Pergamon

S50360-8352(96)00012-5

Computers ind. Engng Vol. 30, No. 3, pp. 433–447, 1996
Copyright © 1996 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0360-8352/96 $15.00 + 0.00

A TABU SEARCH PROCEDURE FOR PERIODIC JOB SHOP SCHEDULING

JU-SEOG SONG and TAE-EOG LEE

Department of Industrial Engineering, Korea Advanced Institute of Science and Technology, 373-1 Gusung-dong, Yusong-Gu, Taejon 305-701, Korea

Abstract—We consider job shops where an identical mixture of items is repetitively produced. We discuss the sequencing problem that finds the processing order at each machine that maximizes the throughput rate of the mixtures or equivalently minimizes the cycle time. We present an effective tabu search procedure for the problem. To do this, we characterize the neighborhood structure that generates feasible solutions by reversing the order of two operations on a critical circuit in the associated graph. We develop an efficient method of approximately evaluating the cycle times of neighborhood solutions. Computational results are reported. Copyright © 1996 Elsevier Science Ltd

1. INTRODUCTION

We consider a scheduling problem for a shop that repetitively produces an identical set of different items. The set is often taken to be the smallest set that has the same proportion of items as the production requirement, called a minimal part set (MPS) [1]. The processing order for each machine is the same for each MPS. The shop is called a periodic or cyclic shop and the scheduling method is called periodic or cyclic scheduling. For example, to produce 100 units of part A, 200 units of part B, and 300 units of part C, we would take the MPS as (1A, 2B, 3C) and repeat the processing of the MPS 100 times.

Periodic scheduling is effective when setup times and costs are insignificant. Periodic scheduling is also applicable to automobile assembly lines [2], flexible manufacturing systems [3], or robot work cells [4]. Periodic scheduling has advantages over conventional scheduling methods, like batch scheduling, random order scheduling or scheduling by dispatching rules, and scheduling the whole job set at once. The advantages include better utilization of machines [5], reduction of the scheduling problem size, predictable shop behavior and simplified flow control, continuous and smooth supply of complete part sets for downstream assembly, and timely delivery and reduced inventory [6, 7].

In periodic scheduling, when the MPS is given and the operations are assigned to each machine, the processing order of the operations at each machine must be specified. The usual measure is the throughput rate (or its reciprocal, cycle time) at which each MPS is produced. The problem of determining a processing sequence that minimizes the cycle time is called a periodic sequencing problem.

There are studies on versions of periodic scheduling problems including [8–14]. Kamoun and Sriskandarajah [2] discuss computational complexity of sequencing problems for versions of periodic shops that involve additional scheduling constraints such as no-wait of jobs between machines, minimum total wait of jobs, and blocking due to finite or no buffer. For basic periodic shops without such scheduling constraints, Hall *et al.* [15] characterize computational complexity of each class of sequencing problems. They show that the job shop sequencing problem where each job has at most two operations is polynomially solvable, however, other general job shop sequencing problems are NP-complete. They also show that for periodic flow shops, the minimum cycle time does not depend on the sequencing decision at all. For general job shops, one would use the branch and bound procedure of Hanen [10] that is developed for a version of periodic scheduling problem for task scheduling in a pipelined computer architecture. However, Hanen [10] reports that the procedure is unsuitable for large problem instances of periodic job shops even beyond problems with 10 jobs and 5 machines. Thus, it remains to develop efficient sequencing algorithms for general periodic job shops.

For periodic job shops without scheduling constraints, Lee and Posner [6] show that for a given sequence there exists a schedule with the minimal cycle time among the schedules that repeat an identical schedule pattern for each MPS. Such minimal cycle time is considered as the minimum of the schedule widths of the machines for a single MPS. They also show that for a given sequence, the minimal cycle time is the critical circuit ratio of the graph associated with the processing sequence of the MPS. Thus, the periodic sequencing problem reduces to a sequencing problem of a single MPS that minimizes the new performance measure, the cycle time or the critical circuit ratio. We have a new class of scheduling problems with a new measure, cycle time.

There have been numerous studies on the conventional job shop scheduling problem (CJSSP) that minimizes the makespan measure. The problem has been a challenging combinatoiral problem. Recently, the shifting bottleneck procedure [16], sophisticated branch and bound procedures [17, 18], simulated annealing [19], and tabu search procedures [20–22] have been reported to be successful for large problems even with more than 15 jobs and 15 machines. The tabu search procedures are based on rather easier characterizations of the neighborhood solutions and fast evaluation methods for the neighborhood solutions. The basic periodic job shop scheduling problem without scheduling constraints (PJSSP) has basically the same problem structure as the CJSSP and the only difference is the performance measure. The cycle time measure is determined by the critical circuits in a graph while the makespan measure is characterized by the longest paths in a graph. We thus expect that the existing results on the search algorithms for CJSSP can be extended to PJSSP.

In this article, we develop an effective tabu search procedure for PJSSP that has no scheduling constraints like no-wait, minimum wait, and finite buffer. To focus on the sequencing issue, we assume that the MPS is determined and that the operations are assigned to the machines. Further assumptions include deterministic processing times, infinite buffers, negligible setup and transportation times, and no machine breakdowns. We characterize the neighborhood structure that generates a feasible solution by reversing the order of two operations on a machine that correspond to nodes of a critical circuit of the associated graph. To do this, we extend the results on the neighborhood structure on CJSSP of Taillard [20] to PJSSP. We make use of structural properties of critical circuits and their analogies with longest paths for CJSSP. Since the neighborhood evaluation method based on computation of the critical cirucit ratio requires a relatively high order of computational steps, we develop an efficient approximate evaluation method that is based on an approximate cycle time evaluation method. Computational results are reported.

2. PROBLEM DEFINITION

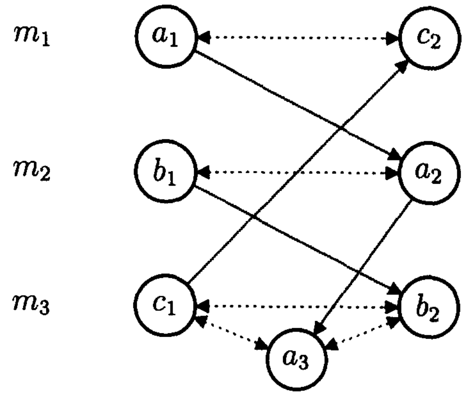
We first review the PJSSP model and the results of Lee [23], Lee and Posner [6], and Hall *et al.* [15] that are essential for development of scheduling algorithms for PJSSP.

PJSSP is defined as follows. Let J be the set of jobs (or items) that comprises an MPS. Jobs can undergo one or multiple operations. Let N be the set of all operations of an MPS, M be the set of machines. Operation $i \in N$ has processing time p_i. Each operation is assigned to a machine. Subset $N_m \subset N$ represents the set of operations that are assigned to machine $m \in M$. They may require technological processing orders between the operations of a given job. An ordered pair of operations (i, j) represents that operation $i \in N$ precedes operation $j \in N$. The set of ordered pairs that correspond to the required processing orders is denoted by A. The sequencing decision determines the processing order in which the operations assigned to each machine are processed.

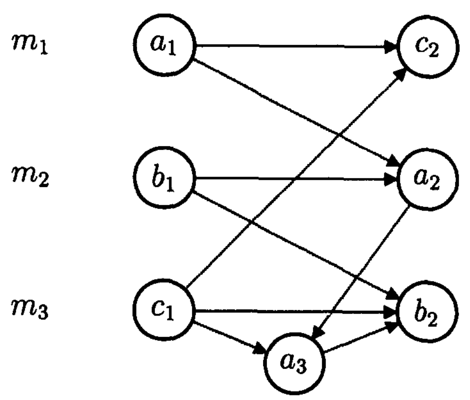
For each two operations i and j in N_m, there are two alternative processing orders, (i, j) or (j, i), which together are called a disjunctive pair. We let $D \equiv \{(i, j), (j, i) | i, j \in N_m, m \in M\}$, the set of disjunctive pairs. The sequencing decision is thus to choose one from each disjunctive pair of arcs in D so that no cyclic ordering is formed and the required precedence relations given by A are not violated (see Balas [24]). We denote by S ($S \subset D$) the set of ordered pairs of nodes that is selected from D by a sequencing decision.

