ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Theoretical Computer Science 363 (2006) 257–265

Theoretical Computer Science

www.elsevier.com/locate/tcs

Improved algorithms for two single machine scheduling problems☆

Yong He, Weiya Zhong*, Huikun Gu

Department of Mathematics, State Key Lab of CAD & CG, Zhejiang University, Hangzhou 310027, PR China

Abstract

In this paper, we investigate two single machine scheduling problems. The first problem addresses a class of the two-stage scheduling problems in which the first stage is job production and the second stage is job delivery. For the case that jobs are processed on a single machine and delivered by a single vehicle to one customer area, with the objective of minimizing the time when all jobs are completed and delivered to the customer area and the vehicle returns to the machine, an approximation algorithm with a worst-case ratio of $\frac{5}{3}$ is known and no approximation can have a worst-case of $\frac{3}{2}$ unless $P = NP$. We present an improved approximation algorithm with a worst-case ratio of $\frac{53}{35}$, which only leaves a gap of $\frac{1}{70}$. The second problem is a single machine scheduling problem subject to a period of maintenance. The objective is to minimize the total completion time. The best known approximation algorithm has a worst-case ratio of $\frac{20}{17}$. We present a polynomial time approximation scheme.

Keywords: Scheduling; Delivery; Worst-case ratio

1. Introduction

In this paper, we consider two single machine scheduling problems, which have strong background in supply chain management and manufacture management.

The first problem is a scheduling problem with job delivery coordination, which is first proposed by Chang and Lee [2], and can be described as follows: We are given n jobs $N = \{J_1, J_2, \ldots, J_n\}$ which must be first non-preemptively processed in a manufacturing system and then delivered to respective customers. Job J_j, $j = 1, 2, \ldots, n$ needs a processing time of p_j in the manufacturing system, and has a size s_j which represents the physical space J_j occupies when this job is loaded in the vehicle. One vehicle is available to deliver finished jobs in batches, and has a capacity z which means that finished jobs can be arranged to fit in the physical space provided by the vehicle as long as their total size does not exceed z. The vehicle is initially located at the manufacturing facility. All jobs delivered together in one shipment are defined as a delivery batch. A transportation time depending on customer area is associated with each delivery batch. Furthermore, we define a one customer area as a location where a group of customers are located in close proximity to each other. The goal is to find a schedule for processing jobs in manufacturing system and delivering finished jobs to the corresponding customers such that the time required for all jobs in N to be processed and delivered to the respective customer(s) is minimized. To evaluate this goal, we define the makespan of a schedule, denoted by

☆ Research supported by National Natural Science Foundation of China (10271110, 60021201).
* Corresponding author.
E-mail address: zhongcherry@163.com (W. Zhong).

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.04.014

$C_{\max}$, as *the time when the vehicle finishes delivering the last batch to the customer site*(*s*) *and returns to the machine*(*s*). Then the problem is to find a schedule to minimize the makespan.

As we know, coordination of activities among different stages in the supply chain has become one of the most important topic in production and operations management research in last decade. For the research on the coordination of production and delivery schedule, one may refer to [2]. Different from traditional scheduling problems which implicitly assume that there are infinitely many vehicles for delivering finished products to their destinations so that finished products can be transported to customers without delay, the above problem incorporates the delivery plan of a vehicle into a manufacturing system. It models a class of the two-stage scheduling problem in which the first stage is job production and the second stage is job delivery. The focus is on the study of the integration of production scheduling with delivery of finished products to customers, which measures the customer service level.

Three strongly *NP*-hard cases of the above problem are considered in [2]: for the case that the manufacturing system consists of a single machine and there is one customer area, Chang and Lee presented a polynomial time algorithm $H1$ with a worst-case ratio of $\frac{5}{3}$, while no polynomial time algorithm can have a worst-case ratio of smaller than $\frac{3}{2}$ unless $P = NP$. For the case that the manufacturing system consists of two identical machines and there is one customer area, they presented a polynomial time algorithm $H2$ with a worst-case ratio of 2. For the case that the manufacturing system consists of a single machine and there are two customer areas, they presented a polynomial time algorithm $H3$ with a worst-case ratio of 2, too.

