The Permutation Flow Shop Problem with Sum-of-Completion Times Performance Criterion

Selcuk Karabati
Management Department, Faculty of Business Administration, Bilkent University, 06533 Bilkent, Ankara, Turkey

Panagiotis Kouvelis
The Fuqua School of Business, Duke University, Durham, North Carolina, 27706

In this article we address the non-preemptive flow shop scheduling problem for minimization of the sum of the completion times. We present a new modeling framework and give a novel game-theoretic interpretation of the scheduling problem. A lower-bound generation scheme is developed by solving appropriately defined linear assignment problems. This scheme can also be used as a heuristic approach for the solution of the problem with satisfactory results. Its main use, however, is as a bounding scheme within a branch-and-bound procedure. Our branch-and-bound procedure improves significantly upon the best available enumerative procedures in the current literature. Extensive computational results are used to qualify the above statements. © 1993 John Wiley & Sons, Inc.

1. INTRODUCTION

We address the non-preemptive permutation flow shop scheduling problem with the sum of completion times as the performance criterion. The above problem can be stated as follows. Each of the n jobs, $J_1, J_2, \ldots, J_n$, has to be processed on m machines, $M_1, M_2, \ldots, M_m$, in that order. The processing of job J_i on machine M_j requires an uninterrupted period of processing time $p_{i,j}$. Each machine can process at most one job at a time. The objective is to find a permutation schedule, i.e., a single ordering of the jobs on all machines, such that the sum of completion times is minimized. Minimizing the sum of completion times amounts to minimizing the mean completion time of jobs as well. A permutation schedule is represented by $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$, where $\sigma(i)$ is the ith job in the processing order. In our further discussion when we refer to a schedule σ we mean a permutation schedule, unless specified otherwise.

Because the problem is *NP*-hard for $m \geq 2$ (see Gonzalez and Sahni [6]), enumerative and/or heuristic approaches are essentially unavoidable. For $m = 2$, solution procedures have been developed by Ignall and Schrage [7] and Kohler and Steiglitz [8]. A lower bounding scheme for $m = 2$ was proposed by Ignall

Naval Research Logistics, Vol. 40, pp. 843–862 (1993)
 CCC 0894-069X/93/060843-20

and Schrage [7]. Their procedure was generalized to arbitrary m by Bansal [3]. Ahmadi and Bagchi [1] improved upon Bansal's bounding scheme. Miyazaki, Nishiyama, and Hashimoto [11], and Szwarc [15] have developed sufficient optimality conditions for the problem using a pairwise interchange method. Miyazaki et al. [11] presented also a heuristic solution procedure.

In this article we expand on the modeling framework of Monma and Rinnooy Kan [12] for permutation flow shops. We model the problem as a two-person zero-sum game. Using this new modeling approach, we present a lower-bounding scheme that requires the solution of a series of linear assignment problems. Using this bounding scheme and a new branching approach, we develop an efficient branch-and-bound procedure for the problem. The main ideas of our bounding scheme can also be used to generate a good heuristic solution to the problem.

The structure of the article is as follows. In Section 2 we present our modeling framework and discuss a class of linear assignment problems which will play an important role in the rest of the article. A novel interpretation of the flow shop scheduling problem in a game theoretic context is presented in Section 3. In Section 4 we discuss a new lower-bound generation approach based on the results of Sections 2 and 3. Capitalizing on the results of Section 4, we develop a new branch-and-bound scheme for permutation flow shops in Section 5. We report on our computational experience with this new approach in Section 6. Finally, Section 7 summarizes our results and discusses some potential extensions of our modeling approach.

2. CRITICAL PATHS, ASSIGNMENT PROBLEMS, AND THE PERMUTATION FLOW SHOP SCHEDULING PROBLEM

For a schedule σ we can define a recursive relationship to find the completion time $C_{\sigma(i),j}$ of job $J_{\sigma(i)}$ on machine M_j as follows (for detailed explanations refer to Baker [2] and Monma and Rinnooy Kan [12]):

$$C_{\sigma(i),j} = \max\{C_{\sigma(i),j-1}, C_{\sigma(i-1),j}\} + p_{\sigma(i),j}, \qquad 1 \le i \le n, \qquad 1 \le j \le m, \quad (1)$$

where $C_{\sigma(0),j} = C_{\sigma(i),0} = 0$. The recursive structure (1) can be equivalently presented, as pointed out by Monma and Rinnooy Kan [12], by the directed graph G depicted in Figure 1. Vertices $(\sigma(i), j)$ are defined for each element $\sigma(i)$ of the permutation and each machine M_j. A weight $p_{\sigma(i),j}$ is associated with vertex $(\sigma(i), j)$. Directed arcs are defined from each vertex $(\sigma(i), j)$ toward $(\sigma(i + 1), j)$ and $(\sigma(i), j + 1)$.

Given the above-defined graph G, the completion time $C_{\sigma(i),j}$ of job $J_{\sigma(i)}$, on machine M_j, in the permutation schedule σ, is equal to the maximum-weighted directed path from $(\sigma(1), 1)$ to $(\sigma(i), j)$ in the graph. Therefore, the sum of completion times of jobs in schedule σ, $F(\sigma)$, is given by

$$F(\sigma) = \sum_{i=1}^{n} C_{\sigma(i),m}.$$

Any path from $(\sigma(1), 1)$ to $(\sigma(i), m)$ which attains the value $C_{\sigma(i),m}$ is called a critical path for job $J_{\sigma(i)}$ in schedule σ. Consequently, the problem of finding

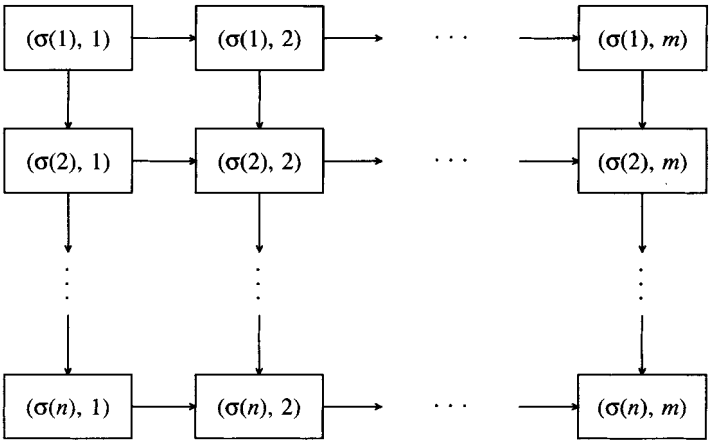

Figure 1. Directed graph (G) representation of a permutation schedule σ.

the sum of completion times of a schedule is analogous to the determination of n maximum paths, one for each job, on an acyclic graph with positive arc lengths. The computational complexity of determining all such paths is $O(nm)$ (Lawler [10]).

