Theoretical Computer Science 434 (2012) 45–52

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Neighborhood search procedures for single machine tardiness scheduling with sequence-dependent setups

Ching-Jong Liao *, Hsin-Hui Tsou, Kuo-Ling Huang

Department of Industrial Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan

ARTICLE INFO

Article history:
Received 19 March 2011
Received in revised form 24 December 2011
Accepted 24 January 2012
Communicated by D.-Z. Du

Keywords:
Single machine
Neighborhood
Sequence-dependent setups
Weighted tardiness

ABSTRACT

In this paper we address the $1/s_{ij}/\sum w_jT_j$ problem, for which we improve the time complexities of searching the interchange, insertion and twist neighborhoods from $O(n^3)$ to $O(n^2 \log n)$. Further, we improve the time complexity of searching the insertion and twist neighborhoods in which a candidate job is selected from among k jobs nearest to the selected job from $O(n^2k)$ to $O(nk \log k + n^2)$.

1. Introduction

The $1/s_{ij}/\sum w_jT_j$ problem can be described as follows. We have n jobs that are all available for processing at time zero on a continuously available single machine. The machine can process only one job at a time. Associated with each job j are the processing time p_j, due date d_j, and weight w_j. In addition, there is a setup time s_{ij} incurred when job j follows job i immediately in the processing sequence. Let $\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ be a sequence of jobs, where k is the job index of job $\pi(k)$ in π. The completion time of job $\pi(k)$ is defined as $C_{\pi(k)} = \sum_{l=1}^{k} (s_{\pi(l-1),\pi(l)} + p_{\pi(l)})$, where $\pi(0) = 0$ is a dummy job. Let $C_{\pi(k)} - d_{\pi(k)}$ be the lateness of job $\pi(k)$, and $T_{\pi(k)} = (C_{\pi(k)} - d_{\pi(k)})^+$ be the tardiness of job $\pi(k)$, where $(x)^+ = max(x, 0)$. Let $WT(\pi, k) = \sum_{l=1}^{k} w_{\pi(l)}T_{\pi(l)}$ denote the partial weighted sum of tardiness of the first k jobs in π. The goal is to find a sequence π that minimizes the objective function $f(\pi) = WT(\pi, n)$.

The study of this problem is well justified in practice because the neighborhood search is crucial and in general is the most time-consuming step in the design of improvement algorithms. Therefore, it is worthwhile to improve the time complexity of searching the most commonly used neighborhoods—interchange, insertion, and twist neighborhoods.

2. Literature review

Lawler [10] showed that the $1//\sum w_jT_j$ problem is strongly NP-hard. Since the incorporation of setup times complicates the problem, the $1/s_{ij}/\sum w_jT_j$ problem is also strongly NP-hard. In addition, the unweighted version $1/s_{ij}/\sum T_j$ is strongly NP-hard since $1/s_{ij}/C_{max}$ is strongly NP-hard (Pinedo [13]). Therefore, the heuristic approach is well justified for $1/s_{ij}/\sum w_jT_j$ and $1/s_{ij}/\sum T_j$ to obtain near-optimal schedules within a reasonable computation time.

Scheduling heuristic algorithms can be broadly classified into two categories: the constructive and the improvement approaches. In the literature the best constructive algorithm for the $1/s_{ij}/\sum w_jT_j$ problem is Apparent Tardiness Cost with

* Correspondence to: Department of Industrial Management, National Taiwan University of Science and Technology, 43 Keelung Road, Section 4, Taipei 106, Taiwan. Tel.: +886 2 2737 6337.
E-mail address: cjliao@mail.ntust.edu.tw (C.-J. Liao).

0304-3975/$ – see front matter
doi:10.1016/j.tcs.2012.01.043

Setups algorithm (ATCS), which was proposed by Lee et al. [11]. Like other constructive heuristics, ATCS can derive a feasible solution quickly but the solution quality is usually unsatisfactory, especially for large-sized problems. On the other hand, the improvement heuristic can produce better solutions but with longer computational times. Among the improvement algorithms for the $1/s_{ij}/\sum w_jT_j$ problem, Cicirello [2] developed four different improvement algorithms, including LDS (limited discrepancy search), HBSS (heuristic-biased stochastic sampling), VBSS (value-biased stochastic sampling), and VBSS-HC (hill-climbing using VBSS), to obtain solutions for a set of 120 benchmark problem instances each with 60 jobs. Recently, an ant colony optimization algorithm has been used by Liao and Juan [12] to update most of these instances.

For the unweighted problem $1/s_{ij}/\sum T_j$, several metaheuristic algorithms were proposed in the literature (see e.g. [1,5–7,11,12,14,15]). Some of their embedded local search algorithms were based on the insertion or interchange moves. Among the authors who studied this problem, França et al. [5] used a new memetic algorithm with a hybrid population approach and developed some reduction rules to improve the average computation time for searching the interchange and insertion neighborhoods. Gagné et al. [6,7] developed an ant colony optimization algorithm and a tabu-vns (variable neighborhood search) hybrid algorithm for the problem and yielded competitive computation results. Rubin and Ragatz [14] applied the genetic algorithm with their developed crossover operator while Tan and Narasimhan [15] proposed a simulated annealing approach for the problem and tested it against a multiple start technique.

For the common due date problem $1/d_j = d/\sum w_jT_j$, Kellerer and Strusevich [8] developed a fully polynomial-time approximation scheme (FPTAS) for the problem. The FPTAS is obtained by converting an especially designed pseudopolynomial dynamic programming algorithm.

For the $1//\sum w_jT_j$ problem, Kolliopoulos and Steiner [9] designed a pseudopolynomial algorithm and transformed to an FPTAS for the case where the weights are polynomially bounded. Ergun [3] used a special balanced tree data structure to improve the search of swap neighborhood from $O(n^3)$ bound to $O(n^2 \log n)$. Further, Ergun and Orlin [4] presented an efficient algorithm for searching the insertion, interchange, and twist neighborhoods, improving the time complexity from the previous $O(n^3)$ bound to $O(n^2)$. Their main idea is to use a set of piecewise linear convex functions to evaluate the weighted sum of tardiness. In this paper we extended these ideas to develop neighborhood search algorithms for the $1/s_{ij}/\sum w_jT_j$ problem.

