

Competitive Agent Scheduling with Controllable Processing Times

Guohua Wan

College of Management, Shenzhen University, Shenzhen 518060, China, gh_wan@china.com

Joseph Y.-T. Leung

Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, U.S.A.,
leung@oak.njit.edu

Michael Pinedo

Stern School of Business, New York University, 44 West Fourth Street, New York, NY 10012, U.S.A.,
mpinedo@stern.nyu.edu

We consider several competitive agent scheduling problems with controllable processing times, where two agents A and B compete for a single machine to process their jobs. The objective function for agent B is always the same, namely f_{\max} . Several different objective functions are considered for agent A, including the total compression cost subject to deadline constraints (the imprecise computation model), the total flow time plus compression cost, the maximum tardiness plus compression cost and the maximum lateness plus compression cost. These problems have various applications in computer systems as well as in operations management. We provide NP-hardness proofs for the more general problems and polynomial time algorithms for several special cases of the problems.

Keywords: Agent Based Scheduling, Single Machine, Controllable processing times, Availability constraints, Imprecise computation, Total flow time; Maximum tardiness, Maximum lateness.

1 Introduction

We consider several competitive agent scheduling problems where two sets of jobs N_1 and N_2 (belonging to agents A and B, respectively) have to be processed on a single machine. Agent A has to schedule n_1 jobs in N_1 and agent B has to schedule n_2 jobs in N_2 . Let n denote the total number of jobs, i.e., $n = n_1 + n_2$. The processing time, release date and due date of job $j \in N_1$ (N_2) are denoted by p_j^a , r_j^a and d_j^a (p_j^b , r_j^b and d_j^b), respectively. Unlike the classical model in deterministic scheduling where all processing times are fixed and known in advance, the processing times are in what follows controllable and can be chosen by the decision maker. In this paper, we assume that agent A's jobs are controllable, while agent B's jobs are not. Formally, for each job $j \in N_1$, there is a maximum value of the processing time \bar{p}_j^a which can be compressed to a minimum value \underline{p}_j^a ($\underline{p}_j^a \leq \bar{p}_j^a$). Compressing \bar{p}_j^a to some actual processing time $p_j^a \in [\underline{p}_j^a, \bar{p}_j^a]$ may decrease the job completion time, but incurs an additional cost $w_j^a x_j^a$, where $x_j^a = \bar{p}_j^a - p_j^a$ is the amount of compression of job $j \in N_1$ and w_j^a is the compression cost per unit time. The total compression cost is represented by a linear function $\sum_{j \in N_1} w_j^a x_j^a$. We consider the optimization problem in which the value of the objective function of agent A has to be minimized, while the value of the objective function of agent B must be kept at less than or equal to a fixed value Q .

The classical notation for machine scheduling is based on a triplet $\alpha | \beta | \gamma$. Agnetis et al. (2004) extend this notation for the two agent problem to $\alpha | \beta | \gamma^a : \gamma^b$. Their optimization problems can be described as follows: Given that agent B keeps the value of its objective function γ^b less than or equal to Q , agent A has to minimize the value of its objective function γ^a . In this paper, we may assume that either one set or both sets of jobs have different release dates and that

either one set or both sets of jobs are subject to preemptions. If the jobs of both agents are subject to the same processing restrictions and constraints (as in Agnetis et al. (2004)), then the single machine problem will be referred to as $1 | \beta | \gamma^a : \gamma^b$. If the processing restrictions and constraints of agent A's jobs are different from the processing restrictions and constraints of agent B's jobs, we refer to the single machine problem as $1 | \beta^a : \beta^b | \gamma^a : \gamma^b$.

Scheduling models with competitive agents have already received some attention in the literature. Baker and Smith (2003) and Agnetis et al. (2004) consider single machine scheduling problems with two agents in which all jobs of the two sets are released at time 0 and both sets of jobs are subject to the same processing restrictions and constraints. The objective functions considered in their research include the total weighted completion time ($\sum w_j C_j$), the number of tardy jobs ($\sum U_j$) and the maximum of regular functions (f_{\max}). Leung et al. (2006) consider a scheduling environment with $m \geq 1$ identical machines in parallel and two agents, and generalize the results of Baker and Smith (2003) and Agnetis et al. (2004) by including the total tardiness objective, allowing for preemptions, and considering jobs with different release dates.