Let $\tau(i)$ be a path from $(\sigma(1), 1)$ to $(\sigma(i), m)$ on the directed graph G. Szwarc [15] makes the observation that the length $f(\sigma(i), \tau(i))$ of the path $\tau(i)$ can be presented as

$$f(\sigma(i), \tau(i)) = \sum_{k=1}^{t(i,1)} p_{\sigma(1),k} + \sum_{k=t(i,1)}^{t(i,2)} p_{\sigma(2),k} + \cdots + \sum_{k=t(i,i-1)}^{m} p_{\sigma(i),k}, \tag{2}$$

where $1 \le t(i, 1) \le t(i, 2) \le \cdots \le t(i, i-1) \le m$ are appropriate integers which uniquely define $\tau(i)$. Let $T_{i,m}$ denote the set of all such paths on the direct graph G for job $J_{\sigma(i)}$. Then $F(\sigma)$, the sum of completion times, can be rewritten as

$$F(\sigma) = \sum_{i=1}^{n} \left(\max_{\tau(i) \in T_{i,m}} f(\sigma(i), \tau(i)) \right). \tag{3}$$

By letting $T = T_{1,m} \times T_{2,m} \times \cdots \times T_{n,m}$, and $\tau = (\tau(1), \tau(2), \ldots, \tau(n)) \in T$, we can rewrite (3) as

$$F(\sigma) = \max_{\tau \in T} \sum_{i=1}^{n} f(\sigma(i), \tau(i)). \tag{4}$$

Then, if S_n is the set of all permutation schedules, the permutation flow shop scheduling problem for minimization of the sum of completion times can be formulated as follows:

$$\min_{\sigma \in S_n} F(\sigma) = \min_{\sigma \in S_n} \max_{\tau \in T} \sum_{i=1}^{n} f(\sigma(i), \tau(i)). \tag{5}$$

Using the above established modeling framework, we can now state the following proposition. For presentation convenience, we use the usual abbreviation $n/m/P/\Sigma C_i$ for the n-job m-machine permutation flow shop scheduling problem with the sum-of-completion-times performance criterion.

PROPOSITION 1: *For the $n/m/P/\Sigma C_i$ problem the following relationship holds:*

$$\max_{\tau\in T}\min_{\sigma\in S_n}\sum_{i=1}^{n} f(\sigma(i), \tau(i)) \le \min_{\sigma\in S_n}\max_{\tau\in T}\sum_{i=1}^{n} f(\sigma(i), \tau(i)). \tag{6}$$

PROOF: Let $\sigma_0 \in S_n$ be a specific permutation schedule and $\tau_0 \in T$ be a path vector on the directed graph G. Then

$$\min_{\sigma\in S_n}\sum_{i=1}^{n} f(\sigma(i), \tau_0(i)) \le \sum_{i=1}^{n} f(\sigma_0(i), \tau_0(i)) \le \max_{\tau\in T}\sum_{i=1}^{n} f(\sigma_0(i), \tau(i)). \tag{7}$$

However, (7) holds for every $\sigma_0 \in S_n$ and $\tau_0 \in T$, therefore the validity of (6) follows immediately. □

Let $F(\sigma, \tau) = \Sigma_{i=1}^{n} f(\sigma(i), \tau(i))$, $\sigma \in S_n$ and $\tau \in T$. Then Proposition 1 states that for any path vector $\tau \in T$, the solution to the optimization problem

$$(AP): \min_{\sigma\in S_n} F(\sigma, \tau)$$

provides a lower bound for the problem as formulated in (5). Some further insight on the nature of (AP) is provided by the following result.

PROPOSITION 2: *(AP) is equivalent to a linear assignment problem.*

PROOF: According to (2), for a specific path $\tau(i)$ in path vector $\tau \in T$, there exists a unique set of integers $(t(i, 1), t(i, 2), \ldots, t(i, i-1))$ such that for any $\sigma \in S_n$, we have the following:

$$f(\sigma(i), \tau(i)) = \sum_{r=1}^{n}\sum_{j=1}^{i}\sum_{k=t(i,j-1)}^{t(i,j)} p_{r,k}1\{\sigma(j) = J_r\}, \tag{8}$$

where $1\{\cdot\}$ is an indicator function and the integers $t(i, 0)$ and $t(i, i)$ assume the values 1 and m, respectively. Note that we have $1 \le t(i, 1) \le t(i, 2) \le \cdots \le t(i, i-1) \le m$. Now by letting $t(i, k) = 0$, $i < k \le n$ we can rewrite (8) as

$$f(\sigma(i), \tau(i)) = \sum_{r=1}^{n}\sum_{j=1}^{n}\sum_{k=t(i,j-1)}^{t(i,j)} p_{r,k}1\{\sigma(j) = J_r\}.$$

Then $F(\sigma, \tau)$ becomes

$$F(\sigma, \tau) = \sum_{i=1}^{n} f(\sigma(i), \tau(i)) = \sum_{i=1}^{n} \sum_{r=1}^{n} \sum_{j=1}^{n} \sum_{k=t(i,j-1)}^{t(i,j)} p_{r,k} 1\{\sigma(j) = J_r\}.$$

By reordering of the above summations we obtain

$$F(\sigma, \tau) = \sum_{i=1}^{n} f(\sigma(i), \tau(i)) = \sum_{r=1}^{n} \sum_{j=1}^{n} \left(\sum_{i=1}^{n} \sum_{k=t(i,j-1)}^{t(i,j)} p_{r,k} \right) 1\{\sigma(j) = J_r\}. \quad (9)$$

Let us denote by

$$a_{r,j}^{\tau} = \sum_{i=1}^{n} \sum_{k=t(i,j-1)}^{t(i,j)} p_{r,k}.$$

Then (9) can be rewritten as

$$F(\sigma, \tau) = \sum_{i=1}^{n} f(\sigma(i), \tau(i)) = \sum_{r=1}^{n} \sum_{j=1}^{n} a_{r,j}^{\tau} 1\{\sigma(j) = J_r\}. \quad (10)$$

Using relationship (10) one can easily observe that (AP) is equivalent to a linear assignment problem with cost matrix $A^{\tau} = (a_{r,j}^{\tau})$. □

EXAMPLE: We now present a numerical illustration of Proposition 2. Consider a three-job, two-machine problem with the paths given in Figure 2. For these paths we have the following path weights:

$$f(\sigma(1), \tau(1)) = p_{\sigma(1),1} + p_{\sigma(1),2},$$

$$f(\sigma(2), \tau(2)) = p_{\sigma(1),1} + p_{\sigma(1),2} + p_{\sigma(2),2},$$

$$f(\sigma(3), \tau(3)) = p_{\sigma(1),1} + p_{\sigma(2),1} + p_{\sigma(3),1} + p_{\sigma(3),2}.$$

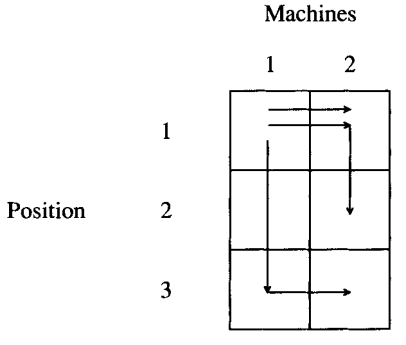

Figure 2. The paths for the example problem.