In this paper, we revisit the first case of the above problem, which is denoted by $1 \to D, k = 1|v = 1, c = z|C_{\max}$. Here "$1 \to D, k = 1$" means that jobs are first processed on a single machine and then delivered to customer(s) who are located in one area. "$v = 1, c = z$" means that there is only one vehicle with capacity z. We will present a modified algorithm $MH1$ with a worst-case ratio of $\frac{3}{2} + \frac{1}{70} = \frac{53}{35}$, which greatly improves the known upper bound of $\frac{5}{3}$ and is quite close to the lower bound of $\frac{3}{2}$.

The second considered problem in this paper is a single machine scheduling problem with a machine availability constraint, which can be described as follows: we are given n jobs $N = \{J_1, J_2, \ldots, J_n\}$ which must be processed on a single machine. Job J_j has a processing time p_j, $j = 1, 2, \ldots, n$. All jobs are available at time zero, whereas the machine has a maintenance period during the processing of jobs, i.e., the machine cannot process any job during the given time window $[R, R + L]$. Preemptions are not allowed. Hence a job that is preempted due to the maintenance must be restarted after the machine is repaired. The objective is to find a schedule such that the total completion time is minimized. This problem is denoted by $1, h_1||\sum C_i$ [6].

Adiri et al. [1] and Lee and Liman [5] showed that the problem $1, h_1||\sum C_i$ is *NP*-hard. They also studied algorithm *SPT* (*shortest processing time*) as an approximation algorithm solving this problem. Lee and Liman [5] proved that the worst-case ratio of *SPT* is $\frac{9}{7}$. Recently, Sadfi et al. [6] proposed a modified algorithm *MSPT* with a worst-case ratio of $\frac{20}{17}$. This algorithm is based on a post-optimization of the *SPT* algorithm by applying a 2-OPT procedure. In this paper, we will extend this idea to a general k, k-exchange procedure. Then we will propose a polynomial time approximation scheme (*PTAS*) based on this new procedure. Hence our result greatly improves the known result.

2. Problem $1 \to D, k = 1|v = 1, c = z|C_{\max}$

This section considers the two-stage scheduling problem with a single machine and one customer area: $1 \to D, k = 1|v = 1, c = z|C_{\max}$. Let P be the total processing time of all the jobs. Let t be the one-way transportation time between the machine and the customer, therefore, each delivery has the same transportation time $T = 2t$.

2.1. Preliminaries and algorithm description

Property 1 (*Chang and Lee [2]*). *There exists an optimal schedule for the problem* $1 \to D, k = 1|v = 1, c = z|C_{\max}$ *that satisfies the following conditions*:

(1) *Jobs are processed on the machine without idle time.*
(2) *Jobs assigned to one batch are processed consecutively on the machine.*
(3) *Jobs assigned to one batch can be processed on the machine in any order.*
(4) *Batches are delivered in non-decreasing order of the total processing time of jobs in each batch.*

Therefore, only schedules satisfying the above properties are considered further. Also, batches will be indexed and delivered in non-decreasing order of the total processing time of the jobs in each batch.

Lemma 1 (*Chang and Lee [2]*). *For any schedule satisfying Property* 1, *if* $C_{\max} > P + T$, *then* $P_1 < T$ *and* $C_{\max} = P_1 + KT$, *in which* P_1 *denotes the total processing time of the jobs in the first batch and* K *denotes the number of batches in the schedule*.

Since different jobs with different sizes can be delivered in one batch, the packing step can be viewed as a bin-packing problem. Algorithms *FF* (*first fit*) and *FFD* (*first fit decreasing*) are two classical algorithms for the bin-packing problem. We will apply them as sub-procedures for solving our problem. Note that algorithms *FF* and *FFD* are based on the job sizes and the vehicle capacity z in this paper.

Lemma 2. *For an instance* I *of the bin-packing problem, let* $OPT(I)$, $FF(I)$, $FFD(I)$ *be the numbers of used bins in an optimal solution, the solutions yielded by FF and FFD, respectively. We have*

(1) (*Simchi-Levi* [7]) $FF(I) \leqslant \frac{7}{4}OPT(I)$;

(2) (*Yue* [8]) $FFD(I) \leqslant \frac{11}{9}OPT(I) + 1$.

The following algorithm $H1$ was proposed by Chang and Lee [2] for solving $1 \to D, k = 1|v = 1, c = z|C_{\max}$.

Algorithm $H1$**.** (1) Assign jobs to batches by algorithm *FFD*. Let the total number of resulting batches be b_1.