Scheduling models with controllable processing times have received considerable attention in the literature, see, for example, the survey by Nowicki and Zdrzalka (1990). The study of these models is motivated by their various applications to production and operations management, computer systems, among others. The main issue here is to establish a trade-off between job completion times and the costs of compression.

Studies of scheduling problems with controllable processing times were initiated by Vickson (1980a, 1980b). The problem introduced by Vickson has attracted a lot of attention, see, for example, Hoogeveen and Woeginger (2002), Janiak et al. (2005), Wan et al. (2001), Janiak and Kovalyov (1996), Nowicki and Zdrzalka (1990, 1995), Shakhlevich and Strusevich (2005), and van Wassenhove and Baker (1982), among others. In the triplet notation, we put *ctrl* in the second field to refer to controllable processing times.

In the following, we first state the problems, then we give NP-hardness proofs of the problems, and describe algorithms for solving the problems or their special cases. We conclude and discuss some future research directions in the last section.

2 Problem Description

In the scheduling problems considered in this paper, each job j of agent B has a penalty function $f_j^b(C_j^b)$ and the objective function of agent B is simply $f_{\max}^b = \max(f_1^b(C_1^b), \dots, f_{n_2}^b(C_{n_2}^b))$. Given that agent B keeps the value of f_{\max}^b less than or equal to Q , agent A has to minimize the value of one of the following objective functions.

(1) In the first problem each job j of agent A must meet a deadline \bar{d}_j^a . Our goal is to determine the actual processing times of agent A's jobs so that the total compression cost is minimized. It is referred to as imprecise computation model with controllable processing times (see Leung (2004)). Using the notation introduced above, we denote the problems by $1 | ctrl^a, r_j^a, \bar{d}_j^a, pmtn^a : pmtn^b | \sum w_j^a x_j^a : f_{\max}^b$ and $1 | ctrl^a, r_j^a, \bar{d}_j^a, pmtn^a : \circ^b | \sum w_j^a x_j^a : f_{\max}^b$. We also consider the case where all the jobs of both agents A and B are released at time 0; i.e., $1 | ctrl^a, \bar{d}_j^a, pmtn^a : \circ^b | \sum w_j^a x_j^a : f_{\max}^b$.

(2) The second problem considered is to minimize the total flow time plus job compression costs. Again, the jobs of agent A may be preempted. We shall show that when the jobs of agent A have different release dates, the problem is unary NP-hard; i.e., the problem $1 | ctrl^a, r_j^a, pmtn^a : \circ^b | \sum (C_j^a + w_j^a x_j^a) : f_{\max}^b$ is unary NP-hard. However, if the jobs of agent A all have the same release date, then the complexities of the nonpreemptive and preemptive cases are the same; i.e., the problem $1 | ctrl^a : \circ^b | \sum (C_j^a + w_j^a x_j^a) : f_{\max}^b$ and the problem $1 | ctrl^a, pmtn^a : \circ^b | \sum (C_j^a + w_j^a x_j^a) :$

f_{max}^b have the same complexity. Although we do not know the complexity of these two problems, we are able to give a polynomial-time algorithm for the special case when the jobs have agreeable weights, i.e., when $p_1^a \leq p_2^a \leq \dots \leq p_{n_1}^a$ and $w_1^a \leq w_2^a \leq \dots \leq w_{n_1}^a$.

(3) The third and the fourth problem concern the minimization of the maximum tardiness plus job compression cost and the minimization of the maximum lateness plus job compression cost, respectively. The jobs of agent A may again be preempted. If the jobs of agent A have arbitrary release dates, then both problems are unary NP-hard; i.e., the problems $1 \mid ctrl^a, r_j^a, pmtn^a : \circ^b \mid (T_{max} + \sum w_j^a x_j^a) : f_{max}^b$ and $1 \mid ctrl^a, r_j^a, pmtn^a : \circ^b \mid (L_{max} + \sum w_j^a x_j^a) : f_{max}^b$ are both unary NP-hard. When the jobs of agent A have the same release date, then the complexities of the nonpreemptive and preemptive cases are identical. Although we do not know the complexity of these two problems, we are again able to provide a polynomial-time algorithm for the special case when the jobs have agreeable weights, i.e., when $d_1^a \leq d_2^a \leq \dots \leq d_{n_1}^a$ and $w_1^a \leq w_2^a \leq \dots \leq w_{n_1}^a$.