We also have

$$F(\sigma, \tau) = 3p_{\sigma(1),1} + 2p_{\sigma(1),2} + p_{\sigma(2),1} + p_{\sigma(2),2} + p_{\sigma(3),1} + p_{\sigma(3),2}.$$

Now using this relationship we can show that $\min_\sigma F(\sigma, \tau)$ is a linear assignment problem with the following assignment matrix:

	Location		
Job	1	2	3
1	$3p_{1,1} + 2p_{1,2}$	$p_{1,1} + p_{1,2}$	$p_{1,1} + p_{1,2}$
2	$3p_{2,1} + 2p_{2,2}$	$p_{2,1} + p_{2,2}$	$p_{2,1} + p_{2,2}$
3	$3p_{3,1} + 2p_{3,2}$	$p_{3,1} + p_{3,2}$	$p_{3,1} + p_{3,2}$

Using Proposition 1, we can easily state a sufficient condition for the optimality of a schedule $\sigma \in S_n$.

PROPOSITION 3: *Let σ_0 be a schedule in S_n, and τ^* be the path vector for which $F(\sigma_0, \tau^*) = \max_{\tau \in T} F(\sigma_0, T)$. If $F(\sigma_0, \tau^*) = \min_{\sigma \in S_n} F(\sigma, \tau^*)$, then σ_0 is an optimal schedule.*

PROOF: From Proposition 1 we know that $\min_{\sigma \in S_n} F(\sigma, \tau^*)$ is a lower bound for the problem. From the statement of the proposition we have

$$F(\sigma_0, \tau^*) = \max_{\tau \in T} F(\sigma_0, \tau) = \min_{\sigma \in S_n} F(\sigma, \tau^*).$$

Therefore, σ_0 is an optimal schedule. □

Proposition 3 can be useful for the class of scheduling problems for which (6) holds as an equality. Our discussion below identifies such a class of problems in ordered flow shops and demonstrates how our modeling framework can be applied to derive one of the known results in the ordered flow shop literature (see Panwalkar and Khan [14]).

DEFINITION 1: *A flow-shop problem is ordered if the following constraints are satisfied:*

1. *$p_{i,t} > p_{k,t}$, for jobs i and k, and for some machine t, also implies $p_{i,j} \geq p_{k,j}$, $j = 1, \ldots, m$,*
2. *$p_{r,j} > p_{r,t}$, for machines j and t, and for some job r, also implies $p_{i,j} \geq p_{i,t}$, $i = 1, \ldots, n$.*

PROPOSITION 4: *Consider the ordered flow shop problem. If the largest processing time of each job is on the first machine, then the $n/m/P/\Sigma C_i$ problem belongs to the special class of scheduling problems for which relationship (6) holds as an equality. The SPT (shortest processing time) sequencing rule provides the optimal solution to the above ordered flow shop problem.*

PROOF: Using results from Panwalkar and Khan [14], we can write the critical path vector $\tau^* = (\tau^*(i), i = 1, \ldots, n)$ of the optimal schedule σ_{SPT} in the following form:

$$\tau^* = ((t(i, j) = 1, j = 1, \ldots, i - 1), i = 1, \ldots, n).$$

Let us now consider the corresponding linear assignment problem for τ^*:

$$(AP): \min_{\sigma \in S_n} F(\sigma, \tau^*).$$

The entries to the assignment matrix $A^{\tau^*} = (a^{\tau^*}_{r,j})$ are given by

$$a^{\tau^*}_{r,j} = (n + 1 - j)p_{r,1} + \sum_{i=2}^{m} p_{r,i}, \qquad r = 1, \ldots, n; \qquad j = 1, \ldots, n.$$

Note that the second term of $a^{\tau^*}_{r,j}$ is independent of j. Therefore the optimal solution to (AP) sequences the jobs in the SPT order, based on their processing times on Machine 1. Then, we have that for the resulting schedule the following relationship holds:

$$\min_{\sigma \in S_n} F(\sigma, \tau^*) = F(\sigma_{SPT}, \tau^*) = \max_{\tau \in T} F(\sigma_{SPT}, \tau).$$

The last equality in the above relationship holds by the definition of τ^* as the critical path vector for σ_{SPT}. Therefore, relationship (6) holds as an equality. □

3. GAME-THEORETIC INTERPRETATION OF THE $n/m/P/\Sigma\ C_i$ PROBLEM

The results of Section 2 point out the min-max nature of the $n/m/P/\Sigma\ C_i$ problem. A problem with a similar min-max formulation has appeared in very different context, that of game theory. This fact motivated us to provide a game-theoretic interpretation of the $n/m/P/\Sigma\ C_i$ problem. We will use this interpretation to directly apply an integer programming formulation of the corresponding game-theoretic problem to formulate our sequencing problem.

We may relate the formulation (5) of permutation flow shop problems to a two-person zero-sum game. Player 1, referred to as the path maker, has a pure strategy set T (set of path vectors) and Player 2, referred to as the schedule maker, has a pure strategy set S_n. For Player 1 the payoff of the game for pure strategies $\sigma \in S_n$ and $\tau \in T$ is $F(\sigma, \tau)$. Similarly, the loss of Player 2 is $F(\sigma, \tau)$.

In a two-person zero-sum game, there exists a pure strategy for player 1 by which he is certain to obtain at least a payoff of $\underline{f}$, which is given by

$$\underline{f} = \max_{\tau \in T} \min_{\sigma \in S_n} F(\sigma, \tau).$$

Similarly, there exists a pure strategy for the schedule maker that offers him

the least loss independently of the actions of the path maker. The minimal loss for the schedule maker is given by $\bar{f}$, where

$$\bar{f} = \min_{\sigma \in S_n} \max_{\tau \in T} F(\sigma, \tau).$$

Observe that $\bar{f}$ is the optimal solution of the permutation flow shop problem. From the results of game theory (von Neumann and Morgenstern [18], we can immediately conclude that $\underline{f} \leq \bar{f}$, an indirect verification of Proposition 1. In case the sufficient condition stated in Proposition 3 holds, i.e., $\underline{f} = \bar{f}$ we say that the game has a saddle or equilibrium point.