3. Neighborhoods

In this section we discuss the interchange, insertion, and twist neighborhoods for $1/s_{ij}/\sum w_jT_j$. We present algorithms for searching the interchange, insertion and twist neighborhoods in $O(n^2 \log n)$ time.

3.1. The interchange neighborhood

The interchange neighborhood can be obtained by interchanging two jobs in different positions. Given a sequence $\pi = [\alpha, \pi(i), \beta, \pi(j), \gamma]$ with n jobs, and a pair of indices i and j ($1 \le i < j \le n$), the interchange neighborhood of π by interchanging jobs in i and j can be expressed as

$$IC(\pi, i, j) = [\alpha, \pi(j), \beta, \pi(i), \gamma],$$

where α and γ respectively represent the partial sequences consisting of the first $i-1$ jobs and last $n-j$ jobs in π, and β is a partial sequence consisting of jobs between i and j. The size of the interchange neighborhood is $n(n-1)/2$. For any given sequence with n jobs, it takes $O(n)$ time to calculate the weighted sum of the tardiness. Thus $O(n^3)$ time is required to search the interchange neighborhood using a straightforward algorithm, i.e., interchange a pair of jobs and then calculate the weighted tardiness directly until all of the neighborhoods have been searched. In the following we first develop an algorithm for searching the interchange neighborhood in $O(n^2 \log n)$ time.

The technique used in this paper was developed by Ergun and Orlin [4]. Let *LateList* be a list of all jobs sorted in non-increasing order of lateness with respect to a given sequence π. Using a standard $O(n \log n)$ sorting algorithm such as *Mergesort*, the *LateList* can be obtained in $O(n \log n)$ time. Let $B_k(t)$ denote the weighted sum of tardiness if we schedule only jobs $\pi(k), \pi(k+1), \ldots, \pi(n)$ and in that order, and if job $\pi(k)$ starts at time t. That is,

$$B_k(t) = \begin{cases} \sum_{l=k}^{n} w_{\pi(l)}(t + C_{\pi(l)} - C_{\pi(k-1)} - s_{\pi(k-1),\pi(k)} - d_{\pi(l)})^+ & 1 \le k \le n, \\ 0 & \text{otherwise.} \end{cases}$$

For any given k, $B_k(t)$ is a piecewise linear convex function of t, and its slope changes only at $t = d_{\pi(l)} - C_{\pi(l)} + C_{\pi(k-1)} + s_{\pi(k-1),\pi(k)}$ for $k+1 \le l \le n$, at which at least one more job becomes tardy. Ergun and Orlin [4] referred to these values as *breakpoints,* denoted here by $BP_1, \ldots, BP_r$. It is clear that the value of $C_{\pi(k-1)} + s_{\pi(k-1),\pi(k)}$ is constant and thus the order of these breakpoints coincides with the order of the jobs on *LateList*. Therefore, these breakpoints can be determined, in order, in $O(n)$ time. Further, the function $B_k(t)$ can be expressed as a set of linear functions in the intervals $[BP_k, BP_{k+1}]$ for $k \in \{1, \ldots, r-1\}$. Note that it takes $O(n)$ time to determine all of the breakpoints and the sets of linear functions.

Now we show how to use this technique to calculate the weighted tardiness of $IC(\pi, i, j)$ for $1 \le i < j \le n$. To simplify, let $\pi' = IC(\pi, i, j)$, i.e., from π to π', $\pi(i)$ ($\pi(j)$) equals to $\pi'(j)$ ($\pi'(i)$) whereas the others remain invariable. We further

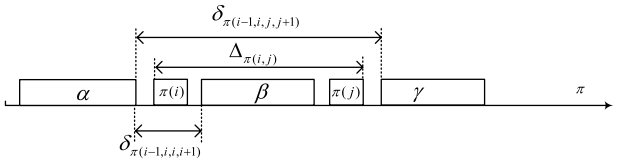

Fig. 1. A graphical representation for $\Delta_{\pi(i,j)}$, $\delta_{\pi(i-1,i,i,i+1)}$, and $\delta_{\pi(i-1,i,j,j+1)}$ in π.

define a new variable $\Delta_{\pi(i,j)}$ to represent the time span between the starting time of $\pi(i)$ and the completion time of $\pi(j)$, which is shown as follows, see Fig. 1:

$$\Delta_{\pi(i,j)} = \begin{cases} C_{\pi(j)} - C_{\pi(i)} + p_{\pi(i)} & 1 \le i \le j \le n, \\ 0 & \text{otherwise.} \end{cases}$$

To extend $\Delta_{\pi(i,j)}$, we define another symbol $\delta_{\pi(l,i,j,m)}$ as the "prior" setup from any other job $\pi(l)$ to $\pi(i)$ plus the time span $\Delta_{\pi(i,j)}$ plus the "posterior" setup from $\pi(j)$ to another job $\pi(m)$, that is,

$$\delta_{\pi(l,i,j,m)} = \begin{cases} s_{\pi(l),\pi(i)} + \Delta_{\pi(i,j)} + s_{\pi(j),\pi(m)} & i \le j, \\ s_{\pi(l),\pi(m)} & \text{otherwise.} \end{cases}$$

In the case $i = j$, $\delta_{\pi(l,i,i,m)} = s_{\pi(l),\pi(i)} + \Delta_{\pi(i,i)} + s_{\pi(i),\pi(m)} = s_{\pi(l),\pi(i)} + p_{\pi(i)} + s_{\pi(i),\pi(m)}$.