The problems described above may find various applications in computer systems as well as in operations management. For instance, in computer networks a server may service several classes of jobs such as file downloading, voice messaging and web browsing, where one class of jobs may have a high priority and another class may have a lower priority. A request to the server for voice messaging or web browsing, constitutes a job. Jobs may have various characteristics such as release dates, due dates, and/or preemption. The server may put the jobs into two classes, say, one for web browsing and the other for voice messaging. In order to provide a satisfactory quality of service, it is necessary to keep on the one hand the maximum penalty of jobs for web browsing less than or equal to some fixed value, and, on the other hand, meet the deadlines of the voice messaging packages. To keep the voice quality at a satisfactory level, it is desirable to discard as few packages as possible, i.e., to minimize the total amount of compression of jobs for voice messaging. This application can be modeled by (1).

3 Imprecise Computation

We first consider the problem $1 \mid ctrl^a, r_j^a, \bar{d}_j^a, pmtn^a : \circ^b \mid \sum w_j^a x_j^a : f_{max}^b$. We show that it is unary NP-hard via a reduction from 3-PARTITION, which is known to be unary NP-hard (see Garey and Johnson (1979)).

3-PARTITION: Given positive integers a_1, \dots, a_{3n} and b with $\frac{b}{4} < a_j < \frac{b}{2}, j = 1, \dots, 3n$ and $\sum_{j=1}^{3n} a_j = nb$, do there exist n pairwise disjoint three element subsets $S_i \subset \{1, \dots, n\}$ such that $\sum_{j \in S_i} a_j = b, i = 1, \dots, n$?

Theorem 1. The problem $1 \mid ctrl^a, r_j^a, \bar{d}_j^a, pmtn^a : \circ^b \mid \sum w_j^a x_j^a : f_{max}^b$ is unary NP-hard.

Proof. The proof is done via a reduction from 3-PARTITION. \square

However, if preemptions are allowed for the jobs of agent B, then the problem is solvable in polynomial time. That is, $1 \mid ctrl^a, r_j^a, \bar{d}_j^a, pmtn^a : pmtn^b \mid \sum w_j^a x_j^a : f_{max}^b$ is solvable in polynomial time. The algorithm is based on the polynomial-time algorithm for minimizing the total weighted error in the imprecise computation model that is due to Leung et al. (1994). The algorithm of Leung-Yu-Wei solves the problem $1 \mid r_j, \bar{d}_j, pmtn \mid \sum w_j x_j$ in $O(n \log n + kn)$ time, where n is the number of jobs and k is the number of distinct values of $\{w_j\}$.

Algorithm 1.

Step 1: For each job j of agent B, compute a “deadline” \bar{d}_j^b via $f_{max} \leq Q$ (assuming f^{-1} can be computed in constant time). Let the release date of job j of agent B be $r_j^b = 0$ and the weight be

$w_j^b = 1 + \max_{1 \leq j \leq n_1} \{w_j^a\}$, for $j = 1, \dots, n_2$. Furthermore, let the jobs of agent B be uncompressible (i.e., $\bar{p}_j^b = \underline{p}_j^b$).

Step 2: Use the algorithm of Leung-Yu-Wei to generate a schedule of all jobs of agents A and B together.

Remark 1. The time complexity of Algorithm 1 is $O((n_1 + n_2) \log(n_1 + n_2) + (k + 1)(n_1 + n_2))$, where k is the number of distinct weights of jobs of agent A.

Remark 2. Algorithm 1 can be generalized to solve the problem $1 \mid ctrl^a, r_j^a, \bar{d}_j^a, pmtn^a : r_j^b, pmtn^b \mid \sum w_j^a x_j^a : f_{max}^b$ as well. In Step 1 of Algorithm 1, we simply let the release date of job j of agent B be r_j^b .

Remark 3. Algorithm 1 can be generalized to solve the problem $1 \mid ctrl^a, \bar{d}_j^a, pmtn^a : \circ^b \mid \sum w_j^a x_j^a : f_{max}^b$ as well. This is because of Property 1 shown below.

Property 1. If $r_j^a = 0, \forall j$, then there exists an optimal schedule in which jobs of agent B are scheduled as late as possible, i.e., against their “deadlines” (computed via $f_{max}^b \leq Q$).