The $n/m/P/\Sigma\ C_i$ problem can also be analyzed as a majorant game. In such a game the schedule maker makes his move and then the path maker chooses his strategy in full knowledge of the other player's move. It can be seen that the equilibrium point of the majorant game is given by

$$f^* = \min_{\sigma \in S_n} \max_{\tau \in T} F(\sigma, \tau),$$

and is equal to the optimal solution of the $n/m/P/\Sigma\ C_i$ problem. The equilibrium value of the majorant game, and as a consequence the solution to the $n/m/P/\Sigma\ C_i$ problem, can be found by solving the following integer programming formulation:

$$(IP): f^* = \min M,$$

subject to

$$\sum_{\sigma \in S_n} F(\sigma, \tau) y_\sigma \leq M, \qquad \tau \in T \tag{11}$$

$$\sum_{\sigma \in S_n} y_\sigma = 1, \tag{12}$$

$$y_\sigma \in \{0, 1\}, \tag{13}$$

where f^* is the value of the game (optimal value of the sum of completion times), and $y_\sigma = 1$ if σ is the optimal pure strategy (schedule) for the schedule maker.

The simple interpretation of the $n/m/P/\Sigma\ C_i$ problem as a majorant game has led us to formulation (IP). We now proceed to write an equivalent formulation (IP), which is more convenient for our further discussion purposes. We can always think of a permutation schedule $\sigma \in S_n$ as a one-to-one assignment of n numbers (or n positions in a sequence) to n jobs. Therefore we can introduce the set of binary integer decision variables $x_{i,j} \in \{0, 1\}$, to equivalently describe a schedule as

$$(\sigma \in S_n) \longleftrightarrow X = (x_{i,j}),$$

where

$$x_{i,j} = \begin{cases} 1, & \text{if } \sigma(j) = J_i, \\ 0, & \text{otherwise.} \end{cases}$$

Using the above set of decision variables and relationship (10) we can state formulation (*IP*) in an equivalent way as follows:

$$(IP1)\colon f^* = \min M,$$

subject to

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j}^{\tau} x_{i,j} \leq M, \qquad \tau \in T, \tag{14}$$

$$\sum_{i=1}^{n} x_{i,j} = 1, \qquad j = 1, \ldots, n, \tag{15}$$

$$\sum_{j=1}^{n} x_{i,j} = 1, \qquad i = 1, \ldots, n, \tag{16}$$

$$x_{i,j} \in \{0, 1\}, \qquad i, j = 1, \ldots, n. \tag{17}$$

Although the assignment matrix of a given path τ (i.e., $a_{i,j}^{\tau}$'s) can be determined in $O(n^3m)$ time, (*IP*1) cannot be directly solved with the available integer programming software due to the number of constraints in constraint set (14), which increases exponentially with the number of jobs and stations. In the next section we present a new lower bounding scheme for the $n/m/P/\Sigma\ C_i$ problem based on a relaxation of the (*IP*1) formulation which requires generation of a finite number of constraints in constraint set (14).

4. A NEW LOWER BOUND GENERATION SCHEME FOR THE $n/m/P/\Sigma\ C_i$ PROBLEM

In order to develop lower bounds for the $n/m/P/\Sigma\ C_i$ problem, we are going to follow a two step relaxation procedure. At first, we restrict our attention only to a subset of T^* of the path vector set T. The relaxed formulation (*IP*1) is given as follows:

$$(RIP1)\colon f_r^* = \min M,$$

subject to

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j}^{\tau} x_{i,j} \leq M, \qquad \tau \in T^*, \tag{18}$$

(15)–(17).

At the second step of our relaxation approach we dualize the constraint set (18). Let $\lambda = \{\lambda_\tau, \tau \in T^*\}$ be the set of Lagrange multipliers. Then, the Lagrangian relaxation of ($RIP1$) is

$$(LR(\lambda)): f(\lambda) = \min M\left(1 - \sum_{\tau \in T^*} \lambda_\tau\right) + \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\sum_{\tau \in T^*} \lambda_\tau a_{i,j}^\tau\right) x_{i,j},$$

subject to (15)–(17). Observe that $f(\lambda)$ is bounded only if $\Sigma_{\tau \in T^*} \lambda_\tau = 1$. Without loss of generality, we can always assume that this is the case because we can normalize the Lagrange multipliers λ_τ, $\tau \in T^*$, by their corresponding summation $\Sigma_{\tau \in T^*} \lambda_\tau$. Therefore, we can always state ($LR(\lambda)$) as an equivalent linear assignment problem as follows:

$$(LRAP(\lambda)): f_{AP}(\lambda) = \min \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\sum_{\tau \in T^*} \lambda_\tau a_{i,j}^\tau\right) x_{i,j},$$

subject to (15)–(17). Let f_r^{LP} be the optimal objective function value of the linear programming relaxation of ($RIP1$). Using standard results from the Lagrangian relaxation theory (for a textbook reference see Nemhauser and Wolsey [13]), we can state the following property:

PROPERTY 1:

$$f_r^{LP} = \max_{\lambda: \Sigma_{\tau \in T^*} \lambda_\tau = 1} f_{AP}(\lambda).$$

From our above discussion, we can conclude that f_r^* is a legitimate lower bound for the $n/m/P/\Sigma\, C_i$ problem, i.e., $f_r^* \leq f^*$. We can always approximate f_r^* from below by solving the linear programming relaxation of ($RIP1$). From well-known results of Lagrangian relaxation theory, and since the constraint set of ($LRAP(\lambda)$) has a network structure, we are guaranteed that our Lagrangian bound cannot be better than the linear programming relaxation bound. The solution of the linear programming relaxation of ($RIP1$), however, proved to be computationally prohibitive when we implemented it within a branch-and-bound approach. Thus, we decided to use a slightly different path. To obtain f_r^{LP}, or a legitimate approximation of it (i.e., a lower bound on it), we solve the linear assignment problem ($LRAP(\lambda)$) for an appropriately chosen vector $\lambda = \{\lambda_\tau, \tau \in T^*\}$. The use of a subgradient optimization method helps us update the multiplier vector λ, and within a small number of iterations generates a very good approximation of f_r^{LP}. Our subgradient optimization procedure starts with an initial multiplier vector $\lambda^0 = \{\lambda_\tau^0, \tau \in T^*\}$ and then defines iteratively a sequence of multiplier vectors λ^τ by use of the following rule

$$\bar{\lambda}_\tau^{t+1} = \max\left\{0, \lambda_\tau^t + \frac{\beta^t \mu_\tau (\tilde{f} - f_{AP}(\lambda^t))}{\|\mu\|^2}\right\},$$

where β^t is a scalar, $\tilde{f}$ is an upper bound on f_r^{LP}, $\mu = (\mu_\tau, \tau \in T^*)$ is a subgradient for $f_{AP}(\lambda^t)$, and $\|\mu\|$ is the Euclidean norm of the vector μ. The notation $\bar{\lambda}^t = (\bar{\lambda}_\tau^t, \tau \in T^*)$ denotes the multiplier vector before the required normalization procedure for our method. The above subgradient optimization procedure is an adaptation of the one presented in Goffin [5], and based on results therein it is guaranteed to converge to f_r^{LP}, when $\Sigma_{t=0}^{\infty} \beta^t = \infty$ and $\lim_{t\to\infty} \beta^t = 0$. For the complete specification of the above procedure we need to describe the subgradient vector μ. This is described in Property 2 below.