(2) Define P_k as the total processing time of the jobs in the kth batch, $k = 1, 2, \ldots, b_1$. Reindex these batches such that $P_1 \leqslant P_2 \leqslant \cdots \leqslant P_{b_1}$, and denote the kth batch as B_k.

(3) Starting with B_1, assign jobs in B_k to the machine, for $k = 1, 2, \ldots, b_1$. Jobs within each batch can be sequenced in an arbitrary order.

(4) Dispatch each finished but undelivered batch whenever the vehicle becomes available. If multiple batches have been completed when the vehicle becomes available, dispatch the batch with the smallest index.

It is clear that the time complexity of $H1$ is $O(n \log n)$. It is shown in [2] that the worst-case ratio of $H1$ is $\frac{5}{3}$. Furthermore, since the bin-packing problem is a special case of our problem, it is impossible to have a polynomial time approximation algorithm with a worst-case ratio of $\frac{3}{2}$ unless $P = NP$.

Note that there is a point which prevents the worst-case ratio of $H1$ to be better: $H1$ assigns jobs to batches by *FFD*, which does not take the processing times of jobs into consideration. The total processing time of jobs in the first batch may be much larger than that in the optimal solution. Our improved algorithm applies a fully polynomial time approximation scheme (*FPTAS*) of the knapsack problem, which enables the vehicle to start delivering jobs earlier. Recall that for any instance of the knapsack problem, we are given n items, each with a profit and a size, and a knapsack with limited capacity. We wish to put items into the knapsack such that the total size of the selected items is not greater than the knapsack capacity and the total profit of the selected items is maximized. For this *NP*-hard problem, among others, Lawler [4] proposed an *FPTAS* with a time complexity of $O(n \log(1/\varepsilon) + 1/\varepsilon^4)$, where $1 - \varepsilon$ is the worst-case ratio; and Kellerer and Pferschy [3] also proposed an *FPTAS* with a time complexity of $O(n \min\{\log n, \log(1/\varepsilon)\} + (1/\varepsilon^2) \min\{n, (1/\varepsilon)\log(1/\varepsilon)\})$.

Now we are ready to present our improved algorithm.

Algorithm $MH1$**.** (1) Run algorithm $H1$. Let the obtained schedule be σ_1 with makespan C_1. If $b_1 \neq 3$, stop; Else, go to Step 2.

(2) Construct an instance of the knapsack problem as follows: for each job J_j, $j = 1, 2, \ldots, n$, construct an item with profit p_j and size s_j, and let the knapsack capacity be z. Run any *FPTAS* for the knapsack problem with $\varepsilon = \frac{2}{35}$, and denote by N_1 the set of items put into the knapsack. Reindex all jobs such that N_1 is at the head.

(3) Assign jobs to batches by algorithm *FF*. Let the total number of resulting batches be b_2.

(4) Run Steps 2–4 of algorithm $H1$ except that denote by B'_k the kth batch, and by P'_k the total processing times of B'_k, $k = 1, 2, \ldots, b_2$. Let the obtained schedule be σ_2 with makespan C_2.

(5) Compare C_1 and C_2. Select the smaller one as output.

Remark 3. The jobs corresponding to the items in N_1 are assigned to the same batch by algorithm *FF* in Step 3 of algorithm $MH1$.

When analyzing our algorithm, we use the following:

b_L^* is the number of batches if the jobs are assigned to batches by an optimal algorithm of the bin-packing problem; b^* the number of batches in the optimal schedule for our problem, P^* the optimal value of the instance of the knapsack problem constructed in Step 2, C^* the optimal makespan for our problem, C_{MH1} the makespan produced by $MH1$, y the total processing time of jobs in the first batch in the optimal solution.

Lemma 4. *If there are only two batches in the optimal schedule*, $P - y \leqslant P^*$.

Proof. Since jobs in each batch constitutes a feasible solution for the instance of the knapsack problem, and $P - y$ is the total processing time of the second batch in the optimal schedule, we have $P - y \leqslant P^*$. □

Lemma 5. $P'_{b_2} \geqslant \frac{33}{35} P^*$, *in which* P'_{b_2} *denotes the total processing time of jobs in the last batch of* σ_2 (*if exists*).

Proof. From Remark 3, we know that there exists k, $1 \leqslant k \leqslant b_2$, such that $P'_k \geqslant \frac{33}{35} P^*$. Since $P'_1 \leqslant P'_2 \leqslant \cdots \leqslant P'_{b_2}$, $P'_{b_2} \geqslant P'_k \geqslant \frac{33}{35} P^*$. □

2.2. Worst-case analysis of algorithm MH1

Lemma 6. *If* $b_1 \neq 3$, $C_{MH1}/C^* < \frac{53}{35}$.