Using the above notation, the total weighted sum of tardiness of π' can be formulated as follows:

$$\begin{aligned} f(\pi') &= [WT(\pi', i-1)] + [w_{\pi'(i)}(C_{\pi'(i)} - d_{\pi'(i)})^+] + [w_{\pi'(j)}(C_{\pi'(j)} - d_{\pi'(j)})^+] \\ &\quad + [B_{i+1}(t_{i+1}^{tmpIC(\pi,i,j)}) - B_j(t_j^{tmpIC(\pi,i,j)})] + [B_{j+1}(t_{j+1}^{IC(\pi,i,j)})] \\ &= [WT(\pi, i-1)] + [w_{\pi'(i)}(t_{i+1}^{tmpIC(\pi,i,j)} - s_{\pi'(i),\pi'(i+1)} - d_{\pi'(i)})^+] \\ &\quad + [w_{\pi'(j)}(t_{j+1}^{IC(\pi,i,j)} - s_{\pi'(j),\pi'(j+1)} - d_{\pi'(j)})^+] \\ &\quad + [B_{i+1}(t_{i+1}^{tmpIC(\pi,i,j)}) - B_j(t_j^{tmpIC(\pi,i,j)})] + [B_{j+1}(t_{j+1}^{IC(\pi,i,j)})]. \end{aligned} \tag{1}$$

where

$$\begin{aligned} t_{i+1}^{tmpIC(\pi,i,j)} &= C_{\pi(i-1)} + s_{\pi'(i-1),\pi'(i)} + p_{\pi'(i)} + s_{\pi'(i),\pi'(i+1)}, \\ t_j^{tmpIC(\pi,i,j)} &= t_{i+1}^{tmpIC(\pi,i,j)} + \Delta_{\pi'(i+1,j-1)} + s_{\pi'(j-1),\pi(j)}, \\ t_{j+1}^{IC(\pi,i,j)} &= C_{\pi(i-1)} + s_{\pi'(i-1),\pi'(i)} + p_{\pi'(i)} + \delta_{\pi'(i,i+1,j-1,j)} + p_{\pi'(j)} + s_{\pi'(j),,\pi'(j+1)}. \end{aligned}$$

Notation $t_{i+1}^{tmpIC(\pi,i,j)}$ and $t_j^{tmpIC(\pi,i,j)}$ respectively represent the starting times of the $i+1$-th job and the j-th job with respect to a temporary sequence $tmpIC(\pi, i, j) = [\alpha, \pi'(i), \beta, \pi(j), \gamma] = [\pi'(1), \ldots, \pi'(i), \pi(i+1) \ldots, \pi(j-1), \pi(j), \ldots, \pi(n)]$, which consists of the first part of π' and the last part of π. Comparing to $\pi, \pi(i)$ in $tmpIC(\pi, i, j)$ has been replaced by $\pi'(i)$ whereas the others remain the same. Using $tmpIC(\pi, i, j)$, one can calculate the weighted sum of tardiness of β easily with the following way. Recall that $B_k(t)$ represents the weighted sum of tardiness of the last $n-k+1$ jobs in a given sequence. The fourth term in (1), the weighted tardiness of the last $n-i$ jobs (i.e., the job set $[\beta, \pi(j), \gamma]$) minus that of the last $n-j+1$ jobs (i.e., the job set $[\pi(j), \gamma]$) in $tmpIC(\pi, i, j)$, hence represents the weighted tardiness of β in π'. Besides, $t_{j+1}^{IC(\pi,i,j)}$ represents the starting time of the $j+1$-th job in π', and thus the last term in (1) is the weighted tardiness of γ. Finally, the first three terms respectively represent the weighted tardiness of α, $\pi'(i)$, and $\pi'(j)$. Fig. 2 illustrates the weighted tardiness contributed from each term.

We now consider the special case $i = j - 1$. It is easy to see that the weighted tardiness of β should be 0, which implies $t_j^{tmpIC(\pi,i,j)} = t_{i+1}^{tmpIC(\pi,i,j)}$. To see this, $t_j^{tmpIC(\pi,i,j)} = t_{i+1}^{tmpIC(\pi,i,j)} + \Delta_{\pi'(i+1,j-1)} + s_{\pi'(j-1),\pi(j)} = t_{i+1}^{tmpIC(\pi,i,j)}$ since $\Delta_{\pi'(i+1,j-1)} = \Delta_{\pi'(j,j-1)} = 0$, and since $s_{\pi'(j-1),\pi(j)} = s_{\pi'(i),\pi(j)} = s_{\pi(j),\pi(j)} = 0$. Now let us consider $t_{j+1}^{IC(\pi,i,j)}$ in (1). Since $\delta_{\pi'(i,i+1,j-1,j)} = \delta_{\pi'(i,j,j-1,j)} = s_{\pi'(i),\pi'(j)}$, we have $t_{j+1}^{IC(\pi,i,j)} = C_{\pi(i-1)} + s_{\pi'(i-1),\pi'(i)} + p_{\pi'(i)} + s_{\pi'(i),\pi'(j)} + p_{\pi'(j)} + s_{\pi'(j),\pi'(j+1)}$, which is actually the starting time of $\pi'(j+1)$. In other cases of $i < j - 1$, we note that $\Delta_{\pi'(i+1,j-1)} + s_{\pi'(j-1),\pi(j)} = \Delta_{\pi(i+1,j-1)} + s_{\pi(j-1),\pi(j)}$ since $\pi'(j-1) = \pi(j-1)$.

To calculate $WT(\pi, i-1)$ for all i, a total of $O(n)$ time is required. If the values of $B_{i+1}(t_{i+1}^{tmpIC(\pi,i,j)})$, $B_j(t_j^{tmpIC(\pi,i,j)})$, $B_{j+1}(t_{j+1}^{IC(\pi,i,j)})$, and $WT(\pi, i-1)$ are calculated in advance, then $f(\pi')$ can be evaluated in constant time.