PROPERTY 2: *Let* $\mu_\tau = \Sigma_{i=1}^n \Sigma_{j=1}^n a_{i,j}^\tau x_{i,j}^* - f_{AP}(\lambda)$, $\tau \in T^*$, *where* $X^* = (x_{i,j}^*)$ *is the optimal solution to the* $(LRAP(\lambda))$ *problem. Then,* $\mu = (\mu_\tau, \tau \in T^*)$ *is a subgradient for* $f_{AP}(\lambda)$.

PROOF: Let λ' be a multiplier vector, where $\Sigma_{\tau\in T^*} \lambda_\tau' = 1$. We want to show that

$$f_{AP}(\lambda) + \mu(\lambda' - \lambda)^t \geq f_{AP}(\lambda'). \tag{19}$$

Note that $(\lambda' - \lambda)^t$ denotes the transpose of the vector $(\lambda' - \lambda)$. We can rewrite the RHS of (19) as follows

$$f_{AP}(\lambda) + \mu(\lambda' - \lambda)^t = f_{AP}(\lambda) + \sum_{\tau\in T^*} \left(\left(\sum_{i=1}^n \sum_{j=1}^n a_{i,j}^\tau x_{i,j}^* - f_{AP}(\lambda) \right) (\lambda_\tau' - \lambda_\tau) \right)$$

$$= f_{AP}(\lambda) + \left(\sum_{\tau\in T^*} \sum_{i=1}^n \sum_{j=1}^n a_{i,j}^\tau x_{i,j}^* \lambda_\tau \right)$$

$$- f_{AP}(\lambda) \left(1 - \sum_{\tau\in T^*} (\lambda_\tau' - \lambda_\tau) \right).$$

Now using $\Sigma_{\tau\in T^*} \lambda_\tau$ and $\Sigma_{\tau\in T^*} \lambda_\tau' = 1$, we obtain

$$f_{AP}(\lambda) + \mu(\lambda' - \lambda)^t = \sum_{\tau\in T^*} \sum_{i=1}^n \sum_{j=1}^n a_{i,j}^\tau x_{i,j}^* \lambda_\tau' \geq f_{AP}(\lambda').$$

The last inequality follows from the fact that $x_{i,j}^*$, $i, j = 1, \ldots, n$ is a feasible solution for the $(LRAP(\lambda'))$ problem and therefore

$$\sum_{\tau\in T^*} \sum_{i=1}^n \sum_{j=1}^n a_{i,j}^\tau x_{i,j}^* \lambda_\tau'$$

is an upper bound on $f_{AP}(\lambda')$. □

We have used this new lower-bound generation scheme within a branch-and-bound algorithm. We will discuss the implementation of the new bounding technique in more detail in Section 5. The major disadvantages of the new approach are as follows.

- Each solution to the ($LRAP(\lambda)$) problem is also a feasible solution for the $n/m/P/\Sigma C_i$ problem, and the generated solution is usually very good in terms of sum of completion times. Therefore, our bounding scheme can be also used as a heuristic procedure.
- After implementing the subgradient optimization procedure for a specific node in the branch-and-bound tree we end up with a multiplier vector λ. Now we can use λ as the initial multiplier vector of the nodes generated from this particular node. This approach decreases the number of iterations required in the subgradient method to obtain a good approximation of f_r^{LP}.

In Table 1 we compare the tightness of our bounds with those of Ahmadi and Bagchi [1], which are the best currently available. Ahmadi and Bagchi [1] bounds (AB bounds) are an improved version of Bansal's [3] m machine based bounds. The overall complexity of AB bounds is $O(mn \log n)$. The worst-case complexity of our bounds, on the other hand, is $O(k(n^3 + |T^*|))$, where k is the number of iterations in the subgradient method. Note that at each iteration the multiplier vector λ can be updated in $O(|T^*|)$ time.

We have generated 20 12-job six-machine problems using integer processing times drawn from a uniform (1–100) distribution. For each problem we have generated eight machine based path vectors, i.e., $|T^*| = 8$. A machine based path for position i in the schedule σ is given as $(t(i, j) = k, j = 1, \ldots, i - 1)$, where $1 \leq k \leq m$. We have run the subgradient optimization method for four iterations, i.e., we have solved four consecutive linear assignment problems. The linear assignment solution procedure used is that of Tomizawa [17] and we

Table 1. A comparison of alternative methods.

	Lower bound			CPU ms			KK/AB	KK	Miyazaki
Problem	KK	AB	RIP1	KK	AB	RIP1	ratio	heuristic	heuristic
1	6541.77	6141	6641.11	13	5	1190	1.065	8189	8412
2	6321.20	6182	6439.66	14	6	1170	1.023	7693	8308
3	6213.27	6089	6223.96	12	4	2020	1.020	7952	8859
4	6059.72	5920	6069.82	12	5	980	1.024	6975	8103
5	5763.12	5686	5802.37	13	5	1660	1.014	7193	9203
6	5805.66	5658	5815.55	11	4	2340	1.026	6851	8033
7	6982.06	6986	6996.30	13	6	2080	0.999	8065	9192
8	6150.58	6166	6201.04	13	3	1790	0.997	7927	8847
9	7369.89	7434	7434.48	11	5	960	0.991	8442	9445
10	5750.64	5587	5830.13	14	4	1690	1.029	7107	8187
11	7319.16	7477	7540.00	11	5	660	0.979	7691	9046
12	7284.41	7292	7320.80	12	6	1950	0.999	8397	9885
13	6019.20	5860	6028.90	13	3	1500	1.027	6984	7993
14	6140.56	5818	6180.91	14	5	1060	1.055	7509	8210
15	6442.93	6253	6498.20	12	6	1690	1.030	7188	9192
16	5739.64	5579	5753.34	13	4	2500	1.029	6930	8358
17	6012.45	5864	6022.56	11	5	1020	1.025	7320	9455
18	5388.45	5154	5407.08	14	6	1620	1.045	6268	7828
19	5660.77	5396	5700.61	11	5	1430	1.049	6673	8256
20	5918.83	6171	5986.00	11	5	1900	0.959	7606	8257

have used the code developed by Burkard and Derigs [4]. The initial λ vector is taken as

$$\lambda^0 = \left(\lambda_\tau^0,\ \tau \in T^*\colon \lambda_\tau^0 = \frac{1}{|T^*|}\right),$$

and β^t is set equal to $25.0/(t+1)^2$, $t \geq 0$.