Proof. In this case, algorithm $MH1$ is just $H1$, hence $C_{MH1} = C_1$. Chang and Lee [2] proved that $C_1/C^* \leqslant \frac{5}{3}$. In order to obtain $C_1/C^* < \frac{53}{35}$, more careful analysis are necessary.

From Lemma 1, we have

$$C^* = \max\{y + b^*T, P + T\}. \tag{1}$$

If $C_1 = P + T$, we have $C_1 = C^*$ clearly, and we are done. Hence, we suppose that $C_1 > P + T$ in the following. Then by Lemma 1,

$$C_1 = P_1 + b_1 T \quad \text{and} \quad P_1 < T. \tag{2}$$

It is obvious that $b^* \geqslant 2$. Otherwise, $b_1 = b^* = 1$ and $MH1$ yields an optimal solution. If $b_1 \leqslant b^*$, by (1) and (2), we have

$$\frac{C_1}{C^*} \leqslant \frac{P_1 + b_1 T}{y + b^* T} < \frac{P_1 + b^* T}{b^* T} = 1 + \frac{P_1}{T} \cdot \frac{1}{b^*} < 1 + \frac{1}{b^*} \leqslant \frac{3}{2} < \frac{53}{35}. \tag{3}$$

Hence, we only need to consider the case that $b_1 > b^*$.

Noting that the jobs are assigned to batches according to algorithm *FFD* in $H1$, by Lemma 2(2), we have

$$b_1 \leqslant \frac{11}{9} b_L^* + 1 \leqslant \frac{11}{9} b^* + 1. \tag{4}$$

If $b^* = 2$, then $b_1 \leqslant \frac{31}{9} < 4$. From $b_1 > b^*$ we know $b_1 = 3$, contradicting the Lemma's assumption $b_1 \neq 3$. Therefore, we suppose $b^* \geqslant 3$ in the following. To obtain the desired worst-case ratio, we distinguish two cases according to (1).

Case 1: $C^* = y + b^*T$. Then (1) implies $y + b^*T \geqslant P + T$, i.e., $P \leqslant y + (b^* - 1)T$. Recall that $P_1 \leqslant P_2 \leqslant \cdots \leqslant P_{b_1}$. We establish that $P_1 \leqslant P/b_1 \leqslant (y + (b^* - 1)T)/b_1$, and thus

$$\begin{aligned}\frac{C_1}{C^*} &= \frac{P_1 + b_1T}{y + b^*T} \leqslant \frac{((y + (b^* - 1)T)/b_1) + b_1T}{y + b^*T} \\ &= \frac{1}{b_1} \cdot \frac{y + b^*T + (b_1^2 - 1)T}{y + b^*T} \\ &= \frac{1}{b_1} + \frac{1}{b_1} \cdot \frac{(b_1^2 - 1)T}{y + b^*T} \\ &< \frac{1}{b_1} + \frac{1}{b_1} \cdot \frac{b_1^2 - 1}{b^*}. \end{aligned} \tag{5}$$

If $b^* = 3$, (4) states that $b_1 \leqslant \frac{42}{9} < 5$. Combing it with $b_1 > b^*$, we have $b_1 = 4$. Then from (5), it follows that $C_1/C^* < \frac{3}{2}$.

Similarly, if $b^* = 4$, then $b_1 = 5$ and thus $C_1/C^* < \frac{7}{5}$; if $b^* = 5$, then $b_1 = 6, 7$ and thus $C_1/C^* < \frac{4}{3}$ (for $b_1 = 6$) or $C_1/C^* < \frac{53}{35}$ (for $b_1 = 7$); if $b^* = 6$, then $b_1 = 7, 8$, and thus $C_1/C^* < \frac{9}{7}$ (for $b_1 = 7$) or $C_1/C^* < \frac{23}{16}$ (for $b_1 = 8$).

If $b^* \geqslant 7$, (4) implies that $b^* \geqslant (9(b_1 - 1))/11$. Substituting it into (5), we obtain

$$\frac{C_1}{C^*} < \frac{1}{b_1} + \frac{1}{b_1} \cdot \frac{(b_1^2 - 1)}{9(b_1 - 1)/11} = \frac{11}{9} + \frac{20}{9b_1} < \frac{11}{9} + \frac{20}{9} \cdot \frac{1}{8} = \frac{3}{2}, \tag{6}$$

where the last inequality is from $b_1 > b^* \geqslant 7$.