A PJSSP can be represented by a directed graph. A directed graph, denoted by $G(N, A \cup D)$, that has node set N and arc set $A \cup D$ is called a scheduling graph (see Balas [24]). A sequencing decision selects an acyclic subgraph $G(N, A \cup S)$ from $G(N, A \cup D)$ by taking an arc from each disjunctive pair of arcs so that S does not violate the required precedence relations of A and $G(N, A \cup S)$ does not include any circuit. The graph $G(N, A \cup S)$ may include many redundant arcs. For instance, consider three nodes $i, j, k \in N_m$ for some machine m. The scheduling graph includes three disjunctive pairs of arcs, $((i,j),(j,i))$, $((i,k),(k,i))$, and $((i,k),(k,j))$. Suppose that we select (i,j), (j,k), and (i,k). Then, arc (i,k) is redundant. We let E denote the subset of S that has no such redundant arcs. Hence, $G(N, A \cup E)$ is the graph obtained from $G(N, A \cup S)$ by taking only those arcs from S that connect successive operations on the same machine. An example of sequencing decision is given below.

Example 1. Consider an MPS that consists of three jobs a, b, and c, each of which requires operations $\{a_1, a_2, a_3\}$, $\{b_1, b_2\}$, and $\{c_1, c_2\}$, respectively. Thus, $N = \{a_1, a_2, a_3, b_1, b_2, c_1, c_2\}$. The jobs are processed at machines m_1, m_2, and m_3. $M = \{m_1, m_2, m_3\}$. $N_{m_1} = \{a_1, c_2\}$, $N_{m_2} = \{b_1, a_2\}$, and $N_{m_3} = \{c_1, b_2, a_3\}$. $A = \{(a_1,a_2),(a_2,a_3),(b_1,b_2),(c_1,c_2)\}$. $D = \{(a_1,c_2),(c_2,a_1),(b_1,a_2),(a_2,b_1),(c_1,a_3),(a_3,c_1),(a_3,b_2),(b_2,a_3),(c_1,b_2),(b_2,c_1)\}$. The associated scheduling graph is presented in Fig. 1(a). A feasible sequencing decision chooses ordered pairs (a_1,c_2) on m_1, (b_1,a_2) on m_2, and $(c_1,b_2),(c_1,a_3)$, and (a_3,b_2) on m_3. Thus, $S = \{(a_1,c_2),(b_1,a_2),(c_1,b_2),(c_1,a_3),(a_3,b_2)\}$. The resulting graph $G(N, A \cup S)$ is given in Fig. 1(b). Notice that arc (c_1,b_2) is redundant.



(a) $G(N, A \cup D)$



(b) $G(N, A \cup S)$

Fig. 1. A scheduling graph and a selected graph.

For a given sequence, there can be a lot of schedules depending on the starting times of operations. The starting time of operation i of the r-th MPS is represented by x_i^r. A schedule is called *stable* if $x_i^{r+1} - x_i^r = \mu$ for some constant μ and for all $i \in N$ and $r \geqslant 1$. The definition implies that the timing pattern repeats every MPS.

Lee and Posner [6] define a cycle time measure that can evaluate all possible schedules regardless of its schedule pattern and the number of MPSs to be produced. Let $\alpha(m)$ and $\beta(m)$ be the first and the last operation of an MPS on machine m, respectively. For a given schedule $\{x_i^r | i \in N, r = 1, 2, \ldots, n\}$ of n MPSs, the *cycle time* is given as

$$\mu(n) = \max_{m \in M} \frac{x^n_{\beta(m)} + p_{\beta(m)} - x^1_{\alpha(m)}}{n}. \tag{1}$$

Observe that $x^n_{\beta(m)} + p_{\beta(m)} - x^1_{\alpha(m)} m$ is the time it takes to process n MPSs on machine m. The cycle time can be interpreted as the average time taken to produce an MPS.

For a sequence, the *minimal* cycle time is defined to be the minimum of the cycle times of all feasible schedules. The minimal cycle time of n MPSs for a given sequence s is denoted by $\mu_s^*(n)$. The following lemma of Lee and Posner[6] reduces PJSSP to a sequencing problem of a single MPS with a new measure, cycle time.

Lemma 1. (Lee and Posner [6]) *For a given sequence s, there always exists a stable schedule with the minimal cycle time. Further,* $\mu_s^*(n) = \mu_s^*(1)$ *for all* $n \geqslant 1$. *Finally,* $\mu_s^*(1)$ *and a stable schedule with cycle time* $\mu_s^*(1)$ *are computed in* $O(|N|^3)$ *steps.*

The value of $\mu_s^*(1)$ can be interpreted as the *schedule width* because it is the maximum of the times it takes to process the operations of a single MPS on each machine. We let $\mu_s^* \equiv \mu_s^*(1)$ and simply call it the cycle time measure for the single MPS scheduling problem.

We introduce a graph of Lee and Posner [6], modified from $G(N, A \cup E)$, that explains the periodic sequencing problem. Let $R \equiv \{(\beta(m), \alpha(m)) | m \in M\}$. Each (i,j) in R represents the precedence relation between the last operation i of an MPS on a given machine and the first operation j of the next MPS on the same machine. It represents the recycling of each machine. By adding the arc set R to the arc set of $G(N, A \cup E)$, we have a graph $G(N, A \cup E \cup R)$. For this graph, associated with each arc $(i,j) \in E$ are two weights. The first weight, p_i, corresponds to the processing time of operation i. The second weight is τ_{ij}, where

$$\tau_{ij} = \begin{cases} 1 & \text{if } (i,j) \in R, \text{ and} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The resulting graph $G(N, A \cup E \cup R)$ with the weights is called a *precedence constraints graph* (PCG). A sequence defines a PCG and vice versa. The PCG for Example 1 is illustrated in Fig. 2(a). Figure 2(b) is its expanded graph for each MPS. The second weights are not represented for convenience. The curved arcs are *recycling arcs*.

Lee and Posner [6] characterize the cycle time measure using the PCG.

Lemma 2. (Lee and Posner [6]) *For a given sequence s, the minimal cycle time* μ_s^* *is the same as the maximal ratio of the sum of the first weights of the arcs to the sum of the second weights over all the circuits in the PCG.*

A circuit in a PCG that has the maximal ratio is called a *critical circuit* and the maximal ratio is called the *critical circuit ratio*. The sequencing problem can be viewed as the problem of designing a PCG from the scheduling graph so that the critical circuit ratio is minimized over all possible sequences.

Lee [23] shows that the sequencing problem can be decomposed by each strongly connected component in the scheduling graph $G(N, A \cup D)$ and that the minimum cycle time is the maximum of the minimum cycle times of the decomposed subproblems. The computational effort is reduced significantly if $G(N, A \cup D)$ is decomposed into many smaller strongly connected components. Further, the operations of each machine belong to the same strongly connected component. Consequently, the sequencing problem is decomposed by subsets of machines each of which