In Table 1, Column 1 tabulates the lower bounds generated by the subgradient method. AB bounds are presented in Column 2. Optimal solution to the linear programming relaxation of the (*RIP*1) problem is documented in Column 3. A comparison of Columns 1 and 3 demonstrates the rapid convergence of the subgradient method. Columns 4–6 tabulate the CPU times for these three different approaches (i.e., our subgradient procedure, the AB bounding scheme, and the solution of the linear programming relaxation of (*RIP*1) using the DDLPRS subroutine of IMSL). In Column 7 we present the ratio of our bounds to AB bounds. The average improvement is 1.9%, and in 14 problems our bounds have been strictly greater than the AB bounds. In Column 8 we report on the best upper bound obtained after four iterations of the subgradient approach. Note that at each iteration of the subgradient procedure we solve a linear assignment problem, and the solution of this problem is a feasible schedule for the $n/m/P/\Sigma\ C_i$ problem. We refer to the best schedule generated in four iterations of the subgradient approach as our heuristic solution (i.e., the KK heuristic) to the problem. Finally in Column 9, the performance of the heuristic developed by Miyazaki et al. [11] is tabulated. (The average performance of the KK heuristic is 16.5% better than that of Miyazaki heuristic.)

The overall performance of our approach is very promising. We can generate very good upper bounds (i.e., feasible schedules to the problem) while improving the currently available lower-bounding scheme. Note that AB bounds are obtained by generating a preemptive schedule for an n-job one-machine problem with ready times and sum of completion times criterion (i.e., $n/1/r_i \geq 0/\Sigma\ C_i$ problem). Therefore, the schedule that provides the lower bound in the AB method is not necessarily a feasible schedule for the original $n/m/P/\Sigma\ C_i$ problem.

However, the two approaches, i.e., our subgradient method and AB bounds, differ in terms of their worst-case computational complexities, and we will address the tradeoff between goodness of a lower bound and its computational complexity in Section 6, where we compare the performances of these two approaches when they are embedded in an implicit enumeration scheme.

5. A NEW BRANCH-AND-BOUND APPROACH TO THE $n/m/P/\Sigma\ C_i$ PROBLEM

Most research on optimal algorithm development for non-preemptive permutation flow shops has focused on enumerative methods. In this section we present a new branch-and-bound approach to the $n/m/P/\Sigma\ C_i$ problem.

Most of the enumerative approaches for permutation flow shop problems use as a branching scheme the assignment of jobs to the lth position in the schedule

at the lth level of the search tree. Thus, at a node at that level a partial schedule $(\sigma(1), \sigma(2), \ldots, \sigma(l))$ has been formed. The bounding scheme generates lower bounds on the value of all possible completions of the partial schedule. As will become clear from the description below, our branching-and-bounding schemes are distinctively different than those previously discussed in the literature.

BRANCHING SCHEME: Let S be a set of schedules and σ_0 be a schedule in S. Then the set $S\backslash\{\sigma_0\}$ can be divided into at most $n-1$ nonempty, mutually exclusive, and collectively exhaustive subsets as follows.

$$S_1 = \{\sigma \in S \quad \text{and} \quad \sigma(1) \neq \sigma_0(1)\},$$

$$S_2 = \{\sigma \in S \quad \text{and} \quad \sigma(1) = \sigma_0(1),\ \sigma(2) \neq \sigma_0(2)\},$$

$$S_3 = \{\sigma \in S \quad \text{and} \quad \sigma(1) = \sigma_0(1),\ \sigma(2) = \sigma_0(2),\ \sigma(3) \neq \sigma_0(3)\},$$

$$\vdots \qquad\qquad \vdots$$

and

$$S_{n-1} = \{\sigma \in S \quad \text{and} \quad \sigma(1) = \sigma_1(1), \ldots, \sigma(n-1) \neq \sigma_0(n-1)\}.$$

The above sets $S_1, S_2, \ldots, S_{n-1}$ satisfy the above-mentioned properties, i.e.,

$$\cup_{i=1}^{n-1} S_i = S\backslash\{\sigma_0\} \quad \text{and} \quad \cap_{i=1}^{n-1} S_i = \emptyset.$$

BOUNDING SCHEME: Let S be a set of schedules. Using the results of Section 4 we can conclude that the solutions to the $(LRAP(\lambda))$ problem, subject to the constraint that the resulting schedule is in S, provide lower bounds for this particular set S. Due to the special structure of our branching scheme, the above-mentioned constraints can be easily incorporated into the objective function, so that the resulting assignments (or schedules) are in S. For example, if $\sigma(1) \neq \sigma_0(1)$ in S, then we set $a^{\tau}_{\sigma_0(1),1} = B$, $\tau \in T^*$ during the solution of $(LRAP(\lambda))$, where B is arbitrarily large.

At each node, we choose $|T^*|$ path vectors for lower-bound generation. In the current version of the algorithm, the path vector set T^* contains different combinations of machine based paths for different positions in the schedule. For example, for location i in the schedule, the following set of integers is a machine based path: $(t(i, j) = k, k, j = 1, \ldots, i-1)$ where $1 \leq k \leq m$. However, as we move down in the search tree, the number of fixed jobs increases, and partial schedules of significant size are formed. Then, the set T^* is formed in a way that the selected paths attain the maximum possible value on these partial schedules.

The performance of our algorithm is very much affected by the selection of the initial multiplier vector λ in the subgradient optimization method. Due to the above-mentioned selection of T^*, the path vector sets of a node and its children nodes are very similar, and in most cases the multiplier vector obtained

in the last iteration of the subgradient method is a very good initial multiplier vector for the children nodes of this particular node.

For each new subset of schedules generated, i.e., each node in our search tree, we use the same approach in developing lower and upper bounds. The iterations of the subgradient method provide us with lower bounds and feasible schedules. The sum of completion times of these schedules give us new upper bounds.

FATHOMING CRITERION: If the current upper bound is less than or equal to the lower bound on that subset of schedules.

NODE TO EXAMINE NEXT CRITERION: In the current version of our algorithm we examine next the subset of schedules that has the smallest lower bound.

EXAMPLE: We will illustrate the basic ideas of the algorithm using the three-job and two-machine problem given in Table 2. Let S be the set of all schedules such that $\sigma(1) \neq 1$, i.e., $S = \{(2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)\}$. Let $|T^*| = 2$, and let τ_1 and τ_2 be given as follows:

$$\tau_1 = ((2, 0, 0), (1, 2, 0), (1, 1, 2)),$$

$$\tau_2 = ((2, 0, 0), (2, 2, 0), (2, 2, 2)).$$

The assignment matrices A^{τ_1} and A^{τ_2} for S are given in Tables 3 and 4. Let $\lambda^0 = (\frac{1}{2}, \frac{1}{2})$. Then the assignment matrix for $(LRAP(\lambda^2))$ is as given in Table 5. The solution of assignment problem is $\sigma = (2, 1, 3)$ and $f_{AP}(\lambda^0)$ is equal to 19. Now, the subgradient is given by $\mu = (20 - 19, 18 - 19) = (1, -1)$. We can also find an upper bound by computing the sum of completion times of schedule σ, which is 20. Let $\beta^0 = 2.5$; then the new multiplier vector, before the normalization, is given by

$$\bar{\lambda} = \left(\max\left\{0, \frac{1}{2} + \frac{2.5(1)(20 - 19)}{2}\right\}, \max\left\{0, \frac{1}{2} + \frac{2.5(-1)(20 - 19)}{2}\right\}\right)$$

or $\bar{\lambda} = (1.75, 0)$, and therefore $\lambda^1 = (1, 0)$. In the next iteration we solve the linear assignment problem on the matrix A^{τ_1}, and the optimal solution is $\sigma = (2, 1, 3)$ and $f_{AP}(\lambda^1) = 20$, which is equal to the upper bound. Therefore, for the set of schedules S, $\sigma = (2, 1, 3)$ is an optimal schedule.