Case 2: $C^* = P + T$. Then (1) implies $C^* = P + T \geqslant y + b^*T > b^*T$. Combining it with (4), we have $P > (b^* - 1)T \geqslant (\frac{9}{11}b_1 - \frac{20}{11})T$. As $P_1 \leqslant P/b_1$, we conclude that

$$\begin{aligned}\frac{C_1}{C^*} &= \frac{P_1 + b_1T}{P + T} \leqslant \frac{(P/b_1) + b_1T}{P + T} \\ &= \frac{1}{b_1} \cdot \frac{P + T + (b_1^2 - 1)T}{P + T} \\ &= \frac{1}{b_1} + \frac{b_1^2 - 1}{b_1} \cdot \frac{T}{P + T} < \frac{1}{b_1} + \frac{b_1^2 - 1}{b_1} \cdot \frac{T}{b^*T} \\ &= \frac{1}{b_1} + \frac{b_1^2 - 1}{b_1} \cdot \frac{1}{b^*}. \end{aligned} \tag{7}$$

Note that (7) is the same as (5). Therefore, the same arguments as those in Case 1 can complete the proof. □

Lemma 7. *If* $b_1 = 3$, $C_{MH1}/C^* < \frac{53}{35}$.

Proof. Similarly, we can suppose that $b^* \geqslant 2$. If $C_1 = P + T$ or $C_2 = P + T$, then $\min\{C_1, C_2\} = C^*$ by (1). Hence, we suppose that $C_1 > P + T$ and $C_2 > P + T$. Then by Lemma 1, $P_1 < T$ and $P_1' < T$, where P_1 and P_1' are the total processing times of jobs in the first batches in σ_1 and σ_2, respectively.

If $b_1 \leqslant b^*$, we have shown in Lemma 6 that $C_1/C^* < \frac{3}{2}$ (see the proof of (3)). Hence, we suppose $b_1 > b^*$. Then $b^* = 2$. Hence, by Lemma 2(1), we have $b_2 \leqslant \frac{7}{4} * 2 = \frac{7}{2}$, that is, $b_2 \leqslant 3$. If further $b_2 \leqslant b^*$, by similar arguments to show (3) in the proof of Lemma 6, we can obtain $C_2/C^* \leqslant \frac{3}{2}$. Hence, we suppose $b_2 > b^*$. Combining it with $b_2 \leqslant 3$ and $b^* = 2$, we know that $b_2 = 3$.

$b^* = 2$ and (1) states that $C^* = \max\{y + 2T, P + T\}$. Two cases are considered as follows.

Case 1: $C^* = y + 2T$. Then $y + 2T \geqslant P + T$, and thus $P \leqslant y + T$. From Lemmas 4 and 5, we get $P_3' \geqslant \frac{33}{35}P^* \geqslant \frac{33}{35}(P - y)$. Noting that $P_1' \leqslant P_2' \leqslant P_3'$, we obtain

$$P_1' \leqslant \frac{P - P_3'}{2} \leqslant \frac{P - (33/35)(P - y)}{2} = \frac{2P + 33y}{70} \leqslant \frac{2(y + T) + 33y}{70} = \frac{y}{2} + \frac{T}{35}. \tag{8}$$

Therefore,

$$\frac{C_2}{C^*} = \frac{P_1' + 3T}{y + 2T} \leqslant \frac{y/2 + (T/35) + 3T}{y + 2T} = \frac{(1/2)(y + 2T) + (2 + 1/35)T}{y + 2T} < \frac{1}{2} + \frac{(2 + 1/35)T}{2T} = \frac{53}{35}. \quad (9)$$

Case 2: $C^* = P + T$. Then $P + T \geqslant y + 2T$, and thus $P \geqslant y + T$. From Lemmas 4 and 5, we know $P_3' \geqslant \frac{33}{35} P^* \geqslant \frac{33}{35}(P - y) > \frac{33}{35}T$. By $P_1' \leqslant P_2' \leqslant P_3'$ and $P = \sum_{i=1}^{3} P_i'$, we have $P_1' \leqslant \frac{1}{2}(P - P_3') < \frac{1}{2}(P - \frac{33}{35}T)$. Therefore,

$$\begin{aligned}\frac{C_2}{C^*} &= \frac{P_1' + 3T}{P + T} < \frac{(1/2)(P - (33/35)T) + 3T}{P + T} \\ &= \frac{(1/2)(P + T) + (5/2 - (33/70))T}{P + T} \\ &< \frac{1}{2} + \frac{(5/2 - (33/70))T}{2T} = \frac{53}{35}. \quad \square\end{aligned}$$

Theorem 8. $C_{MH1}/C^* < \frac{53}{35}$.