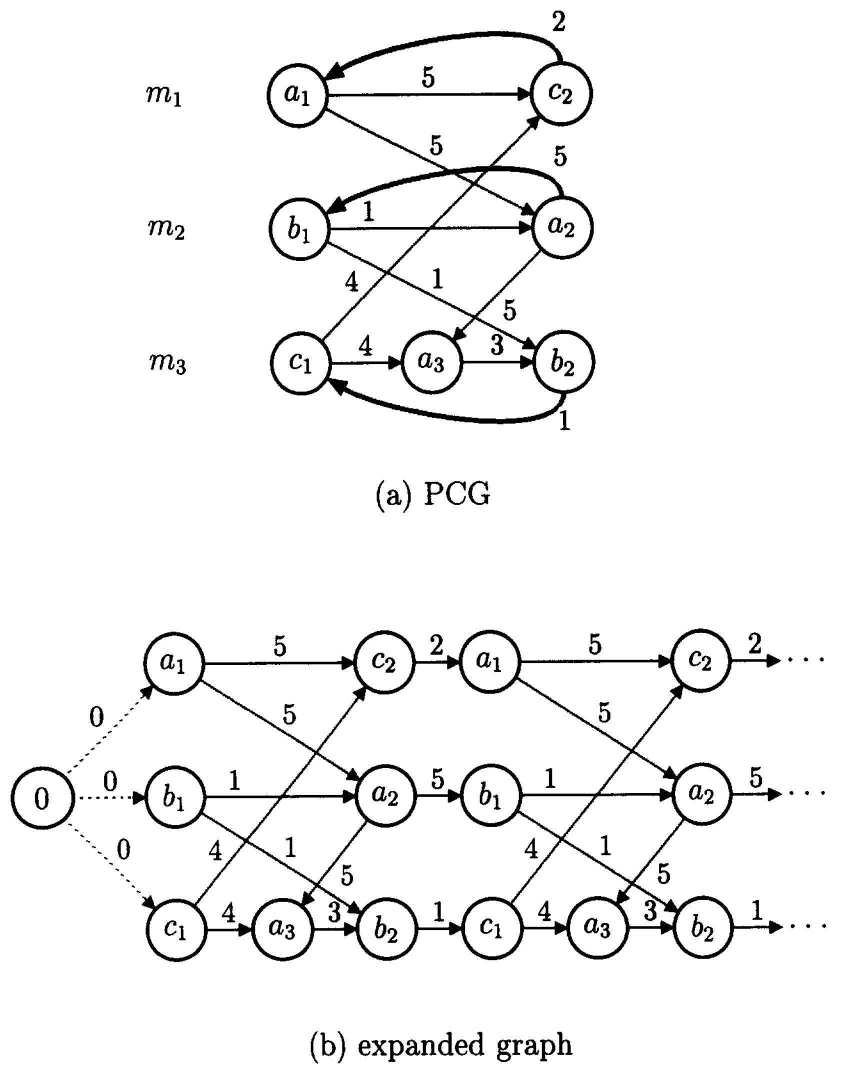


Fig. 2. A PCG and expanded graph.

processes the operations of each strongly connected component. Therefore we consider only the case for which the scheduling graph is strongly connected.

3. THE TABU SEARCH PROCEDURE

The tabu search method is a local search based meta-strategy that has been applied to numerous combinatorial problems. The method uses a special memory structure to drive a solution to the global optimum through iterations. At each iteration, all or part of the neighborhood solutions are evaluated, and the most promising one is selected as the next solution. During the evaluation, the short term memory structure prevents the repetition of the recently visited solutions. The long term memory structure is used to exploit the information on the previous solution trajectory (see Glover [25, 26] for a complete description).

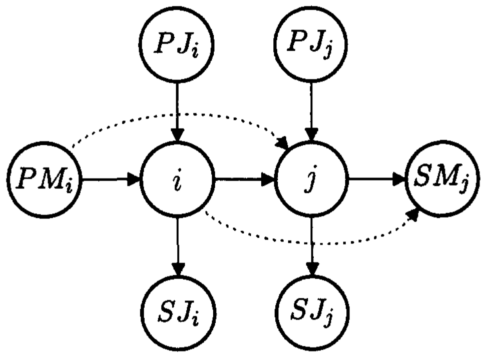
To apply the tabu search method to PJSSP, we first extend the results of Taillard [20] on the neighborhood structure of CJSSP to PJSSP. Then, by making use of the structural properties of critical circuits in the PCG, we develop a fast approximate neighborhood evaluation method.

3.1. The neighborhood structure

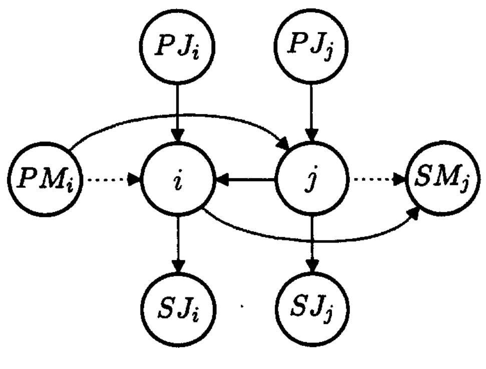
Consider a feasible sequence s and the corresponding PCG, $G_s \equiv G(N, A \cup E \cup R)$, which is strongly connected. Since every node in G_s has at least two immediate predecessors and at least two immediate successors, we introduce the following notations. For node $i \in N$, let PJ_i and SJ_i be the nodes such that $(PJ_i, i) \in A$ and $(i, SJ_i) \in A$, respectively. They represent the immediate predecessor and successor (if it exists) of operation i among the operations of the same job. Similarly, let PM_i and SM_i be the nodes such that $(PM_i, i) \in E \cup R$ and $(i, SM_i) \in E \cup R$, respectively. They represent the immediate predecessor and successor of operation i among the operations on the same machine (see Fig. 3(a)). In addition, for a circuit or path π, we let $d_1(\pi) \equiv \Sigma_{(i,j) \in \pi} p_i$ and $d_2(\pi) \equiv \Sigma_{(i,j) \in \pi} p_i \tau_{ij}$ be the sums of the first and second weights, respectively.

For an arc $(i,j) \in E$, we define a move called *swap*(i,j) that generates a new sequence s' from the current sequence s by reversing the order in which i and j are processed on the same machine. Thus, swap (i,j) results in reversing arc (i,j) and replacing the arcs (PM_i, i) and (j, SM_j) by (PM_i, j) and (i, SM_j), respectively (see Fig. 3(b)). We will show by Theorems 1 and 2 that the arc to be swapped should belong to a critical circuit. An arc or a node is said to be *critical* if it belongs to a critical circuit.

Our definition of the swap move for PJSSP is similar to that of Taillard [20] for CJSSP. Taillard [20] defines the swap move that reverses the order of two adjacent operations i and j at the same machine that belong to a longest path (that is, a critical arc (i,j)). To do this, he uses two basic properties of the disjunctive graph for CJSSP that are developed by Balas [24]: (1) reversal of the selected critical arc generates a feasible sequence and (2) to improve the makespan measure, at least one selected critical arc must be reversed. Therefore, by exploiting the analogies between our swap move and Taillard's and between the longest path and the critical circuit, we extend the results of Taillard [20] and Balas [24] to PJSSP and establish the following two theorems. However, we make use of unique properties of critical circuits.



(a) before swap(i, j)



(b) after swap(i, j)

Fig. 3. The swap move.

Theorem 1. *If arc $(i,j) \in E$ is a critical arc in G_s, the new sequence s' generated from the current sequence s by swap(i,j) is a feasible sequence.*

Proof. Let $\pi^* = (p, \ldots, i, j, \ldots, p)$ be a critical of G_s in which arc (i,j) is swapped. We note that (i,j) is not a recycling arc. Suppose that there is a circuit $\pi' \in G_{s'}$ such that $d_2(\pi') = 0$. Since $d_2(\pi) \geqslant 1$ for all circuits $\pi \in G_s$, π' should contain the reversed arc (j, i). Let $\pi' = (j, i, q, \ldots, j)$. However, the segment path $(i, q, \ldots, j)$ also exists in G_s and it has no recycling arc because $d_2(\pi') = 0$. There is already a path from j to i in G_s as a segment of π^*. Consequently, there is a circuit $\pi'' = (p, \ldots, i, q, \ldots, j, \ldots, p) \in G_s$ that has $d_1(\pi'')/d_2(\pi'', > d_1(\pi^*)/d_2(\pi^*)$, where $d_2(\pi'') = d_2(\pi^*)$. By Lemma 2, this contradicts the assumption that π^* is a critical circuit. It is seen that $G_{s'}$ has no circuit with the second weight 0 if and only if sequence s' is feasible.