Theorem 1. *For a given sequence π with n jobs, it takes $O(n^2 \log n)$ time to compute $f(IC(\pi, i, j))$ for all $1 \le i < j \le n$.*

Proof. For a given i, let $T_{i+1}^{tmpIC(\pi,i,*)} := \{t_{i+1}^{tmpIC(\pi,i,l)} : i < l \le n\}$ be the set consisting of all possible starting times of the $i+1$-th job in $tmpIC(\pi, i, l)$. $T_{i+1}^{tmpIC(\pi,i,*)}$ can be sorted by any standard sorting technique in $O(n \log n)$ time, and thus

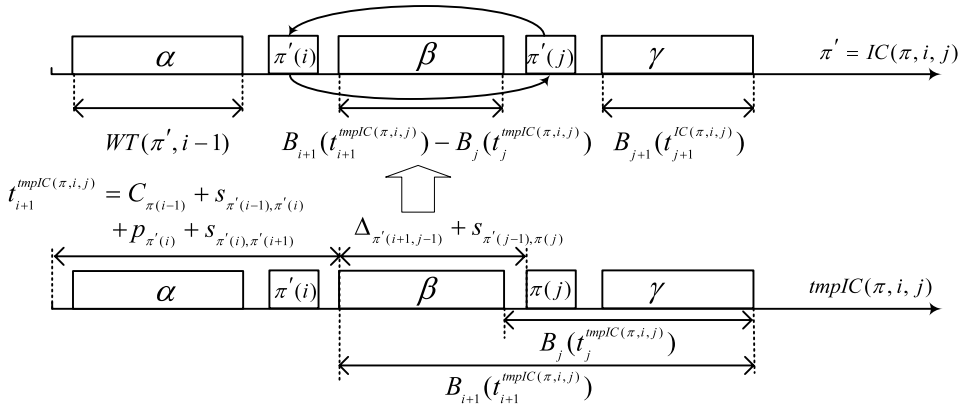

Fig. 2. A graphical representation for the weighted tardiness of π' and the relation between $IC(\pi, i, j)$ and $tmpIC(\pi, i, j)$. Note that the only difference is the j-th job.

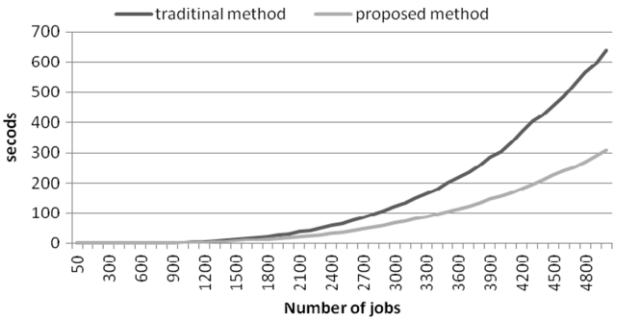

Fig. 3. Computing time with different number of jobs for interchange neighborhood search.

determining which interval $[BP_k, BP_{k+1}]$ the value of $t_{i+1}^{tmpIC(\pi,i,l)}$ falls in, in order, takes $O(n)$ time. Hence, $B_{i+1}(t_{i+1}^{tmpIC(\pi,i,l)})$ for all $t_{i+1}^{tmpIC(\pi,i,l)} \in T_{i+1}^{tmpIC(\pi,i,*)}$ can be evaluated with a total of $O(n \log n)$ time.

For a given j, let $T_j^{tmpIC(\pi,*,j)} := \{t_j^{tmpIC(\pi,l,j)} : 1 \le l < j\}$ be the set consisting of the starting time of the j-th job in $tmpIC(\pi, i, l)$, and $T_{j+1}^{IC(\pi,*,j)} := \{t_{j+1}^{IC(\pi,l,j)} : 1 \le l < j\}$ be the set consisting of the starting time of the $j+1$-th job in $IC(\pi, l, j)$. Similarly, both $B_j(t_j^{tmpIC(\pi,l,j)})$ for all $t_j^{tmpIC(\pi,l,j)} \in T_j^{tmpIC(\pi,*,j)}$ and $B_{j+1}(t_{j+1}^{IC(\pi,l,j)})$ for all $t_{j+1}^{IC(\pi,l,j)} \in T_{j+1}^{IC(\pi,*,j)}$ can be evaluated with a total of $O(n \log n)$ time.

Summing up for all $1 \le i < j \le n$, one can evaluate the values of $B_{i+1}(t_{i+1}^{tmpIC(\pi,i,l)})$ for all $t_{i+1}^{tmpIC(\pi,i,l)} \in T_{i+1}^{tmpIC(\pi,i,*)}$, $B_j(t_j^{tmpIC(\pi,l,j)})$ for all $t_j^{tmpIC(\pi,l,j)} \in T_j^{tmpIC(\pi,*,j)}$, and $B_{j+1}(t_{j+1}^{IC(\pi,l,j)})$ for all $t_{j+1}^{IC(\pi,l,j)} \in T_{j+1}^{IC(\pi,*,j)}$ in $O(n^2 \log n)$ time. □

Here, we present an implementation of the interchange neighborhood search for single machine tardiness scheduling with sequence-dependent setups. We use the three problem coefficients (τ, R, η) to constitute the dimensions of the problem instances (see Lee et al. [11]). In our study we set $\tau = 0.3$, $R = 0.25$, $\eta = 0.25$. Instances with $n = 50, 100, 200, \ldots, 4900, 5000$ were randomly generated, and for each job j ($j = 1, \ldots, n$), an integer processing time p_j is generated from the uniform distribution on [50, 150] and the weight w_j is generated from the uniform distribution on [0, 10]. The mean processing time $\bar{p}$ is therefore 100. The mean setup time $\bar{s}$ is then determined by η and the setup times are uniformly distributed over the interval $[0, 2\bar{s}]$. The due dates are generated from a composite uniform distribution based on R and τ. With probability τ the due date is uniformly distributed over the interval $[\bar{d} - R\bar{d}, \bar{d}]$ and with probability $(1-\tau)$ over the interval $[\bar{d}, \bar{d} + (C_{max} - \bar{d})R]$. We coded in C++ and run on an Intel Core 2 CPU (2.5 GHz) computer with 1.96 GB RAM. The time to compute the total weighted tardiness of interchange neighborhood is shown in Fig. 3, and the experimental result is consistent with the Theorem 1.