Proof. First, note that in an optimal schedule, all the jobs of agent B must satisfy $f_{max}^b \leq Q$. Suppose that for a job k of agent B, $f_k^b < Q$, then we can always move it backwards so that $f_k^b = Q$ (or reaching another job of agent B) and at the same time move some pieces of jobs of agent A forward. Clearly, this will not violate the constraint $f_{max}^b \leq Q$ and it will not increase $\sum w_j^a x_j^a$. \square

Using Property 1, we can develop our algorithm as follows. We schedule all the jobs of both agents together using the Leung-Yu-Wei algorithm. Although in the schedule created jobs of agent B may be preempted, we can always combine the pieces of a job of agent B together and move it towards its “deadline” \bar{d}_j^b , by Property 1. In this way, we can always convert a schedule where jobs of agent B may be preempted into one where jobs of agent B are not preempted.

Remark 4. Algorithm 1 can be generalized to solve the problem $1 \mid ctrl^a, \bar{d}_j^a : \circ^b \mid \sum w_j^a x_j^a : f_{max}^b$ as well. We first solve the problem $1 \mid ctrl^a, \bar{d}_j^a, pmtn^a : \circ^b \mid \sum w_j^a x_j^a : f_{max}^b$. We then merge the preempted pieces of jobs of agent A by moving the jobs of agent B forward. Clearly, this will not violate the constraints of the jobs of either agent.

4 Total flow time plus compression cost

We now turn to the problem $1 \mid ctrl^a, r_j^a, pmtn^a : \circ^b \mid \sum (C_j^a + w_j^a x_j^a) : f_{max}^b$. We will show that it is unary NP-hard via a reduction from 3-PARTITION.

Theorem 2. The problem $1 \mid ctrl^a, r_j^a, pmtn^a : \circ^b \mid \sum (C_j^a + w_j^a x_j^a) : f_{max}^b$ is unary NP-hard.

Proof. The proof is done via a reduction from 3-PARTITION. \square

We now consider the problem when the jobs of agent A are all released at time 0. At present, the complexity of this problem is not known. However, we do know that the complexities of the nonpreemptive case and the preemptive case are the same; i.e., the problem $1 \mid ctrl^a, pmtn^a : \circ^b \mid \sum (C_j^a + w_j^a x_j^a) : f_{max}^b$ and the problem $1 \mid ctrl^a : \circ^b \mid \sum (C_j^a + w_j^a x_j^a) : f_{max}^b$ have the same complexity. This is due to Property 1 described in Section 3.

Although the complexity of the general case is not known, we are able to provide a polynomial-time algorithm for the special case with agreeable weights, i.e., when $p_1^a \leq p_2^a \leq \dots \leq p_{n_1}^a$ and $w_1^a \leq w_2^a \leq \dots \leq w_{n_1}^a$.

We use the following terminology in the algorithm. Job j of agent A is referred to as “compressed” if $p_j^a = \underline{p}_j^a$, i.e., it is fully compressed and cannot undergo any additional compression. Job j of agent A is referred to as “uncompressed” if $p_j^a > \underline{p}_j^a$; it can undergo some (additional) compression. Job 0 with $p_0^a = 0$ is a dummy job that belongs to Agent A and that is scheduled all the way at the beginning. Job 0 is considered “compressed”.

Algorithm 2. (A polynomial-time algorithm for a special case)

Step 1: For each job j of agent B, compute its “deadline” \bar{d}_j^b via $f_{\max}^b \leq Q$. Starting from $\max_{1 \leq j \leq n_2} \{\bar{d}_j^b\}$, schedule the jobs of agent B backwards so that each job is scheduled as close to its deadline as possible.

Step 2: Define block i as the i^{th} set of contiguously processed jobs of agent B. Let there be $k_1 \leq n_2$ blocks, and let h_i be the length of block i , $1 \leq i \leq k_1$.

Step 3: For each job j of agent A, let $p_j^a = \bar{p}_j^a$. Let $p_0^a = 0$.

Step 4: Let $P = \{p_j^a \mid 1 \leq j \leq n_1\}$. (P is the current set of processing times of agent A’s jobs.) Set $Cost = 0$ and $Benefit = 0$.

Step 5: Schedule the jobs of agent A by the preemptive SPT rule, skipping the time slots occupied by jobs of agent B. Let l_i , $1 \leq i \leq k_2$, be the length of the job piece of agent A immediately following block i . Note that $k_2 \leq k_1$.