Theorem 2. *If a sequence s' is generated by swapping a non-critical arc $(i,j) \in E$ in G_s, the cycle time of s' cannot be shorter than that of s.*

Proof. Suppose that the swapped arc (i,j) is not critical in G_s. If neither (PM_i, i) nor (j, SM_j) is critical, a critical circuit of G_s that passes i or j, if any, should pass (PJ_i, i, SJ_i) or (PJ_j, j, SJ_j), and it also exists in $G_{s'}$ (see Fig. 3). Thus $\mu^*_{s'} \geqslant \mu^*_s$. Suppose that (PM_i, i) is critical. Then, (i, SJ_i) should be critical. By swap(i,j), a new path (PM_i, j, i, SJ_i) is created. Therefore, $\mu^*_{s'} \geqslant \mu^*_s + p_j/d_2(\pi^*)$, where π^* is a critical circuit of G_s that passes (PM_i, i). It also can be shown by a similar method that $\mu^*_{s'} \geqslant \mu^*_s + p_i/d_2(\pi^*)$ when (j, SM_j) is critical.

From Theorems 1 and 2, for a given sequence s, the set of candidate arcs to be swapped is $C(s) \equiv \{(i,j) | (i,j) \in E$ and (i,j) is critical$\}$. Then, $C(s)$ defines the neighborhood of the sequence s, where $|C(s)| \leqslant \Sigma_{m \in M}(|N_m| - 1) = |N| - |M|$.

3.2. The neighborhood evaluation method

For each new PCG that is formed by a swap move, the critical circuit ratio can be computed by the algorithm of Karp and Orlin [27] in $O(|N| |A \cup E \cup R| \log |N|)$ steps. To determine the best move, we should apply the algorithm to all possible moves of $C(s)$ that can be as many as $|N| - |M|$. Thus, we should develop an efficient approximate evaluation method. To do this, we first develop a sharp lower bound on the critical circuit ratio and by modifying the bound we obtain a fast approximation method.

We define a measure for circuits or paths of G_s. The *μ-length* or arc (i,j) is $p_i - \mu\tau_{ij}$, where μ is a real-valued parameter. For a path or circuit π in G_s, the μ-length of π is $d_1(\pi) - \mu d_2(\pi)$, and is denoted by $L_\mu(\pi)$. For given two nodes i and j, the one among the paths from i to j that has the maximum μ-length is called a *μ-longest path* from i to j. Lee and Posner [6] show the following property.

Lemma 3. (Lee and Posner [6]) *Suppose that μ^*_s is the critical circuit ratio of G_s. Then every circuit π in G_s has $L_{\mu^*_s}(\pi) \leqslant 0$. Further, $L_{\mu^*_s}(\pi^*) = 0$ if and only if π^* is a critical circuit.*

Since the μ-length increases as the parameter μ decreases, μ^*_s is interpreted as the smallest value of μ that makes G_s to have no positive μ-length circuit. Thus, the μ-longest path can be defined for $\mu \in [\mu^*_s, \infty]$. The μ^*_s-longest path starting from node i and returning to itself corresponds to the circuit in G_s that has the maximum circuit ratio (see Lemma 2) among all circuits that pass node i. We use this measure to estimate $\mu^*_{s'}$.

We develop a sharp lower bound on the critical circuit ratio of $G_{s'}$. The PCG $G_{s'}$ obtained by swap(i,j) differs from G_s only in the arcs connected to node i or j. Since the circuits in $G_{s'}$ that contains neither i nor j also appear in G_s, in order to estimate the critical circuit ratio of $G_{s'}$, it suffices to consider only the circuits that pass at least one of i and j. Let $P \equiv \{PM_i, PJ_i, PJ_j\}$ *and* $Q \equiv \{SM_j, SJ_i, SJ_j\}$. Then, for each mode $p \in P$ and $q \in Q$, we can find a μ^*_s-longest path among the paths from q to p in G_s that do not pass any of i and j. Denote such a path by ψ_{qp}. The identical path also exist in $G_{s'}$. We let ψ_{pq} be a path in $G_{s'}$ that connects p, at least one of $\{i, j\}$, and q. For instances, $\psi_{PJ_jSJ_j} = (PJ_j, j, SJ_j)$ and $\psi_{PJ_jSJ_i} = (PJ_j, j, i, SJ_i)$ (see Fig. 4(a))). Then, for some $p \in P$ and $q \in Q$, the pair of ψ_{pq} and ψ_{qp} forms a circuit π' in $G_{s'}$. Consequently, the critical circuit ratio of

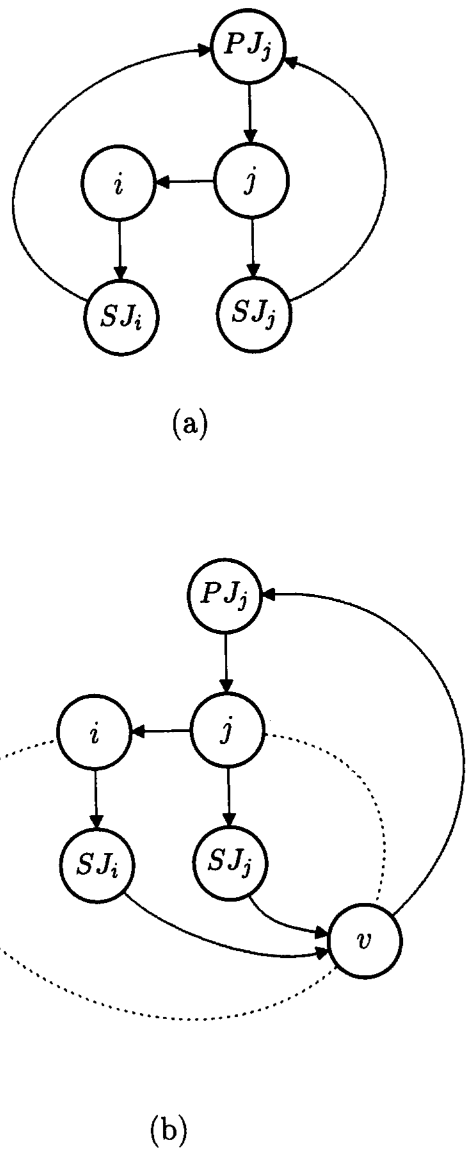


Fig. 4. Examples of the paths and circuits in $G_{s'}$.

$G_{s'}$ is no less than $d_1(\pi')/d_2(\pi')$. Therefore, we establish the following theorem.

Theorem 3. *For $G_{s'}$ that is formed from G_s by swap(i,j), we have*

$$\mu_{s'}^* \geqslant \mu_1(i,j) \equiv \max_{p \in P, q \in Q} \frac{d_1(\psi_{pq}) + d_1(\psi_{qp})}{d_2(\psi_{pq}) + d_2(\psi_{qp})}. \tag{3}$$

μ_s^*-longest path computation from $q \in Q$ to every other nodes in G_s can be done by the algorithm of Bellman [28] in $O(|N||A \cup E \cup R|)$ steps. The computation of $\mu_1(i,j)$ requires $|Q|(\leqslant 3)$ runs of the longest path algorithm. Thus, the complexity of computation of $\mu_1(i,j)$ is $O(|N||A \cup E \cup R|)$, and evaluation of all candidate moves in $C(s)$ requires $O((|N| - |M|)|N||A \cup E \cup R|)$ steps. This approximate evaluation method is faster only by order of $O(\log |N|)$ than the exact evaluation method that requires $O((|N| - |M|)|N||A \cup E \cup R|\log |N|)$ steps. Further, when $\log |N| \leqslant 3$ the approximate method takes more time. Thus a faster approximate evaluation method is needed.