Proof. This is a direct conclusion of Lemmas 6 and 7. $\square$

Since both *FFD*, *FF* and $H1$ run in time $O(n \log n)$, and the *FPTAS* of Lawler or Kellerer and Pferschy for the knapsack problem runs in time $O(n)$ when we take $\varepsilon = \frac{2}{35}$, the time complexity of $MH1$ is $O(n \log n)$, the same as that of $H1$.

3. Problem 1, $h_1||\sum C_i$

This section addresses the problem 1, $h_1||\sum C_i$. To present our improved algorithm, we will propose a local search procedure, called k, k-*exchange procedure*, which is an extension of 2-OPT procedure proposed in [6].

Using the same notations as those in [5,6], denote by S^* and S an optimal schedule and the schedule yielded by algorithm *SPT* (*shortest processing time first*), respectively. Denote by B the set of the jobs scheduled before the maintenance period in S, and by A the set of the remaining jobs scheduled after it. Denote by X the set consisting of the $|B|$ jobs scheduled first in S^*, and by Y the set of the remaining $|A|$ jobs scheduled last. Denote by δ^* and δ the idle times on the machine before the maintenance period, respectively, in the schedules S^* and S.

Definition 9. Let $\bar{a} \leqslant |A|$, $\bar{b} \leqslant |B|$ and k are positive integers satisfying $k \geqslant \bar{b} \geqslant \bar{a}$. An k, k-exchange procedure is an exchange of $\bar{a}$ jobs in A with $\bar{b}$ jobs of B in the schedule S, under the constraint that the total processing times of $\bar{b}$ jobs in B plus δ is no less than the total processing times of $\bar{a}$ jobs in A. After exchange, the jobs are reordered before and after the maintenance in non-decreasing order of their processing times.

Obviously an k, k-exchange procedure is essentially a post-optimization of the *SPT* schedule using local search method. With this procedure, our improved algorithm, denoted by *SPTE* can be formulated as follows.

Algorithm *SPTE*. (1) Process all the jobs according to the *SPT* rule.

(2) For a given positive integer k, try all k, k-exchange procedures to generate new schedules.

(3) Choose the best one from the schedules generated in Steps 1 and 2 as output.

Clearly, by setting $k = 1$, the above algorithm becomes *MSPT*, which was proposed in [6]. Hence *SPTE* is a generalization of *MSPT*. Sadfi et al. showed that *MSPT* has a worst-case ratio of $\frac{20}{17}$. Hence we assume that $k \geqslant 2$ in the following. We will show that *SPTE* is a *PTAS*. It shows that local search method is powerful for the considered problem.

Denote by S' the schedule yielded by algorithm *SPTE*. With straightforward notation, B' and A' represent the job partition of S'. Finally, we denote by C_i, C_i' and C_i^* the completion times of job J_i in schedules S, S' and S^*, respectively.

The following Lemmas 10 and 11 are cited from in [6], and Lemma 12 is parallel to Lemma 4 of that paper which can be shown similarly.

Lemma 10 (*Sadfi et al. [6]*). $\delta \geqslant \delta^*$.

Lemma 11 (*Sadfi [6]*). *If* (*at least*) *one job of X is scheduled after the maintenance period in the optimal solution, then*

$$\sum_{i=1}^{n} C_i' \leqslant \sum_{i=1}^{n} C_i^* + (|Y| - 2)(\delta - \delta^*). \tag{10}$$

Lemma 12. *If* (*at least*) $k + 1$ *jobs of B are scheduled after the maintenance period in the optimal solution, then*

$$\sum_{i=1}^{n} C_i^* \geqslant \left(\frac{|Y|(|Y| + 1)}{2} + k + 1\right)(\delta - \delta^*). \tag{11}$$

Theorem 13. *For any given integer* $k \geqslant 2$, *algorithm SPTE has a worst-case ratio of at most* $1 + 2/(5 + 2\sqrt{2k + 8})$, *and runs in* $O(n^{2k+1})$. *Therefore, SPTE is a PTAS for the problem* $1, h_1 || \sum C_i$.