3.2. The insertion neighborhood

For a given sequence π, the insertion neighborhood can be obtained by extracting a job from its position and reinserting it into another position. That is, given a pair of indices i and j ($1 \le i, j \le n$), we let $\pi = [\alpha, \pi(i), \beta, \pi(j), \gamma]$ if $i < j$ and $\pi = [\alpha, \pi(j), \beta, \pi(i), \gamma]$ if $i > j$. The insertion neighborhood of π by extracting $\pi(i)$ and reinserting it into position j can be expressed as

$$IS(\pi, i, j) = \begin{cases} [\alpha, \beta, \pi(j), \pi(i), \gamma] & \text{if } i < j, \\ [\alpha, \pi(i), \pi(j), \beta, \gamma] & \text{if } i > j. \end{cases}$$

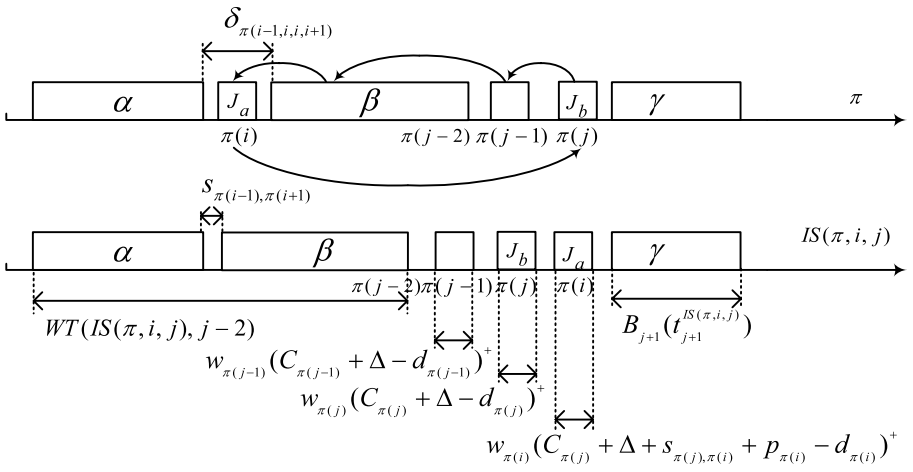

Fig. 4. A graphical representation for the weighted tardiness of $IS(\pi, i, j)$. Note that $\Delta = s_{\pi(i-1),\pi(i+1)} - \delta_{\pi(i-1,i,i,i+1)}$.

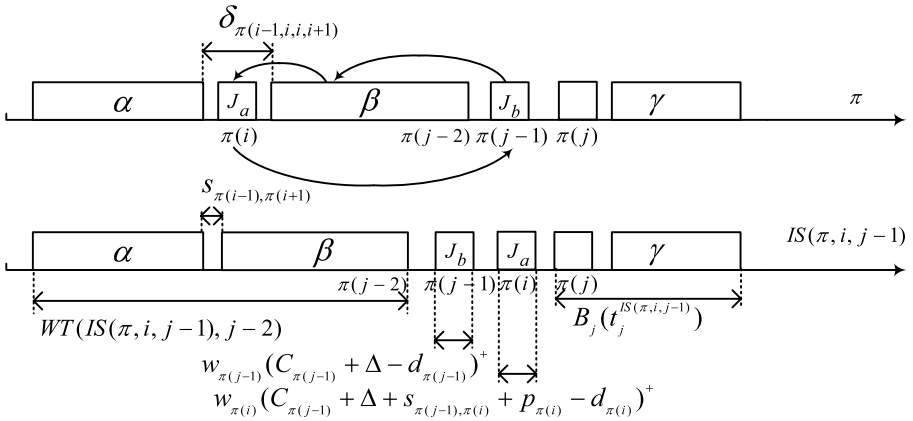

Fig. 5. A graphical representation for the weighted tardiness of $IS(\pi, i, j-1)$.

Here we only consider the case $i < j$. The case $i > j$ can be derived in a similar manner. Let us observe that

$$\begin{aligned} f(IS(\pi,i,j)) &= [WT(IS(\pi,i,j),j-2)] + [w_{\pi(j-1)}(C_{\pi(j-1)}+\Delta-d_{\pi(j-1)})^+] + [w_{\pi(j)}(C_{\pi(j)}+\Delta-d_{\pi(j)})^+] \\ &\quad + [w_{\pi(i)}(C_{\pi(j)}+\Delta+s_{\pi(j),\pi(i)}+p_{\pi(i)}-d_{\pi(i)})^+] + [B_{j+1}(t_{j+1}^{IS(\pi,i,j)})], \end{aligned} \tag{2}$$

$$f(IS(\pi,i,j-1)) = \begin{cases} [WT(IS(\pi,i,j-1),j-2)] + [w_{\pi(j-1)}(C_{\pi(j-1)}+\Delta-d_{\pi(j-1)})^+] \\ \quad +[w_{\pi(i)}(C_{\pi(j-1)}+\Delta+s_{\pi(j-1),\pi(i)}+p_{\pi(i)}-d_{\pi(i)})^+] \\ \quad +[B_j(t_j^{IS(\pi,i,j-1)})] \quad \text{if } i < j-1, \\ f(\pi) \quad \text{if } i = j-1, \end{cases} \tag{3}$$

where

$$\begin{aligned} &\Delta = s_{\pi(i-1),\pi(i+1)} - \delta_{\pi(i-1,i,i,i+1)}, \\ &t_j^{IS(\pi,i,j-1)} = C_{\pi(j-1)} + \Delta + s_{\pi(j-1),\pi(i)} + p_{\pi(i)} + s_{\pi(i),\pi(j)}, \\ &t_{j+1}^{IS(\pi,i,j)} = C_{\pi(j)} + \Delta + s_{\pi(j),\pi(i)} + p_{\pi(i)} + s_{\pi(i),\pi(j+1)}. \end{aligned}$$

We introduce a new notation Δ, which is used for adjusting the time span between α and β from π to $IS(\pi, i, j)$, and further define $t_j^{IS(\pi,i,j-1)}$ and $t_{j+1}^{IS(\pi,i,j)}$ as the starting times of $\pi(j)$ and $\pi(j+1)$ in $IS(\pi, i, j-1)$ and $IS(\pi, i, j)$, respectively. The first and the last terms in (2) (and in (3)) respectively define the weighted tardiness of the first $j-2$ and the last $n-j$ (and last $n-j+1$) jobs in $IS(\pi, i, j)$ (in $IS(\pi, i, j-1)$). The second, third, and fourth terms in (2) define the weighted tardiness of $\pi(j-1)$, $\pi(j)$, and $\pi(i)$ in $IS(\pi, i, j)$. Similarly, the second and the third terms in (3) define the weighted tardiness of $\pi(j-1)$ and $\pi(i)$ in $IS(\pi, i, j-1)$. To clarify, the tardiness values contributed from each term in (2) and (3) are respectively illustrated in Fig. 4 and Fig. 5.