By adapting the first method, we develop a faster approximate evaluation method. We take a critical circuit π^* in G_s that contains arc (i,j) to be swapped and let v be a node of π^*. Let ψ_{qvp}

be the μ_s^*-longest path in G_s from q to p through v, where $q \in Q, p \in P$ (see Fig. 4(b)). Then, we have a new lower bound on $\mu_{s'}^*$, $\mu_1'(i,j)$, as follows.

$$\mu_1'(i,j) \equiv \max_{p \in P, q \in Q, v \in \pi^*} \frac{d_1(\psi_{pq}) + d_1(\psi_{qvp})}{d_2(\psi_{pq}) + d_2(\psi_{qvp})}. \tag{4}$$

Clearly, $\mu_{s'}^* \geqslant \mu_1(i,j) \geqslant \mu_1'(i,j)$. We note $\mu_1'(i,j)$ restricts the set of circuits to be evaluated to those that pass the critical circuit π^*. Nonetheless, $\mu_1'(i,j)$ is expected to be close to $\mu_1(i,j)$. It is because either PM_i or PJ_i and either SM_j or SJ_j must be critical nodes in π^*, and thus ψ_{qp} for μ_1 is likely to contain some critical nodes in π^*. Furthermore, the critical circuits in the modified graph $G_{s'}$ are expected not to be greatly different from those in G_s and a critical circuit in $G_{s'}$ would be the one that is more or less modified from that in G_s. The restriction is utilized to develop a new measure that can be efficiently computed.

By modifying $\mu_1'(i,j)$, we propose a new measure

$$\mu_2(i,j) \equiv \frac{d_1(\pi')}{d_2(\pi')}, \tag{5}$$

where $L_{\mu_s^*}(\pi') = \max_{p \in P, q \in Q}\{L_{\mu_s^*}(\psi_{pq}) + L_{\mu_s^*}(\psi_{qvp})\}$ for a fixed $v \in \pi^*$, and ψ_{qvp} is allowed to pass i or j. π' is the circuit that maximizes the μ_s^*-length among all the circuits that should pass at least one of i and j and at least one of critical nodes in π^* (including i and j), and may pass arc (i,j). $\mu_2(i,j)$ is its circuit ratio. We justify the new measure μ_2.

First, compared with $\mu_1'(i,j)$, $\mu_2(i,j)$ relaxes the constraint that ψ_{qvp} should not pass any of i and j. The relaxation does not critically harm the accuracy of estimating the critical circuit ratio $\mu_{s'}^*$ because ψ_{qvp} rarely passes the arc (i,j) to be swapped. For example, suppose that the path ψ_{SJ_j,v,PM_i} passes (i,j). Then, the path should contain the path segment (PJ_i, i, j, SM_j) or (PJ_i, i, j, SJ_j). Since the μ_s^*-length of (SJ_i, v, PJ_i, i, j) is negative, the μ_s^*-length of ψ_{SJ_j,v,PM_i} is less than that of ψ_{PM_j,v,PM_i} or ψ_{PJ_j,v,PM_i}. Thus, ψ_{SJ_j,v,PM_i} is dominated by other paths during evaluation. Unfortunately, such an argument is not always supported for all cases. However, due to this relaxation, we can greatly reduce the computational steps. The following theorem states that when we compute $\mu_2(i,j)$, we need to consider only one node, v, in the chosen critical circuit in G_s.

Theorem 4. *Let ψ_{qvp} be a μ_s^*-longest path from q through v to p in G_s. Then, $L_{\mu_s^*}(\psi_{qvp})$ is the same for every node v in a critical circuit.*

Proof. Let π^* be a critical in G_s. Suppose that there are two critical nodes, $v, w \in \pi^*, v \neq w$, such that $L_{\mu_s^*}(\psi_{qvp}) > L_{\mu_s^*}(\psi_{qwp})$. There always exist two paths ψ_{vw} and ψ_{wv} along π^*, in which the sum of the μ_s^*-lengths is 0 by Lemma 3. By definition,

$$\begin{aligned} L_{\mu_s^*}(\psi_{qwp}) &\geqslant L_{\mu_s^*}(\psi_{qv}) + L_{\mu_s^*}(\psi_{vw}) + L_{\mu_s^*}(\psi_{wv}) + L_{\mu_s^*}(\psi_{vp}) \\ &= L_{\mu_s^*}(\psi_{qv}) + L_{\mu_s^*}(\psi_{vp}) \\ &= L_{\mu_s^*}(\psi_{qvp}). \end{aligned}$$

A contradiction.

Second, $\mu_2(i,j)$ finds μ_s^*-longest circuit (π') instead of the maximum ratio circuit. To explain why $\mu_2(i,j)$ measure can approximate the critical circuit ratio of $G_{s'}$, we assume that π' does not contain arc (i,j) through this paragraph. It is because the relaxation on ψ_{qvp} does not degrade the accuracy much. If $L_{\mu_s^*}(\pi')$ has a non-negative value, then $\mu_{s'}^* \geqslant \mu_s^*$. In this case, we cannot expect any improvement on the cycle time measure. In contrast if $L_{\mu_s^*}(\pi') < 0$, then $\mu_{s'}^* \leqslant \mu_s^*$. Since the circuit ratio of π' is also a lower bound on $\mu_{s'}^*$, it is seen that the one among all candidate moves that minimizes $\mu_2(i,j)$ has more potential of improvement on the cycle time measure. Further, if $\mu_s^* = \mu_{s'}^*$, maximizing $L_{\mu_s^*}(\psi_{pq}) + L_{\mu_s^*}(\psi_{qvp})$ is equivalent to maximizing the circuit ratio $(d_1(\psi_{pq}) + d_1(\psi_{qvp}))/((d_2(\psi_{pq}) + d_2(\psi_{qvp}))$ due to Lemma 3 and $\mu_{s'}^*$ is expected not to deviate much μ_s^* in a single move.

The computational procedure of $\mu_2(i,j)$ is as follows. We select an arbitrary node v from a critical circuit in G_s. To compute $\mu_2(i,j)$ for each candidate arc $(i,j) \in C(s)$, we need to compute only ψ_{qv} and ψ_{vp} for all $p \in P$ and $q \in Q$. For ψ_{vp}, μ_s^*-longest paths from v to every node in G_s are computed by Bellman's algorithm. For ψ_{qv}, we can also apply Bellman's algorithm with slight modifications. Therefore, only two runs of the longest path algorithm are required to estimate the cycle time of all candidate moves in $C(s)$. The second approximate evaluation method using μ_2 measure takes only $O(|N||A \cup E \cup R|)$ steps to evaluate all possible moves. The computation is much faster by order of $O((|N| - |M|) \log |N|)$ than the exact evaluation method. Computational performances of using $\mu_1(i,j)$ and $\mu_2(i,j)$ are discussed in Section 4.

3.3. The search strategies

The search starts with an initial feasible solution. In order to obtain a good starting solution, we examine the active and non-delay forms of priority dispatching schedules that have been developed for CJSSP (see Baker [29]). It is because there is no priority dispatching rule that is developed for PJSSP. The selected priority rules are shortest processing time, longest processing time, most work remaining, least work remaining, and most operations remaining. We tested these rules for hundreds of PJSSP instances. Like in the CJSSP, there was no single priority rule that dominated the others. Hence, we apply all the selected dispatching rules and choose the schedule with the minimum cycle time as the starting solution. Such a policy is used by Barnes and Chambers [22] for the CJSSP.

We take a dynamic strategy that changes the length of the tabu list periodically (see Glover [26]). By extensive experiments, we found that the rules of Dell'Amico and Trubian [21] show a good performance with our problem too. The selected rules are as follows. If the cycle time of the current solution is less than the best value found, then the length of tabu list, $\theta = 1$. If the search is in an improving phase, that is, the cycle time of the current solution is less than the value at the previous iteration, set $\theta = \max\{\theta - 1, \alpha\}$, and otherwise set $\theta = \min\{\theta + 1, \beta\}$, where the thresholds, $\alpha \in [a, b]$ and $\beta \in [A, B]$, are chosen randomly at every *t_resize* iteration. An explicit cycle detection procedure is invoked at every *t_cycle* iterations. This procedure traces the history of the last Θ moves, where Θ is an integer greater than θ, and checks the occurrence of a cycling phenomenon.

For these purposes, the tabu list is implemented as a doubly-linked circular list with the size of Θ, in which the index of the reversed operations and the cycle time are memorized. The current top position of the list is shifted by one when a new tabu element is stored. The recent previous θ elements are used to check the tabu state, and the remaining elements are used for cycle detection. When a cycle is detected, the top position is shifted to the beginning of the cycle.