Proof. We first prove (10) and (11). Two cases are considered as follows.

Case 1: No k, k-exchange procedure exists. Then the number of jobs from A should be no less than the number of B, and $S' = S$. If $B = X$, *SPT* schedule is optimal. If $B \neq X$, in order to process some job(s) from A before the maintenance period, we have to remove at least $k + 1$ jobs from B. Hence if S is not optimal, then at least one job from X is processed after the maintenance period in S^*. Hence the condition of Lemma 11 is satisfied, and thus (10) is true. Furthermore, at least $k + 1$ jobs from B are processed after the maintenance period in S^*, which states the condition of Lemma 12 is satisfied. It follows that (11) is true.

Case 2: k, k-exchange procedures generate new schedules. Without loss of generality, we suppose that S^* cannot be generated by these procedures.

Suppose that the optimal schedule S^* can be generated by exchanging b' jobs from B with a' jobs from A, where $|A| \geqslant a' \geqslant 1$ and $|B| \geqslant b'$. Since S^* cannot be generated by any k, k-exchange procedure, we have $b' > k$. It states that the condition of Lemma 12 is satisfied. Hence, (11) is true.

Since the processing time of any job from A is no less than that of any job from B, $b' \geqslant a'$. We distinguish two subcases according to this inequality.

Subcase 1: $b' > a'$. Since S^* can be generated by exchanging b' jobs from B with a' jobs from A, the number of jobs processed before the maintenance period in S^* must be $|B| - b' + a' < |B|$. Since $|X| = |B|$, we conclude that at least one job from X is scheduled after the maintenance period in S^*. Eq. (10) follows.

Subcase 2: $b' = a'$. Then $a' > k \geqslant 2$. For this subcase, the condition of Lemma 11 may not be satisfied, but we show that (10) is still true assuming all jobs in X are processed before the maintenance period.

Let q_A and q_B be the total processing time of the biggest k jobs from A in X and the total processing time of the biggest k jobs from B in Y, respectively. Let W_A and W_B be the sets of the other $a' - k$ jobs of $X \cap A$ and $Y \cap B$, respectively. Moreover, let p_A and p_B be the sums of their processing times. Note that $p_A \geqslant p_B$ by the construction of the schedule S. Now we construct a schedule S'' from S^* by exchanging the jobs of W_A and W_B and processing jobs in *SPT* order before and after the maintenance period. Let δ'' be the idle time on the machine before the maintenance period, X'' be the set consisting of the $|B|$ jobs scheduled first in S'', and Y'' be the set of the remaining $|A|$ jobs scheduled last. Let $\Delta = p_A - p_B \geqslant 0$ for short. Denote by C_i'' the completion times of job J_i in S''.

By comparing schedules S'' and S^* (see Figs. 1 and 2), we have

$$C_i^* \geqslant C_i'', \quad 1 \leqslant i \leqslant |B| - k,$$

and

$$C_i^* \geqslant C_i'' + \Delta, \quad |B| - k + 1 \leqslant i \leqslant |B|,$$

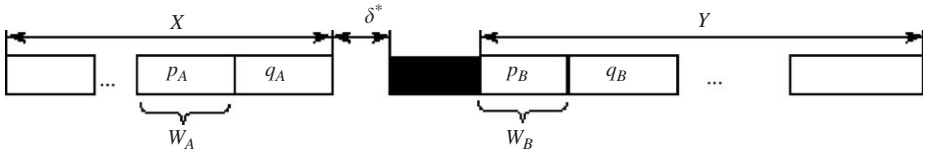

Fig. 1. Schedule S^*.