Table 2. Example problem.

	Machine	
Job	1	2
1	1	2
2	3	1
3	4	2

Table 3. Assignment matrix $A^{[1]}$.

	Location		
Job	1	2	3
1	B	4	3
2	10	7	4
3	14	10	6

Table 4. Assignment matrix $A^{[2]}$.

	Location		
Job	1	2	3
1	B	4	2
2	12	2	1
3	18	4	2

Table 5. Assignment matrix for $(LRAP(\lambda^0))$.

	Location		
Job	1	2	3
1	B	4	3.5
2	11	4.5	2.5
3	16	7	4

6. COMPUTATIONAL PERFORMANCE OF THE BRANCH-AND-BOUND PROCEDURE

We have tested the performance of our algorithm using randomly generated problems. The method of problem generation follows that given in Lageweg, Lenstra, and Rinnooy Ken [9]; i.e., four different problem classes are considered: random problems, problems with correlation between the processing times of each job, problems for which the processing times of each job have a positive trend, and finally problems with both correlation and positive trend. In Table 6 we present the distributions of these classes, where $c(i)$ is the correlation coefficient of job i, $i = 1, \ldots, n$, with a uniform [0–4] distribution, and j is the machine index.

In Tables 7 and 8 we present the results comparing the performance of our branch-and-bound procedure to the one developed by Bansal [3], with the improved lower bounds of Ahmadi and Bagchi [1], for $n = 8$ and $n = 10$, respectively. Bansal's algorithm is a general approach which works with partial assignments, as described in the beginning of this section. Since AB bounds dominate the bounds developed by Bansal, we have used the improved bounds in the implementation of this approach. Both methods have been coded in FORTRAN, on a multiuser IBM 3081-D system.

In Tables 7 and 8 Column 1 documents the parameters of a specific set of problems, where m is the number of machines, $|T^*|$ is the number of path vectors

Table 6. Problem classes.

Problem class	Distribution	Parameters
Random (R)	Uniform	1,100
Correlation (C)	Uniform	$20c(i) + 1, 20c(i) + 20$
Trend (T)	Uniform	$12.5(j - 1) + 1, 12.5(j - 1) + 100$
Correlation/trend (CT)	Uniform	$2.5(j - 1) + 20c(i) + 1, 2.5(j - 1) + 20c(i) + 20$

Table 7. Comparison of branch-and-bound methods for $n = 8$.

Parameters	Problem class	CPU ms		No. of nodes		Storage req.	
		KK	Bansal	KK	Bansal	KK	Bansal
$m = 2$ $\lvert T^* \rvert = 3$ Steps = 2	R	619	624	108.20	131.65	15.15	95.65
	C	115	286	18.05	53.40	2.80	34.30
	T	456	516	78.45	102.80	11.40	74.80
	CT	127	302	19.35	52.50	3.40	40.55
$m = 3$ $\lvert T^* \rvert = 4$ Steps = 3	R	1262	1019	190.15	186.25	23.05	134.75
	C	268	506	43.70	85.25	5.75	65.60
	T	921	677	135.80	127.45	16.20	85.80
	CT	268	452	41.15	76.05	5.70	58.60
$m = 4$ $\lvert T^* \rvert = 5$ Steps = 3	R	3860	2584	452.10	366.45	58.25	236.65
	C	289	500	34.20	61.30	5.50	43.60
	T	920	608	108.25	79.55	12.50	57.55
	CT	609	745	76.15	100.75	11.10	68.20
$m = 5$ $\lvert T^* \rvert = 6$ Steps = 4	R	4930	3051	448.25	312.85	53.05	215.40
	C	447	682	40.65	64.55	6.70	44.35
	T	1604	1104	151.25	114.15	19.75	80.00
	CT	1055	1074	99.95	108.65	13.80	76.20

Table 8. Comparison of branch-and-bound methods for $n = 10$.

Parameters	Problem class	CPU ms		No. of nodes		Storage req.	
		KK	Bansal	KK	Bansal	KK	Bansal
$m = 2$ $\lvert T^* \rvert = 3$ Steps = 3	R	6,119	7,381	714.84	1,009.70	75.25	785.80
	C	461	672	52.75	129.75	6.45	106.25
	T	4,199	5,337	491.25	675.70	61.50	524.50
	CT	562	808	76.15	169.85	9.45	128.30
$m = 3$ $\lvert T^* \rvert = 4$ Steps = 3	R	10,045	8,094	1,095.95	883.50	102.45	681.10
	C	673	1,140	69.95	141.80	9.65	112.70
	T	6,183	5,107	699.10	590.80	63.65	473.25
	CT	2,575	3,354	302.35	412.15	30.10	325.45
$m = 4$ $\lvert T^* \rvert = 5$ Steps = 3	R	28,602	25,396	2,233.75	1,783.65	197.75	1,416.25
	C	2,064	3,281	211.05	290.40	25.40	28.20
	T	5,705	4,510	562.80	406.45	56.70	324.15
	CT	2,636	3,056	231.05	270.40	24.45	210.70
$m = 5$ $\lvert T^* \rvert = 6$ Steps = 4	R	44,903	32,797	3,394.10	2,515.55	304.85	1,977.00
	C	2,281	3,298	176.55	241.30	22.00	186.05
	T	6,163	4,600	485.00	312.80	50.20	251.55
	CT	5,859	7,514	472.90	544.30	59.65	418.65

used in our lower-bound generation scheme and "Steps" is the number of iterations in the subgradient method. Column 2 of Tables 7 and 8 denotes the problem class. For each class we have generated 20 problems; hence each entry in Tables 7 and 8 represents the average of 20 problems. In Columns 3 and 4 of Tables 7 and 8, we compare the computational effort required by our method (KK) and by Bansal's procedure, respectively. In Columns 5 and 6 we present the average number of nodes generated by the two different approaches. In Columns 7 and 8, we report the average maximum number of nodes stored by our method, and by Bansal's method, respectively. Note that the maximum number of nodes stored denotes the maximum number of active nodes (i.e., nodes that have been generated but not eliminated) at any time point during the actual run of the branch-and-bound procedure. This could be an important factor for large-size problems, especially if the storage constraints are more pressing than the CPU time constraint.