A move swap(i, j) is penalized additively by the amount of P_1 (a sufficiently large number) if (j, i) has been reversed during the recent θ previous iterations. As an *aspiration criterion*, another usual strategy in tabu search that overrides the tabu status in particular situations, we cancel the penalization if the estimated value is less than the value of the best solution found before the current iteration.

For larger problems, the diversification mechanism that drives the search into a new region would be useful. To do this, we use a frequency-based long term memory that counts the number of swap moves for each unordered pair of operations. When certain conditions are met, the moves are penalized in proportion to their frequency, i.e., $P_2 \cdot f(i;j)$, where P_2 is a constant and $f(i;j)$ is the number of moves in which (i,j) or (j,i) is swapped. The penalization applies to the following case; the more recent improvement to the best solution found, did not occur during the last k iterations, and $t_1 \leqslant k \leqslant t_1 + t_2$, where t_1 and t_2 are the parameters that determines the diversification interval. Finally, we also use the following *restarting* strategy. If the algorithm did not improve the best solution during the last *t_restart* iteration, then it sets the current solution to be the best solution. The algorithm terminates if $k \geqslant$ *t_max*.

The following is a brief description of the overall tabu search procedure adapted for PJSSP, the algorithm TS2 using the proposed approximate evaluation method μ_2.

```
Algorithm TS2
begin
  Initialize tabu list $T$ and frequency memory $f$;
  Choose the best dispatching solution $s$ and construct PCG $G_s$;
  $k := 0$; $C_t^* := \infty$;
```

Table 1. Accuracy of evaluation methods

					TS0		TS1		TS2	
Problem	$\|M\|$	$\|J\|$	LB	Z_{init}	Z_{best}	Z_{av}	Z_{best}	Z_{avg}	Z_{best}	Z_{avg}
T1-P01	5	5	377	437.0	*377.0	–	*377.0	–	*377.0	–
T1-P02	6	6	407	650.0	493.0	–	493.0	–	493.0	–
T1-P03	7	7	472	746.0	553.0	–	553.0	–	553.0	–
T1-P04	8	8	514	769.0	655.0	660.2	646.0	660.6	646.0	659.2
T1-P05	9	9	500	736.0	549.0	549.6	549.0	549.0	549.0	549.1
T1-P06	10	10	706	957.0	723.0	724.6	723.0	728.7	723.0	725.9
T1-P07	11	11	722	1145.0	794.5	804.8	794.5	800.7	794.5	800.5
T1-P08	12	12	720	1019.0	851.0	851.4	851.0	852.2	851.0	851.9
T1-P09	13	13	827	1328.0	908.0	930.4	919.0	924.0	918.0	926.2
T1-P10	14	14	842	1531.0	1008.0	1021.2	1017.0	1027.5	1009.0	1022.8
T1-P11	15	15	1004	1416.0	1094.0	1103.5	1104.0	1107.9	1091.0	1103.0

```
while $k \leqslant t\_max$ do
begin
    Find a critical circuit $\pi^*$ and the critical circuit ratio $\mu_s^*$ of $G_s$;
    if $\mu_s^* < C_t^*$ or $k > t\_restart$ then $k:=0$; $C_t^* = \mu_s^*$; else $k:=k+1$;
    Find the set of candidates $C(s)$ from $\pi^*$;
    if $C(s) = \varnothing$ then $s$ is optimal, stop;
    Select a critical node $v$ from $\pi^*$;
    Calculate the $\mu_s^*$-longest length from $v \to i$ and $i \to v$ for each $i \in N$;
    for each $(i,j) \in C(s)$ do
    begin
        $\mu(i,j) := \mu_2(i,j)$;
        if $(j,i) \in T$ then $\mu(i,j) := \mu(i,j) + P_1$;
        if $t_1 \leqslant k \leqslant t_1 + t_2$ then $\mu(i,j) := \mu(i,j) + P_2 \cdot f(i;j)$;
    end
    Select $(i,j)$ such that $\mu(i,j) = \min_{(p,q) \in C(s)} \mu(p,q)$;
    $G_s := swap(i,j)$;
    Insert $(i,j)$ into $T$; $f(i;j) := f(i;j) + 1$;
end
end
```

4. IMPLEMENTATION AND COMPUTATIONAL RESULTS

We coded the proposed algorithms in C and ran them on a SUN4/20 workstation. For the parameters of the tabu list, we take $a = 2, b = a + \eta, A = \alpha + 6, B = A + \eta$, and $t\_resize = 10$, where $\eta \equiv \lfloor(|J| + |M|)/3\rfloor$. The size of the circular list, Θ, is set to 4 times of the largest possible value of the length of the tabu list. Thus the algorithm can check the occurrence of the cycle of which the length is smaller than $\Theta/2 = 2(8 + 2\eta)$. The parameter t_cycle is set to 10. For the frequency memory parameters, we use $t_1 = 500, t_2 = 100$, and P_2 is set to 1000. The number of restarting is limited to 1, and $t\_restart$ and t_max is set to 2000 and 3000 respectively.

We tested the algorithms on two sets of problem instances. The first set consists of randomly generated problems using Taillard's algorithm [30]. There are exactly $|M|$ operations per job, one operation per machine. The processing times are chosen at random, uniformly distributed between 1 and 99 (integers). The processing order of a job is independent of the other jobs. 28 instances are tested; eleven square ($|M| = |J|$) problems ($T1$), seven small problems with known

Table 2. TS2 for small problems with known optimal

Problem	$\|M\|$	$\|J\|$	Z_{opt}	Z_{init}	Z_{best}	Z_{worst}	Z_{avg}	T_{max}	T_{avg}
T2-P01	6	6	409.0	587.0	*409.0	–	–	14.5	14.2
T2-P02	6	6	367.0	558.0	*367.0	–	–	14.2	14.1
T2-P03	6	6	465.0	666.0	*465.0	–	–	14.6	14.1
T2-P04	6	6	406.5	625.0	*406.5	–	–	24.8	18.1
T2-P05	6	6	394.0	529.0	*394.0	–	–	13.9	13.5
T2-P06	6	7	474.0	636.0	*474.0	–	–	16.3	16.1
T2-P07	7	7	465.0	592.0	*465.0	–	–	28.0	23.1