Step 6: Let z be the first “uncompressed” job; i.e., the first $z - 1$ jobs are fully compressed. If every job is “compressed” then go to Step 9. Let q be the first block of agent B’s jobs that appears after the start time of job z . Let $l_{\min} = \min_{q \leq i \leq k_2} \{l_i\}$. Let $x_z = p_z^a - \underline{p}_z^a$ and $y_z = p_z^a - p_{z'}^a$, where z' is the largest index such that $p_{z'}^a < p_z^a$. Let $\delta = \min\{x_z, y_z, l_{\min}\}$.

Step 7: Compress job z by an amount δ ; i.e., $p_z^a = p_z^a - \delta$. We consider the following three cases, in the order they are presented.

[Case 1.] ($\delta = l_{\min}$) $Cost = Cost + w_z * \delta$, $Benefit = Benefit + (n_1 - z + 1) * \delta + \sum_{q \leq j \leq k_2} u_j h_j$, where u_j is the number of jobs that jump over block j (meaning that they complete after block j , before compressing job z and complete before block j afterwards).

[Case 2.] ($\delta = x_z$) $Cost = Cost + w_z * \delta$, $Benefit = Benefit + (n_1 - z + 1) * \delta$, mark job z as “compressed”.

[Case 3.] ($\delta = y_z$) $Cost = Cost + w_z * \delta$, $Benefit = Benefit + (n_1 - z + 1) * \delta$. Reindex the jobs of agent A in ascending order of their processing times, where job z will appear before any job with the same processing time as job z .

Step 8: If $Cost < Benefit$, then go to Step 4 else go to Step 5.

Step 9: Using P (which has the last recorded processing times of the jobs), we schedule the jobs of agent A by the preemptive SPT rule, skipping the time slots occupied by jobs of agent B. This is the final schedule. \square

Theorem 3. Algorithm 2 solves the problem $1 \mid ctrl^a, pmtn^a : o^b \mid \sum(C_j^a + w_j^a x_j^a) : f_{\max}^b$ with agreeable weights in $O(n_2(n_1 + n_2)n_1^2 \log n_1)$ time.

Proof. Since the weights are agreeable, the algorithm always compresses the first compressible job before it proceeds to compress the next compressible job. Thus it maintains the agreeability of the remaining compressible jobs, until all the compressible jobs are compressed (provided the compressions are beneficial). This guarantees the correctness of the algorithm.

Furthermore, note that in the above algorithm, Steps 1 and 2 take at most $O(n_2 \log n_2)$ time. Steps 3 and 4 take $O(n_1)$ time. Steps 5 and 6 take $O(n_1 \log n_1)$ time. Step 7 takes at most $O(n_1)$ time and Step 8 takes constant time. The most expensive time of the algorithm is the loop consisting of Step 5 to Step 8. In the worst case, the algorithm will terminate when every job is “compressed”. In Step 7, Case 2 will compress a job, while Cases 1 and 3 will not. Thus, if we can

determine the number of steps Cases 1 and 3 take before Case 2 is executed, then we can determine the running time of the algorithm. Case 3 takes at most $r = O(n_1)$ steps. Case 1 takes at most $O(n_2(n_1 + n_2))$ steps. This is because there could be at most $n_1 + n_2$ pieces of jobs of agent A (The jobs of agent A are preempted at most n_2 times). In the worst case, these pieces all appear after the last block of agent B's jobs. For each execution of Case 1, one piece will be moved to the left. Therefore, after $O(n_2(n_1 + n_2))$ steps, every piece of agent A's jobs will be moved before the first block of agent B's jobs, and then Case 1 cannot occur again. This means that we have to execute Case 2 (which will compress the job). So the loop will be executed at most $O(n_2(n_1 + n_2))$ steps for each job. Thus, the overall running time of Algorithm 2 is $O(n_2(n_1 + n_2)n_1^2 \log n_1)$. \square

Remark 5. For the problem $1 \mid ctrl^a : \circ^b \mid \sum(C_j^a + w_j x_j^a) : f_{max}^b$ with agreeable weights, we can still run the above algorithm to obtain a schedule with preemption first. Then we merge the preempted pieces of jobs of agent A by moving the jobs of agent B forward. Obviously this will neither violate the constraint for jobs of agent B nor increase the total cost of agent A.

