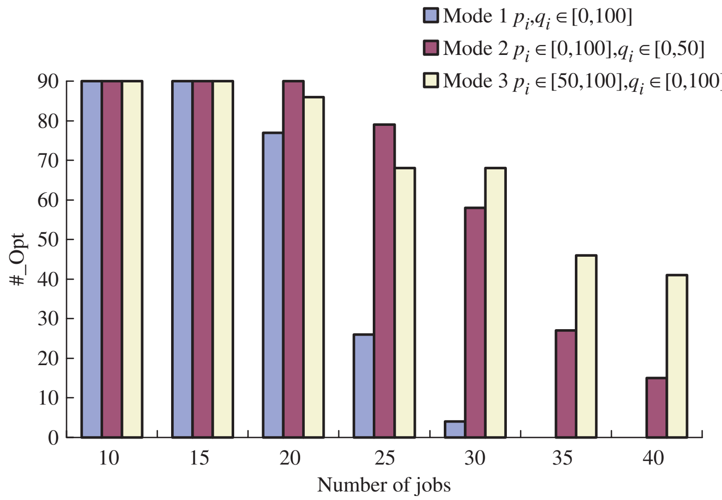

Figure 2. Numbers of instances solved by different data modes.

is obtained by applying a data rearrangement scheme that was previously developed for the scheduling problem with a single objective function. The statistics obtained from computational experiments suggest that a strategy comprising Johnson's algorithm and the data rearrangement scheme is a powerful tool for determining unnecessary branches in the solution tree. The success shown in this study also supports the significance of the data rearrangement scheme for flowshop-related problems. Adapting this approach to other flowshop problems or even other optimization problems could be a worthy direction for future research.

Acknowledgements

The authors are supported, in part, by the NSC of ROC and the NRC of Canada under grant number NSC 91-2213-E-002-111 and project grant NSC-91-2416-H-260–001.

References

Ahmadi, R.H. and Bagchi, U., Improved lower bounds for minimizing the sum of completion times of n jobs over m machines in a flow shop. *European Journal of Operational Research*, 1990, **44**, 331–336.

Cheng, T.C.E., Lin, B.M.T. and Toker, A., Flowshop batching and scheduling to minimize the makespan. *Naval Research Logistics*, 2000, **47**, 128–144.

Della Croce, F., Ghirardi, M. and Tadei, R., An improved branch-and-bound algorithm for the two machine total completion time flow shop problem. *European Journal of Operational Research*, 2002, **139**, 293–301.

Della Croce, F., Narayan, V. and Tadei, R., The two-machine total completion time flow shop problem. *European Journal of Operational Research*, 1996, **90**, 227–237.

Dudek, R.A., Panwalkar, S.S. and Smith, M.L., The lessons of flowshop scheduling research. *Operations Research*, 1992, **40**, 7–13.

Fisher, M.L., The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 1981, **27**, 1–18.

Fisher, M.L., An applications-oriented guide to Lagrangian relaxation. *Interfaces*, 1985, **15**, 10–21.

Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness* (San Francisco, CA: Freedman, 1979).

Garey, M.R., Johnson, D.S. and Sethi, R.R., The complexity of flowshop and jobshop scheduling. *Operations Research*, 1976, **1**, 117–129.

Graham, R.L., Lawler, E.L., Lenstra, J.K. and Rinnoy Kan, A.H.G., Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 1979, **5**, 287–326.

Hoogeveen, H. and Kawaguchi, T., Minimizing total completion time in a two-machine flowshop: analysis of special cases. *Mathematics of Operations Research*, 1999, **24**, 887–913.

Hoogeveen, J.A. and van de Velde, S.L., Stronger Lagrangian bounds by use of slack variables: application to machine scheduling problems. *Mathematical Programming*, 1995, **70**, 173–190.

Ignall, E. and Schrage, L.E., Application of the branch-and-bound technique to some flowshop scheduling problems. *Operations Research*, 1965, **13**, 400–412.

Johnson, S.M., Optimal two and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1954, **1**, 61–68.

Lin, B.M.T. and Wu, J.M., A simple lower bound for total completion time minimization in a two-machine flowshop. *Asia–Pacific Journal of Operational Research*, 2005, **22**, 391–408.

Nagar, A., Haddock, J. and Heragu, S.S., Multiple and bicriteria scheduling: a literature review. *European Journal of Operational Research*, 1995a, **81**, 88–104.

Nagar, A., Sunderesh, S.H. and Haddock, J., A branch-and-bound approach for a two-machine flowshop scheduling problem. *Journal of the Operational Research Society*, 1995b, **46**, 721–734.

Reisman, A., Kumar, A. and Motwani, J., Flowshop scheduling/sequencing research: a statistical review of the literature, 1952–1994. *IEEE Transactions on Engineering Management*, 1997, **44**, 316–329.

van de Velde, S.L., Minimizing the sum of job completion times in the two-machine flow-shop by Lagrangean relaxation. *Annals of Operations Research*, 1990, **26**, 257–268.

Yeh, W.C., A new branch-and-bound approach for the $n/2/\text{flowshop}/F+C_{\max}$ flowshop scheduling problem. *Computers and Operations Research*, 1999, **26**, 1293–1310.

Yeh, W.C., An efficient branch-and-bound algorithm for the two-machine bicriteria flowshop scheduling problem. *Journal of Manufacturing Systems*, 2001, **20**, 113–123.