Table 3. TS2 for the benchmark problems

Problem	$\lvert M\rvert$	$\lvert J\rvert$	$Z_{c_{max}}$	LB	Z_{init}	Z_{best}	Z_{worst}	Z_{avg}	T_{max}	T_{avg}
T3-P01	6	6	55	43	46.0	*46.0	–	–	14.4	13.1
T3-P02	10	10	930	631	804.0	*631.0	–	–	68.7	61.1
T3-P03	5	20	1179	1119	1119.0	*1119.0	–	–	59.8	59.
T3-P04	10	10	1234	868	1252.0	1065.0	1076.0	1071.6	95.0	75.1
T3-P05	10	10	943	688	1048.0	811.0	840.0	816.8	76.1	67.2
T3-P06	5	10	666	666	680.0	*666.0	–	–	22.4	20.4
T3-P07	5	10	655	635	635.0	*635.0	–	–	23.4	21.8
T3-P08	5	10	597	588	616.0	*588.0	–	–	21.6	21.5
T3-P09	5	10	590	537	815.0	553.0	556.0	554.2	38.7	32.1
T3-P10	5	10	593	593	593.0	*593.0	–	–	20.3	20.0
T3-P11	5	15	926	926	926.0	*926.0	–	–	40.2	38.7
T3-P12	5	15	890	869	869.0	*869.0	–	–	43.9	43.2
T3-P13	5	15	863	863	863.0	*897.0	–	–	37.3	36.2
T3-P14	5	15	951	951	951.0	*951.0	–	–	37.8	36.4
T3-P15	5	15	958	958	958.0	*958.0	–	–	42.4	40.1
T3-P16	5	20	1222	1222	1222.0	*1222.0	–	–	63.6	61.9
T3-P17	5	20	1039	1039	1184.0	*1039.0	–	–	63.4	62.7
T3-P18	5	20	1150	1150	1150.0	*1150.0	–	–	61.3	57.0
T3-P19	5	20	1292	1292	1292.0	*1292.0	–	–	60.1	58.5
T3-P20	5	20	1207	1207	1207.0	*1207.0	–	–	67.5	64.0
T3-P21	10	10	945	660	996.0	777.0	780.0	778.6	94.5	75.6
T3-P22	10	10	784	683	878.0	700.5	720.0	710.7	138.5	83.1
T3-P23	10	10	848	623	894.0	768.0	787.0	775.5	102.2	89.6
T3-P24	10	10	842	685	965.0	783.0	796.0	790.0	84.1	69.4
T3-P25	10	10	902	744	1043.0	769.0	–	–	82.7	70.4
T3-P26	10	15	1059	935	1241.0	949.0	958.0	953.9	210.0	152.9
T3-P27	10	15	927	830	1065.0	859.0	872.0	865.3	167.9	136.6
T3-P28	10	15	1032	1032	1239.0	*1032.0	–	–	76.9	75.8
T3-P29	10	15	935	857	1172.0	906.0	930.0	919.7	191.7	137.3
T3-P30	10	15	977	864	1185.0	899.0	909.0	902.2	221.7	173.3
T3-P31	10	20	1218	1218	1609.0	*1281.0	–	–	133.8	129.7
T3-P32	10	20	1270	1188	1658.0	*1188.0	1194.0	1189.4	367.2	300.9
T3-P33	10	20	1276	1216	1587.0	*1216.0	–	–	270.9	227.5
T3-P34	10	20	1202	1105	1477.0	*1105.0	–	–	218.6	158.2
T3-P35	10	20	1355	1355	1546.0	*1355.0	–	–	127.5	120.0
T3-P36	10	30	1784	1784	2074.0	*1784.0	1803.0	1787.8	337.7	244.9
T3-P37	10	30	1850	1850	2142.0	*1850.0	–	–	220.4	209.6
T3-P38	10	30	1719	1719	2043.0	*1719.0	–	–	211.9	198.9
T3-P39	10	30	1721	1721	2062.0	*1721.0	–	–	243.4	237.3
T3-P40	10	30	1888	1888	2201.0	*1888.0	–	–	279.7	274.1
T3-P41	15	15	1268	1028	1544.0	1153.5	1168.0	1160.7	317.0	265.4
T3-P42	15	15	1425	980	1843.0	1278.5	1329.0	1294.0	285.3	254.3
T3-P43	15	15	1232	876	1491.0	1096.0	1148.0	1123.4	331.9	300.8
T3-P44	15	15	1233	1012	1493.0	1141.0	1166.0	1152.3	446.2	315.1
T3-P45	15	15	1238	1027	1537.0	1129.0	1181.0	1159.6	437.4	312.6

optimal solutions (T2), and ten rectangular ($|M|>|J|$) problems (T4). The second set are benchmark problems for CJSSP. Forty five instances are tested, three from Muth and Thompson [31] and 42 problems from Adams *et al.* [16] (T3).

Since our algorithm periodically chooses the length of the tabu list at random, it is necessary to perform multiple runs on the same problem in order to get meaningful results. We solved each problem 5 times with the same initial solution but different random seeds.

The computational results are given in Tables 1–4. The tables use the following symbols:

- LB: The lower bound of the cycle times of all possible permutation schedules, e.g. the maximum workload of machines, $LB=\max_{m\in M}\Sigma_{j\in N_m}p_j$.
- Z_{init}: the value of the initial solution (the best dispatching schedule).
- Z_{best}: The value of the best solution found by the tabu search of 5 runs. If this value is marked with the asterisk, the solution is proved to be optimal or is equal to LB.

Table 4. TS2 for $|M|>|J|$ cases

Problem	$\lvert M\rvert$	$\lvert J\rvert$	$Z_{c_{max}}$	LB	Z_{init}	Z_{best}	Z_{worst}	Z_{avg}	T_{max}	T_{avg}
T4-P01	10	5	662	336	632.0	574.0	–	–	21.0	20.7
T4-P02	10	5	745	358	622.0	602.0	–	–	25.1	22.8
T4-P03	10	5	657	414	589.0	580.0	–	–	25.5	23.4
T4-P04	10	5	644	304	599.0	576.0	–	–	22.1	22.0
T4-P05	10	5	644	356	638.0	541.0	–	–	19.8	19.1
T4-P06	20	5	1194	307	1167.0	952.0	–	–	77.0	74.3
T4-P07	20	5	1304	410	1291.0	1148.0	–	–	75.3	71.9
T4-P08	20	5	1231	380	1101.0	1054.0	–	–	91.2	80.4
T4-P09	20	5	1197	376	1198.0	1060.0	–	–	67.5	66.5
T4-P10	20	5	1200	415	1118.0	941.0	–	–	69.2	67.5

- Z_{worst}: The value of the worst solution found by the tabu search of 5 runs, or nothing if the values are the same for all runs.
- Z_{avg}: The average solution value over 5 runs of tabu search or nothing if the values are the same for all runs.
- T_{max}: The maximum computing time over 5 runs (CPU time in s).
- T_{avg}: The average computing time over 5 runs (CPU time in s).

We first verify the effectiveness of the proposed evaluation methods. The problems of which the sizes are varied from 5 × 5 to 15 × 15 are generated using the algorithm of Taillard [30], and tested at the same condition. Table 1 compares the accuracy of the final solutions obtained by the exact evaluation method using Karp and Orlin's algorithm [27] (TS0), the approximate evaluation method using μ_1 (intermediate method to derive μ_2) (TS1), and finally proposed approximate evaluation method using μ_2 (TS2). As shown in Table 1, both of the methods using μ_1 and μ_2 find the solutions very close to that of exact evaluation method. In fact, we found that relative approximation errors of the measures μ_1 and μ_2 are no more than 5% and 10%, respectively. Figure 5 shows average CPU times over 5 runs of each problem. From this figure, we can see that TS2 is always faster than the others, for instance, TS2 takes 250 s for the 15 × 15 problem while TS0 takes 2250 s (9 times of TS2) and TS1 3750 s (about 13 times of TS2). We note that the performance of TS1 is no better than TS0 despite of its reduced worst case complexity (it is because $\log|N| \ll 3$.) as mentioned earlier. Further, the execution time for TS2 increases rather slowly as the dimension of the problem increases. Consequently, we conclude that the approximate evaluation method using μ_2 is effective and fast. We thus use TS2 for the remaining experiments.

As a way of demonstrating the effectiveness of our algorithm, we would compare our algorithm with the optimal solutions (Z_{opt}). To do this, we use a mixed integer programming model of Lee [23] and a general purpose package, CPLEX. However, we could not get the optimal solutions for the problems larger than 6 × 6 within 10 h except for T2-P06 and T2-P07 given in Table 2. For all problems in Table 2, TS2 always found optimal solutions within 28 s.