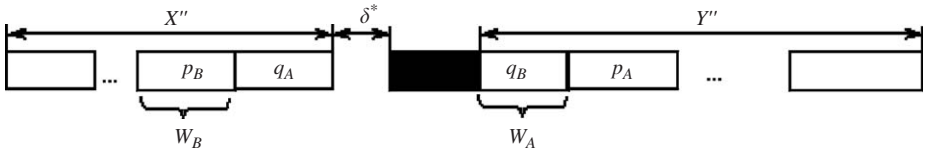

Fig. 2. Schedule S''.

where C_i^* and C_i'' are the completion times of jobs at position i in S^* and S'', respectively. It follows that

$$\sum_{i=1}^{|B|} C_i'' \leqslant \sum_{i=1}^{|B|} C_i^* - k\Delta. \tag{12}$$

On the other hand,

$$C''_{|B|+a'} - C^*_{|B|+a'} = (R + L + q_B + p_A) - (R + L + p_B + q_B) = \Delta.$$

It implies that

$$C_i'' \leqslant C_i^* + \Delta, \quad |B| + 1 \leqslant i \leqslant |B| + a' - 1. \tag{13}$$

and

$$C_i'' = C_i^* + \Delta, \quad |B| + a' + 1 \leqslant i \leqslant n. \tag{14}$$

Summing (12)–(14), and by $|Y| = n - |B|$, we obtain

$$\sum_{i=1}^{n} C_i'' \leqslant \sum_{i=1}^{n} C_i^* + (|Y| - k)\Delta \leqslant \sum_{i=1}^{n} C_i^* + (|Y| - 2)\Delta.$$

Since $p_A + q_A + \delta^* = p_B + q_A + \delta''$, we have $\Delta = \delta'' - \delta^* \leqslant \delta - \delta^*$ (due to $\delta'' \leqslant \delta$). Hence

$$\sum_{i=1}^{n} C_i'' \leqslant \sum_{i=1}^{n} C_i^* + (|Y| - 2)(\delta - \delta^*).$$

Since *SPTE* outputs the best schedule among all k, k-exchange procedures and S'' can be generated by an k, k-exchange procedure, we have

$$\sum_{i=1}^{n} C_i' \leqslant \sum_{i=1}^{n} C_i^* + (|Y| - 2)(\delta - \delta^*),$$

which is just (10).

Now we are ready to get the worst-case ratio. By (10) and (11), we obtain

$$\frac{\sum_{i=1}^{n} C_i' - \sum_{i=1}^{n} C_i^*}{\sum_{i=1}^{n} C_i^*} \leqslant \frac{2(|Y| - 2)}{|Y|(|Y| + 1) + 2k + 2}.$$

Define

$$f(|Y|) = \frac{2(|Y|-2)}{|Y|(|Y|+1)+2k+2}, \quad |Y| > 0.$$

By taking a derivation of $f(|Y|)$, we can see that it is increasing for $|Y| \leqslant 2+\sqrt{2k+8}$ and is decreasing for $|Y| \geqslant 2+\sqrt{2k+8}$ and hence reaches a maximum at $|Y| = 2+\sqrt{2k+8}$ with

$$f(2+\sqrt{2k+8}) = \frac{2}{5+2\sqrt{2k+8}}.$$

Hence, the worst-case ratio of *SPTE* is at most $1+2/(5+2\sqrt{2k+8})$.

It is obvious that there are at most $O(n^{2k})$ k, k-exchange procedures, and computing the objective function value of a schedule takes $O(n)$ time. Hence *SPTE* runs in time $O(n^{2k+1})$, and is a *PTAS* for $1, h_1||\sum C_i$. □

References

[1] I. Adiri, J. Bruno, E. Frostig, A.H.G. Rinnooy Kan, Single machine flow-time scheduling with a single breakdown, Acta Inform. 26 (1989) 679–696.

[2] Y.C. Chang, C.Y. Lee, Machine scheduling with job delivery coordination, European J. Oper. Res. 158 (2004) 470–487.

[3] H. Kellerer, U. Pferschy, A New Fully Polynomial Approximation Scheme for the Knapsack Problem, Lecture Notes in Computer Science, Vol. 1444, 1998, 123–134.

[4] E. Lawler, Fast approximation algorithms for knapsack problems, Math. Oper. Res. 4 (1979) 339–356.

[5] C.Y. Lee, S.D. Liman, Single machine flow-time scheduling with scheduled maintance, Acta Inform. 29 (1992) 375–382.

[6] C. Sadfi, B. Penz, C. Rapine, J. Błazewicz, P. Formanowicz, An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints, European J. Oper. Res. 161 (2005) 3–10.

[7] D. Simchi-Levi, 1994, New worst-case results for the bin packing problem, Naval Res. Logist. 41 (1994) 579–585.

[8] M. Yue, A simple proof of the inequality $FFD(L) \leqslant \frac{11}{9}OPT(L)+1$ $\forall L$, for the *FFD* bin-packing algorithm, Acta Math. Appl. Sin. 7 (1991) 321–331.