Let us observe that the values of $WT(IS(\pi, i, j), j-2)$ and $WT(IS(\pi, i, j-1), j-2)$ are equal because the first $j-2$ jobs in $IS(\pi, i, j)$ and $IS(\pi, i, j-1)$ are the same. Further, the second terms in (2) and (3) are identical. Therefore, from (2) and (3) a recursion equation can be obtained as follows:

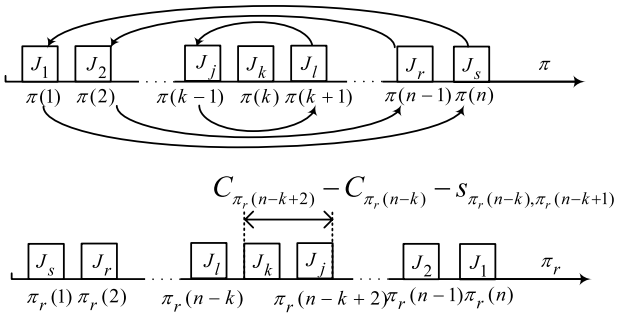

Fig. 6. An example of the time span between $\pi_r(n-k+2)$ and $\pi_r(n-k)$ in π_r.

$$\begin{aligned} f(IS(\pi,i,j)) &= f(IS(\pi,i,j-1)) - [w_{\pi(i)}(C_{\pi(j-1)} + \Delta + s_{\pi(j-1)\pi(i)} + p_{\pi(i)} - d_{\pi(i)})^+] \\ &\quad + [w_{\pi(j)}(C_{\pi(j)} + \Delta - d_{\pi(j)})^+] + [w_{\pi(i)}(C_{\pi(j)} + \Delta + s_{\pi(j),\pi(i)} + p_{\pi(i)} - d_{\pi(i)})^+] \\ &\quad - [B_j(t_j^{IS(\pi,i,j-1)}) - B_{j+1}(t_{j+1}^{IS(\pi,i,j)})]. \end{aligned} \tag{4}$$

If $B_j(t_j^{IS(\pi,i,j-1)})$ and $B_{j+1}(t_{j+1}^{IS(\pi,i,j)})$ are calculated in advance and the value of $f(IS(\pi,i,j-1))$ is given, $f(IS(\pi,i,j))$ can be calculated by performing (4) in constant time since the second, third, and fourth terms in (4) can be calculated in constant time.

Theorem 2. *For a given sequence π with n jobs, it takes $O(n^2 \log n)$ time to compute $f(IS(\pi,i,j))$ for all i and j $(1 \le i, j \le n)$.*

Proof. Let $T_{j+1}^{IS(\pi,*,j)} := \{t_{j+1}^{IS(\pi,l,j)} : 1 \le l < j\}$ for a given j $(1 < j \le n)$. $T_{j+1}^{IS(\pi,*,j)}$ can be sorted in non-decreasing order in $O(n \log n)$. Hence, as in Theorem 1, $B_{j+1}(t_{j+1}^{IS(\pi,l,j)})$ for all $t_{j+1}^{IS(\pi,l,j)} \in T_{j+1}^{IS(\pi,*,j)}$ can be evaluated, in order, in $O(n)$ time. Thus, for all $1 < j \le n$ it takes $O(n^2 \log n)$ time to evaluate the values of $B_{j+1}(t_{j+1}^{IS(\pi,l,j)})$ for all $t_{j+1}^{IS(\pi,l,j)} \in T_{j+1}^{IS(\pi,*,j)}$. With these values, Eq. (4) can be performed in constant time and thus $f(IS(\pi,i,j) : i < j)$ for a given i can be evaluated with a total of $O(n)$ time. Summing up for all i, $O(n^2 \log n)$ time is required to evaluate $f(IS(\pi,i,j : i < j))$, and so is to evaluate $f(IS(\pi,i,j : i > j))$. □

3.3. The twist neighborhood

The twist neighborhood can be obtained by taking a subset of the jobs and processing them in reverse order. That is, given a sequence $\pi = [\alpha, \pi(i), \pi(i+1), \ldots, \pi(j-1), \pi(j), \gamma]$ and a pair of indices i and j $(1 \le i < j \le n)$, the twist neighborhood is

$$TW(\pi,i,j) = [\alpha, \pi(j), \pi(j-1), \ldots, \pi(i+1), \pi(i), \gamma].$$

We now show how to search the twist neighborhood in $O(n^2 \log n)$ time. We first sort jobs of $TW(\pi,i,j)$ in non-increasing order of lateness and store them in an ordered list *RevLatenessList*. Then, we reverse π (i.e., $TW(\pi,1,n)$) to obtain $\pi_r = [\gamma_{reverse}, \pi(j), \ldots, \pi(i), \alpha_{reverse}]$ and compute the completion times of jobs in π_r as follows: $C_{\pi_r(k)} = \sum_{l=1}^{k} (s_{\pi_r(l-1),\pi_r(l)} + p_{\pi_r(l)})$ for all $k \in \{1, \ldots, n\}$, where $\pi_r(0) = 0$ is a dummy job. Note that $\pi(k) = \pi_r(n-k+1)$. In addition, let $F_k(t)$ be the weighted tardiness if we schedule only jobs $\pi_r(n-k+1), \pi_r(n-k+2), \ldots, \pi_r(n)$ and if job $\pi_r(n-k+1)$ starts at time t. Thus,

$$F_k(t) = \sum_{l=n-k+1}^{n} w_{\pi_r(l)}(t + C_{\pi_r(l)} - C_{\pi_r(n-k)} - s_{\pi_r(n-k),\pi_r(n-k+1)} - d_{\pi_r(l)})^+.$$

Fig. 6 illustrates an example of the time span between $\pi_r(n-k+2)$ and $\pi_r(n-k)$ in π_r.