5 Maximum tardiness plus compression cost

In this section, we consider two problems involving due dates, namely, the objective function of agent A is the maximum tardiness (or maximum lateness) plus total compression cost. We shall prove that if the jobs of agent A have different release dates both these problems are unary NP-hard. First, we consider the problem $1 \mid ctrl^a, r_j^a, pmtn^a : \circ^b \mid (T_{max} + \sum w_j^a x_j^a) : f_{max}^b$.

Theorem 4. The problem $1 \mid ctrl^a, r_j^a, pmtn^a : \circ^b \mid (T_{max} + \sum w_j^a x_j^a) : f_{max}^b$ is unary NP-hard.

Proof. The proof is done via a reduction from 3-PARTITION. \square

Remark 6. Using the same proof as above, we can show that the problem $1 \mid ctrl^a, r_j^a, pmtn^a : \circ^b \mid (L_{max} + \sum w_j^a x_j^a) : f_{max}^b$ is also unary NP-hard.

At present, the complexity of the problem is not known if the jobs of agent A are all released at time 0. However, if $r_j^a = 0$ and $d_i^a \leq d_j^a \Rightarrow w_i^a \leq w_j^a$ for all i and j , then the problem becomes polynomially solvable. The algorithm is based on the fact that the jobs of agent A will appear in the optimal schedule in EDD order for any set of processing times of these jobs.

Property 2. There exists an optimal sequence for the problem in which all the jobs of agent A are sequenced by the modified preemptive EDD rule.

Because of this property, we can always compress the processing times of these jobs without changing the optimal job sequence of agent A. Below we describe such a polynomial-time algorithm for this problem. In the algorithm, we schedule jobs of agent B first and against their "deadlines" (the "deadlines" are computed via $f_{max}^b \leq Q$ for all the jobs of agent B).

Algorithm 3. (A polynomial-time algorithm for a special case)

Step 0: Sort the jobs of agent A in increasing order of their due dates. Assume after sorting, we have $d_1^a \leq d_2^a \leq \dots \leq d_{n_1}^a$.

Step 1: For each job j of agent B, compute its "deadline" \bar{d}_j^b via $f_{max}^b \leq Q$. Starting from $\max_{1 \leq j \leq n_2} \{\bar{d}_j^b\}$, schedule the jobs of agent B backwards so that each job is scheduled as close to its deadline as possible.

Step 2: Define block i as the i^{th} set of contiguously processed jobs of agent B. Let there be $k_1 \leq n_2$ blocks, and let h_i be the length of block i , $1 \leq i \leq k_1$.

Step 3: For each job j of agent A, let $p_j^a = \bar{p}_j^a$. For each job j of agent A, mark job j as “uncompressed” if $\bar{p}_j^a > \underline{p}_j^a$; otherwise, mark job j as “compressed”.

Step 4: Let $P = \{p_j^a \mid 1 \leq j \leq n_1\}$ (P is the current set of processing times of agent A’s jobs). $Cost = 0$ and $Benefit = 0$.

Step 5: Schedule the jobs of agent A by the preemptive EDD rule, skipping the time slots occupied by jobs of agent B. Let the maximum tardiness be T_{max} and $j_{max} = \min\{j : T_j = T_{max}\}$. If $T_{max} = 0$, go to Step 10.

Step 6: Let block k be last block before starting of job j_{max} and let the length of the block be h_k . Let l be the distance from time $C_{j_{max}}$ to the end of block k . If there is no such a block, let $l = 0$ and $h = 0$. Furthermore, let $T_{next} = \max\{T_j : \text{job } j \text{ is before job } j_{max}\}$ and $j_{next} = \min\{j : T_j = T_{next}\}$. If there is no such job, then $T_{next} = 0$.

Step 7: Let z be the first “uncompressed” job; i.e., the first $z - 1$ jobs are fully compressed. If every job before job j_{max} and including job j_{max} is “compressed” then go to Step 10. Let $x_z = p_z^a - \underline{p}_z^a$ and $y = T_{max} - T_{next}$.

Step 8: Let $\delta = \min\{x_z, y, l\}$. Compress job z by δ amount; i.e., $p_z^a = p_z^a - \delta$. We consider the following three cases, in the order they are presented.

[Case 1.] ($\delta = l$) $Cost = Cost + w_z * \delta$, $Benefit = Benefit + T_{max} - \max\{T'_j\}$, where T'_j is the new tardiness of jobs between job z and block k after compressing job z by l amount.

[Case 2.] ($\delta = x_z$) $Cost = Cost + w_z * \delta$, $Benefit = Benefit + \delta$, mark job z as “compressed”.