We tested our algorithm for the benchmark problems that has been widely used for the CJSSP algorithms. The problems in Table 3 and the values of the optimal (or near optimal) makespans ($Z_{C_{max}}$) can be found in Table 1 and 2 of Brucker *et al.* [17]. We note that the optimal makespan is an upper bound on the minimum cycle time. For the 10 × 10 benchmark problem (T3-P02) of Muth and Thompson [31], all 5 runs of TS2 found the optimal solution of which the cycle time

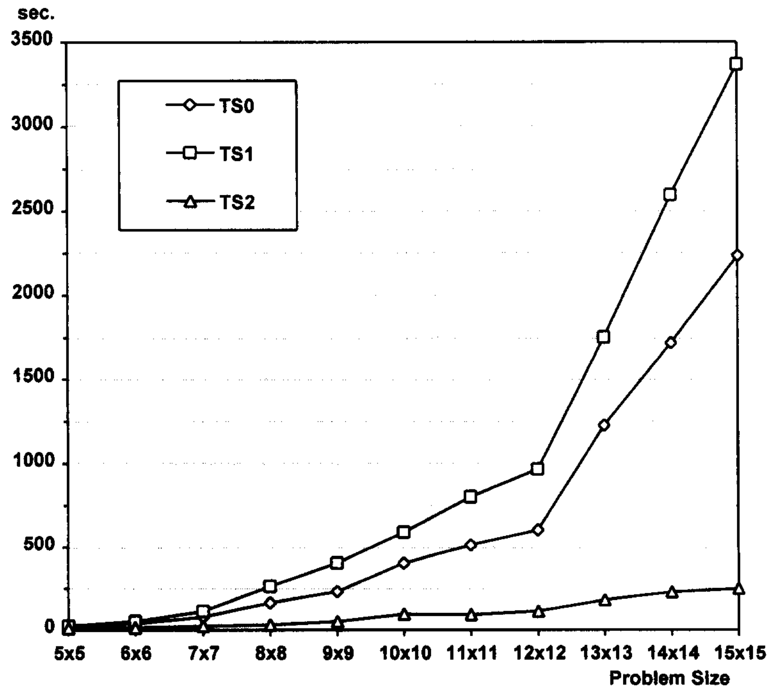


Fig. 5. Computational efficiency of evaluation methods.

is exactly the same as $LB = 631$, with 70 s of CPU time for each run. For the 10×10 and 15×15 benchmark problems of Adams *et al.* [16], *i.e.*, for the square problems, we could not prove the optimality of the best solutions found. We also have good and robust results for the problems with large ratios of the number of jobs to the number of machines (5×10, 5×15, 5×20, 10×30). For most of these cases, the cycle time is close to the optimal makespan. It is because the problems have very "fat" schedules.

Table 4 shows the TS2 results for the problems where the number of machines is more than the number of jobs, which have "slim" schedules. We tested five 10×5 problems and five 20×5 problems that are generated using the algorithm of Taillard [30]. As shown in Table 4, TS2 also shows very robust results although the optimality of the final solutons can not be verified because the *LB*'s are far from them. Nonetheless, it can be seen that the difference between the cycle time and the makespan of a single MPS periodic schedule becomes larger as the ratio of the number of machines to the number of jobs increases.

5. CONCLUDING REMARKS

We have developed an effective tabu search procedure for PJSSP, a new class of job shop scheduling problem with the cycle time measure. By using critical circuits of the associated graph, we have characterized the neighborhood structure. We have proposed an approximation to the cycle time measure to evaluate the neighborhood solutions. There are still points of improvements on our algorithm, particulary the use of more sophisticated moves. However, to use such moves, we should characterize the corresponding neighborhood structure and also develop the cycle time evaluation (or approximation) method accordingly. Further algorithms like shifting bottleneck procedures are yet to be developed.

REFERENCES

1. K. L. Hitz. Scheduling of Flexible Flow Shops-II. Technical Report LIDS-R-1049, *Laboratory for Information and Decision Systems*. M.I.T., Cambridge, MA (1980).
2. H. Kamoun and C. Sriskandarajah. The Complexity of Scheduling Jobs in Repetitive Manufacturing Systems. *Eur. J. Oper. Res.* **70,** 350–364 (1993).
3. J. K. Chaar and E. S. Davidson. Cyclic Job Shop Scheduling using Reservation Tables. *Proc. 1990 IEEE Int.Conf. Robotics and Automation*, Cincinnati, OH, 2128–2135 (1990).
4. N. G. Hall, H. Kamoun and C. Sriskandarajah. *Scheduling in Robotics Cells: Classification, Two and Three Machine Cells.* Working Paper, College of Business, The Ohio State University (1994).
5. S. B. Gershwin, R. Akella and Y. F. Choong. Short-Term Production Scheduling of an Automated Manufacturing Facility. *IBM J. Res. Dev.* **29,** 392–400 (1985).
6. T. E. Lee and M. E. Posner. Performance Measures and Schedule Patterns in Periodic Job Shops. *Opers Res.* (in press)
7. R. H. Ahmadi and H. Wurgaft. Design for Synchronized Flow Manufacturing. *Mgmt Sci.* **40,** 1469–1483 (1994).
8. C. F. G. Bispo and J. J. Sentieiro. An Extended Horizon Scheduling Algorithm for the Job-shop Problem. *Proc. 1988 Int. Conf. Comp. Integrated Manufacturing*, Troy, NY, 249–252 (1988).
9. S. C. Graves, H. C. Meal, D. Stefek and A. H. Zeghmi. Scheduling of Re-entrant Flow Shops. *J. Opers Mgmt* **3,** 197–207 (1983).
10. C. Hanen. Study of a NP-hard Cyclic Scheduling Problem: The Periodic Recurrent Job Shop. *Second Int. Workshop on Project Mgmt and Sched.*, Compiègne, France (1990).
11. H. Matsuo. *Cyclic Sequencing Problems in the Two-Machine Permutation Flow shop: Complexity, Worst Case and Average Case Analysis.* Manuscript. Graduate School of Business, University of Texas, Austin, Texas (1988).
12. S. T. McCormick, M. L. Pinedo, S. Shenker and B. Wolf. Sequencing in an Assembly Line with Blocking to Minimize Cycle Time. *Opers Res.* **37,** 925–935 (1989).
13. R. Roundy. *Cyclic Schedules for Job-Shops with Identical Jobs. Technical Report No. 766.* School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY (1988).
14. R. J. Wittrock. Scheduling algorithms for flexible flow lines. *IBM J. Res. Dev.* **29,** 401–412 (1985).
15. N. G. Hall, T. E. Lee and M. E. Posner. Complexity of periodic shop scheduling problems. (Submitted to *Opers Res. Letts.*).
16. J. Adams, E. Balas and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Mgmt Sci.* **34,** 391–401 (1988).
17. P. Brucker, B. Jurisch and B. Sievers. A branch and bound algorithm for the job shop scheduling. *Discrete Applied Mathematics* **49,** 107–127 (1994).
18. J. Carlier and E. Pinson. Adjustment of heads and tails for the job-shop problem. *Eur. J. Oper. Res.* **78,** 146–161 (1994).
19. P. J. M. Van Laarhoven, E. H. L. Aarts and J. K. Lenstra. Job shop scheduling by simulated annealing. *Opers Res.* **40,** 113–125 (1992).
20. E. Taillard. *Parallel Taboo Search Techniques for the Job Shop scheduling Problem.* Internal Report ORWP 89/11, Départment de Mathématique, Ecole Polytechnique Féderale de Lausanne, Lausanne (1989).
21. M. Dell'Amico and M. Trubian. Applying tabu search to job-shop scheduling problem. *Annals Opers Res.* **41,** 231–252 (1993).
22. J. W. Barnes and J. B. Chambers. Solving the job shop scheduling problem with Tabu search. *IIE Trans.* **27,** 257–263 (1995).

23. T. E. Lee. *Periodic Job Shop Scheduling*. Ph.D. Thesis, The Ohio State University (1991).
24. E. Balas. Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. *Opers Res.* **17**, 941–957 (1969),
25. F. Glover. Tabu Search—Part I. *ORSA J. on Comp.* **1**, 190–206 (1989).
26. F. Glover. Tabu Search—Part II. *ORSA J. on Comp.* **2**, 4–32 (1990).
27. R. M. Karp and J. B. Orlin. Parametric shortest path algorithms with an application to cyclic staffing. *Discrete Applied Mathematics* **3**, 37–45 (1981).
28. R. Bellman. On a routing problem. *Quarterly of Applied Mathematics* **16**, 87–90 (1958).
29. K. R. Baker. *Introduction to Sequencing and Scheduling*. Wiley, New York (1974).
30. E. Taillard. Benchmarks for basic scheduling problems. *Eur. J. Oper. Res.* **64**, 278–285 (1993).
31. J. F. Muth and G. L. Thompson. *Industrial Scheduling*. Prentice-Hall, Englewood Cliffs, NJ (1969).