The method for evaluating $F_j(t)$ is similar to that for evaluating $B_j(t)$. For a given j, $F_j(t)$ is a piecewise linear convex function of t and its slope changes only at $t = d_{\pi_r(l)} - C_{\pi_r(l)} + C_{\pi_r(n-j)} + s_{\pi_r(n-j),\pi_r(n-j+1)}$ for some $l \in \{n-j+1, \ldots, n\}$. The order of the breakpoints agrees with the order of the jobs on *RevLatenessList*, so one can determine the breakpoints and the linear functions on the intervals between these breakpoints, in order, in $O(n)$ time.

Using this notation, the total weighted tardiness of sequence $TW(\pi,i,j)$ for all $i < j$ can be formulated as follows:

$$f(TW(\pi,i,j)) = [WT(\pi,i-1)] + [F_j(t_i^{tmpTW(\pi,i,j)}) - F_{i-1}(t_{j+1}^{tmpTW(\pi,i,j)})] + [B_{j+1}(t_{j+1}^{TW(\pi,i,j)})], \tag{5}$$

where

$$\begin{aligned} t_i^{tmpTW(\pi,i,j)} &= C_{\pi(i-1)} + s_{\pi(i-1),\pi(j)}, \\ t_{j+1}^{tmpTW(\pi,i,j)} &= t_i^{tmpTW(\pi,i,j)} + \delta_{\pi_r(n-j+1,n-i+1)} + s_{\pi_r(n-i+1),\pi_r(n-i+2)}, \\ t_{j+1}^{TW(\pi,i,j)} &= t_i^{tmpTW(\pi,i,j)} + \delta_{\pi_r(n-j+1,n-i+1)} + s_{\pi(i),\pi(j+1)}. \end{aligned}$$

Notation $t_i^{tmpTW(\pi,i,j)}$ and $t_{j+1}^{tmpTW(\pi,i,j)}$ respectively represent the starting times of the i-th job and the $j+1$-th job with respect to a temporary sequence $tmpTW(\pi,i,j) = [\alpha, \pi(j), \pi(j-1), \ldots, \pi(i+1), \pi(i), \alpha_{reverse}]$, which consists of parts of

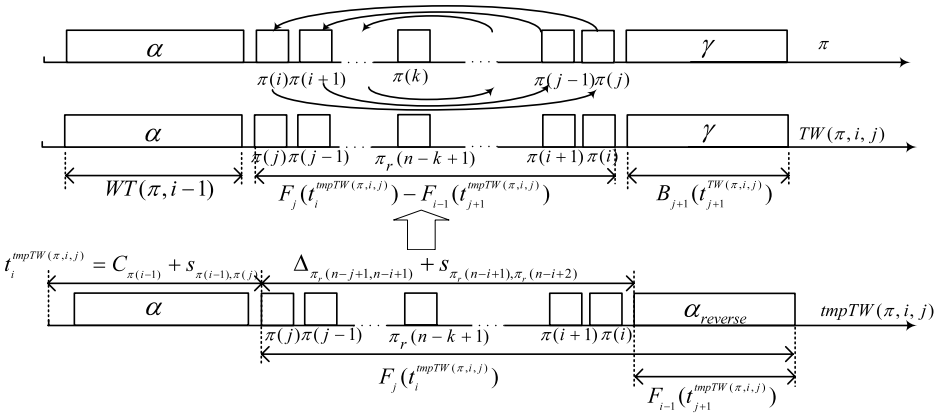

Fig. 7. A graphical representation for the weighted tardiness of $TW(\pi, i, j)$ and the relation between $TW(\pi, i, j)$ and $tmpTW(\pi, i, j)$.

π and $TW(\pi, i, j)$. The second term in (5), the weighted tardiness of the last j jobs (i.e., the job set $[\pi(j), \ldots, \pi(i), \alpha_{reverse}]$) minus that of the last $i-1$ jobs (i.e., the job set $[\alpha_{reverse}]$) in $tmpTW(\pi, i, j)$, represents the weighted tardiness of $[\pi(j), \ldots, \pi(i)]$ in $TW(\pi, i, j)$. The first and the last terms in (5) are the weighted tardiness contributed from α and γ. Fig. 7 illustrates the weighted tardiness contributed from each term.

Theorem 3. *For a given sequence π with n jobs, it takes $O(n^2 \log n)$ time to compute $f(TW(\pi, i, j))$ for all $1 \le i < j \le n$.*

Proof. Let $T_i^{tmpTW(\pi,i,*)} := \{t_i^{tmpTW(\pi,i,l)} : i < l \le n\}$. $T_i^{tmpTW(\pi,i,*)}$ for a given i can be sorted in $O(n \log n)$ time; hence, as in Theorem 1, $F_j(t_i^{tmpTW(\pi,i,l)})$ for all $t_i^{tmpTW(\pi,i,l)} \in T_i^{tmpTW(\pi,i,*)}$ can be evaluated, in order, in $O(n)$ time. Similarly, let $T_{j+1}^{tmpTW(\pi,*,j)} := \{t_{j+1}^{tmpTW(\pi,l,j)} : 1 \le l < j\}$ and $T_{j+1}^{TW(\pi,*,j)} := \{t_{j+1}^{TW(\pi,l,j)} : 1 \le l < j\}$. For a given j, a total of $O(n \log n)$ time is required to evaluate $F_{i-1}(t_{j+1}^{tmpTW(\pi,l,j)})$ for all $t_{j+1}^{tmpTW(\pi,l,j)} \in T_{j+1}^{tmpTW(\pi,*,j)}$ and to evaluate $B_{j+1}(t_{j+1}^{TW(\pi,l,j)})$ for all $t_{j+1}^{TW(\pi,l,j)} \in T_{j+1}^{TW(\pi,*,j)}$. Summing up for all i and j, the values of $F_j(t_i^{tmpTW(\pi,i,l)})$ for all $t_i^{tmpTW(\pi,i,l)} \in T_i^{tmpTW(\pi,i,*)}$, $F_{i-1}(t_{j+1}^{tmpTW(\pi,l,j)})$ for all $t_{j+1}^{tmpTW(\pi,l,j)} \in T_{j+1}^{tmpTW(\pi,*,j)}$, and $B_{j+1}(t_{j+1}^{TW(\pi,l,j)})$ for all $t_{j+1}^{TW(\pi,l,j)} \in T_{j+1}^{TW(\pi,*,j)}$ can be evaluated with a total of $O(n^2 \log n)$ time. □

4. Extensions to restricted neighborhood

To reduce the computation time in a neighborhood search, one may restrict the set of candidate jobs because, experimentally, it seldom occurs that the best candidate job is far from the select job. This neighborhood search version is called the restricted neighborhood search, which has been implemented in [11]. In this section, we extend the developed neighborhood search algorithms to the restricted versions of the insertion and twist neighborhoods.