In two-machine eight-job and two-machine ten-job problems our approach dominates Bansal's method, in terms of both CPU times and the number of nodes, in all of the four problem classes. For $m = 3$, 4, and 5, our method dominates Bansal's approach in problems with correlation (C), and with correlation and trend (CT). In random problems (R) and problems with trend (T), Bansal's method dominates our approach. Apparently for these problems the subgradient method cannot generate a good approximation of f_r^{LP}, within the allowed number of iterations. Increasing the number of iterations in the subgradient method has resulted in larger CPU times, although the number of nodes generated by our approach dominated that of Bansal's method for some problems in classes (R) and (T). In all problems our approach has dominated Bansal's method in terms of storage requirement. The dominance of our method in this category also indicates that our approach generates very good upper bounds early in the search tree so that the number of active nodes is kept at a minimum level.

In order to illustrate the relative importance of this property, we have used our optimal method as a heuristic for large size problems. In Table 9, we doc-

Table 9. Heuristic run of the optimal algorithm.

Parameters	Stopping criterion	Problem class	CPU ms	No. of nodes	Storage req.	$\frac{UB - LB}{LB}$
$n = 20$	$CPU \geq 5{,}000$	R	5,000	508.45	202.20	0.1716
$m = 5$	or	C	1,946	145.90	81.55	0.0192
$\|T^*\| = 6$	$\frac{UB - LB}{LB} \leq 0.02$	T	5,000	508.89	105.45	0.0601
Steps = 5		CT	4,615	446.95	244.00	0.0312
$n = 25$	$CPU \geq 7{,}500$	R	7,500	512.50	214.65	0.1797
$m = 5$	or	C	3,093	138.55	89.45	0.0170
$\|T^*\| = 6$	$\frac{UB - LB}{LB} \leq 0.02$	T	6,849	488.35	110.10	0.0599
Steps = 5		CT	7,500	510.05	292.05	0.0378
$n = 30$	$CPU \geq 12{,}500$	R	12,500	513.10	243.65	0.1888
$m = 5$	or	C	7,476	215.45	136.35	0.0188
$\|T^*\| = 6$	$\frac{UB - LB}{LB} \leq 0.02$	T	11,657	514.20	119.90	0.0801
Steps = 5		CT	12,500	487.70	283.50	0.0363

ument the performance of the heuristic run of our optimal method. In Table 9, Column 1 tabulates the parameters of a specific set of problems. Column 2 documents the stopping criterion used in the heuristic run of our optimal algorithm. For example, in 20-job five-machine problems, we have terminated the optimal algorithm when the total CPU time exceeded 5000 ms or when we found an upper bound which was within 2% of the lower bound. Column 3 documents the different problem classes. For each class we have generated 20 problems, and each entry represents the average of 20 problems. The average CPU time, number of nodes generated, and maximum number of nodes stored for each problem class are presented in Columns 4–6, respectively. In Column 7, we report the average performance of the heuristic run of the our optimal approach. Except for random problems, the heuristic has performed exceptionally well.

We have also used Bansal's approach for the same problems; however, the only complete schedules generated by the Bansal's method with the AB bounds have been the schedules generated when developing the lower bounds on the first machine, and the heuristic schedules developed by our approach dominated all of these schedules.

7. CONCLUSION

In this article we have developed a new modeling framework for the permutation flow shop scheduling problem for the minimization of the sum of completion times. The modeling framework is based on a game-theoretic interpretation of the problem. A Lagrangian relaxation approach on the resulting integer programming formulation is used to develop lower bounds for the problem. Those bounds are implemented in an efficient branch-and-bound algorithm.

The methodology developed in this article is quite flexible and it can handle, with minor modifications, a more general class of problems for the sum-of-completion-times criterion in permutation flow shops: the class of scheduling problems with finite buffer capacities. We have already undertaken research in this direction and we will report our research progress on these problems in a subsequent article.

REFERENCES

[1] 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*, **44**, 331–336 (1990).

[2] Baker, K.R., "A Comparative Study of Flow Shop Algorithms," *Operations Research*, **23**(2), 62–73 (1975).

[3] Bansal, S.P., "Minimizing the Sum of Completion Times of n Jobs over m Machines in a Flow Shop—a Branch-and-Bound Algorithm," *AIIE*, **9**, 306–311 (1977).

[4] Burkard, R.E., and Derigs, U., *Assignment and Matching Problems: Solution Methods with FORTRAN Programs*, Springer-Verlag, Berlin, 1980.

[5] Goffin, J.L., "On Convergence Rates of Subgradient Optimization Methods," *Mathematical Programming*, **13**, 329–347 (1977).

[6] Gonzalez, T., and Sahni, S., "Flow Shop and Job Shop Schedules: Complexity and Approximation," *Operations Research*, **26**(1), 36–52 (1978).

[7] Ignall, E., and Schrage, L.E., "Application of the Branch-and-Bound Technique to Some Flow Shop Problems," *Operations Research*, **13**, 400–412 (1965).
[8] Kohler, W.H., and Steiglitz, K., "Exact, Approximate, and Guaranteed Accuracy Algorithms for the Flow Shop Problem $n/m/F/\bar{F}$," *Journal of the Association for Computing Machinery*, **22**, 106–114 (1975).
[9] Lageweg, B.J., Lenstra, J.K., and Rinnooy Kan, A.H.G., "A General Bounding Scheme for the Permutation Flowshop," *Operations Research*, **26**, 53–67 (1978).
[10] Lawler, E.L., *Combinatorial Optimization: Networks and Matroids*, Rinehart and Winston, 1976.
[11] Miyazaki, S., Nishiyama, N., and Hashimoto, F., "An Adjacent Pairwise Interchange Approach to the Mean Flow Time Scheduling Problem," *Journal of the Operations Research Society of Japan*, **21**(2), 287–299 (1978).
[12] Monma, C.L., and Rinnooy Kan, A.H.G., "A Concise Survey of Efficiently Solvable Special Cases of the Permutation Flowshop Problem," *RAIRO*, **17**, 105–119 (1983).
[13] Nemhauser, G.L., and Wolsey, L.A., *Integer and Combinatorial Optimization*, John Wiley and Sons, New York, 1988.
[14] Panwalkar, S.S., and Khan, A.W., "An Ordered Flow Shop Sequencing Problem with Mean Completion Time Criterion," *International Journal of Production Research*, **14**(5), 631–635 (1976).
[15] Szwarc, W., "Permutation Flowshop Theory Revisited," *Naval Research Logistics Quarterly*, **26**, 557–570 (1978).
[16] Szwarc, W., "The Flow Shop Problem with Mean Completion Time Criterion," *IIE Transactions*, **15**(2), 172–176 (1983).
[17] Tomizawa, N., "On Some Techniques Useful for Solution of Transportation Network Problems," *Networks*, **2**, 179–194 (1972).
[18] von Neumann, J., and Morgenstern, O., *Theory of Games and Economic Behavior*, Princeton University Press, Princeton, NJ, 1947.

Manuscript received April 29, 1991
Revised manuscript received December 31, 1992
Accepted April 27, 1993