Corollary 1. *Given a sequence π with n jobs, it takes $O(nk \log k + n^2)$ time to search the insertion neighborhood in which a candidate job is selected from among k jobs nearest to the selected job.*

Proof. Recall that in Theorem 2 the most time-consuming step is to sort the set $T_{j+1}^{IS(\pi,*,j)} := \{t_{j+1}^{IS(\pi,l,j)} : 1 \le l < j\}$ for a given j $(1 \le j \le n)$. However, the size of $T_{j+1}^{IS(\pi,*,j)}$ in this case has been restricted to at most k elements, implying that $T_{j+1}^{IS(\pi,*,j)}$ can be sorted in $O(k \log k)$ time. In addition, it takes $O(n)$ to evaluate the value of $B_j(t_{j+1}^{IS(\pi,l,j)})$ for all $t_{j+1}^{IS(\pi,l,j)} \in T_{j+1}^{IS(\pi,*,j)}$, in order, and compute the recursion equation (4). Hence, for all j the restricted version of insertion neighborhood is searched in $O(nk \log k + n^2)$ time. □

Corollary 2. *Given a sequence π with n jobs, it takes $O(nk \log k + n^2)$ time to search the twist neighborhood in which a candidate job is selected from among k jobs nearest to the selected job.*

Proof. Similar to the proof of Corollary 1, for any fixed i and j, to sort $T_i^{tmpTW(\pi,i,*)}$, $T_{j+1}^{tmpTW(\pi,*,j)}$, and $T_{j+1}^{TW(\pi,*,j)}$ with at most k elements takes $O(k \log k)$ time, and to evaluate the values of $F_j(t_i^{tmpTW(\pi,i,l)})$ for all $t_i^{tmpTW(\pi,i,l)} \in T_i^{tmpTW(\pi,i,*)}$, $F_{i-1}(t_{j+1}^{tmpTW(\pi,l,j)})$ for all $t_{j+1}^{tmpTW(\pi,l,j)} \in T_{j+1}^{tmpTW(\pi,*,j)}$, and $B_{j+1}(t_{j+1}^{TW(\pi,l,j)})$ for all $t_{j+1}^{TW(\pi,l,j)} \in T_{j+1}^{TW(\pi,*,j)}$, respectively, takes $O(n)$ time. Hence, for all i and j the restricted version of twist neighborhood search is searched in $O(nk \log k + n^2)$ time. □

5. Conclusion

In this paper we have addressed the $1/s_{ij}/\sum w_j T_j$ problem, for which we have improved the time complexity of searching the interchange, insertion and the twist neighborhoods from $O(n^3)$ to $O(n^2 \log n)$. We also have improved the restricted version of insertion and twist neighborhoods from $O(n^2 k)$ to $O(nk \log k + n^2)$.

References

[1] A. Allahverdi, J.N.D. Gupta, T. Aldowaisan, A review of scheduling research involving setup considerations, OMEGA 27 (1999) 219–239.
[2] V.A. Cicirello, Weighted tardiness scheduling with sequence-dependent setups: a benchmark library, Technical Report (2003), Intelligent Coordination and Logistics Laboratory, Robotics Institute, Carnegie Mellon University.
[3] Ö. Ergun, New neighborhood search algorithms based on exponentially large neighborhoods, Operations Research Center Thesis, MIT, 2001.
[4] Ö. Ergun, J.B. Orlin, Fast neighborhood search for the single machine total weighted tardiness problem, Operations Research Letters 34 (2006) 41–45.
[5] P.M. França, A. Mendes, P. Moscato, A memetic algorithm for the total tardiness single machine scheduling problem, European Journal of Operational Research 132 (2001) 224–242.
[6] C. Gagné, W.L. Price, M. Gravel, Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times, Journal of the Operational Research Society 53 (2002) 895–906.
[7] C. Gagné, M. Gravel, W.L. Price, A new hybrid Tabu-VNS metaheuristic for solving multiple objective scheduling problems, in: Proceedings of the Fifth Metaheuristics International Conference, Kyoto, Japan, 2003.
[8] H. Kellerer, V.A. Strusevich, A fully polynomial approximation scheme for the single machine weighted total tardiness problem with a common due date, Theoretical Computer Science 369 (2006) 230–238.
[9] S.G. Kolliopoulos, G. Steiner, Approximation algorithms for minimizing the total weighted tardiness on a single machine, Theoretical Computer Science 355 (2006) 261–273.
[10] E.L. Lawler, A 'pseudopolynomial' algorithm for sequencing jobs to minimize total tardiness, Annals of Discrete Mathematics 1 (1997) 331–342.
[11] Y.H. Lee, K. Bhaskaram, M. Pinedo, A heuristic to minimize the total weighted tardiness with sequence-dependent setups, IIE Transactions 29 (1997) 45–52.
[12] C.J. Liao, H.C. Juan, An ant colony optimization for single machine tardiness scheduling with sequence-dependent setups, Computers & Operations Research 34 (2007) 1899–1909.
[13] M. Pinedo, Scheduling: Theory, Algorithm, and Systems, second ed., Prentice-Hall, 2002.
[14] P.A. Rubin, G.L. Ragatz, Scheduling in a sequence dependent setup environment with genetic search, Computers & Operations Research 22 (1995) 85–99.
[15] K.C. Tan, R. Narasimhan, Minimizing tardiness on a single processor with sequence-dependent setup times: a simulated annealing approach, OMEGA 25 (1997) 619–634.