[Case 3.] ($\delta = z$) $Cost = Cost + w_z * \delta$, $Benefit = Benefit + T_{max} - \max\{T'_j\}$, where T'_j is the new tardiness of jobs between job z and block k after compressing job z by y amount.

Step 9: If $Cost < Benefit$ then go to Step 4 else go to Step 5.

Step 10: Using P (which has the last recorded processing times of the jobs), we schedule the jobs of agent A according to the preemptive EDD rule, skipping the time slots occupied by jobs of agent B. This is the final schedule.

Theorem 5. Algorithm 3 solves the problem $1 \mid ctrl^a, r_j^a, pmtn^a : \circ^b \mid (T_{max} + \sum w_j^a x_j^a) : f_{max}^b$ with agreeable weights in $O(n_2(n_1 + n_2)n_1^2)$ time.

Proof. The correctness part of the proof is similar to that in the proof of Theorem 3.

The computational complexity is similar as well, except it does not need to sort the jobs of agent A in each iteration due to Property 2. \square

Remark 7. If $d_i^a \leq d_j^a \Rightarrow w_i^a \leq w_j^a$ for all i and j , then a polynomial-time algorithm for the problem $1 \mid ctrl^a, pmtn^a : \circ^b \mid (L_{max}^a + \sum w_j x_j^a) : f_{max}^b$ is similar to that for the problem $1 \mid ctrl^a, pmtn^a : \circ^b \mid (T_{max}^a + \sum w_j x_j^a) : f_{max}^b$ with agreeable weights, except that in Step 5, it is not necessary to test if $L_{max} = 0$. The time complexity is the same as well.

Remark 8. We can generalize Algorithm 3 to solve the problem $1 \mid ctrl^a : \circ^b \mid (T_{max}^a + \sum w_j x_j^a) : f_{max}^b$ and the problem $1 \mid ctrl^a : \circ^b \mid (L_{max}^a + \sum w_j x_j^a) : f_{max}^b$ as well. We first solve the preempted version of the problem and then merge the preempted pieces of the jobs of agent A to obtain a nonpreemptive schedule.

6 Concluding remarks

We have studied several competitive agent scheduling problems with controllable processing times, where processing times of jobs of agent A are controllable. We provided either NP-hardness proofs or polynomial-time algorithms for some special cases of the problems. In fact, these results can

be extended to the corresponding cases where the processing times of jobs of agent B are also controllable with the compression costs charged to Agent A.

All the results obtained in this paper for two agent models in which the jobs of Agent A are allowed to be preempted have corresponding results in preemptive single machine scheduling environments with a single agent and availability constraints. The time periods that the machine in this paper was reserved for Agent B were always “against the deadlines of the jobs of Agent B”. The periods set aside for the processing of the jobs of Agent B were basically periods that the machine was not available for the processing of jobs of Agent A. There is an extensive amount of research done on scheduling with availability constraints, see Lee (2004). However, to our knowledge, single machine scheduling with availability constraints and controllable processing times had not yet been considered in the literature.

For future research, we note that the computational complexity of the problems with total flow time and maximum tardiness/lateness of agent A remain open. For those problems that are unary NP-hard, it will be interesting to develop approximation algorithms. Furthermore, it will be of interest to consider problems in a parallel machine environment.

Acknowledgment: We thank two anonymous referees for their helpful comments. The work of the first author is supported in part by the NSF of China Grant (70372058) and Guangdong (China) NSF Grant (031808). The work of the second author is supported in part by the NSF under Grants DMI-0300156 and DMI-0556010. The work of the third author is supported in part by the NSF under Grants DMI-0555999.

References

- [1] A. Agnetis, P. B. Mirchandani, D. Pacciarelli and A. Pacifici (2004), Scheduling Problems with Two Competing Agents, *Operations Research* **52**, 229 – 242.
- [2] K. R. Baker and J. C. Smith (2003), A Multiple-Criterion Model for Machine Scheduling, *Journal of Scheduling* **6**, 7 – 16.
- [3] M. R. Garey and D. S. Johnson (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA.
- [4] H. Hoogeveen and G. J. Woeginger (2002), Some Comments on Sequencing with Controllable Processing Times, *Computing* **68**, 181 – 192.
- [5] A. Janiak and M. Y. Kovalyov (1996), Single Machine Scheduling Subject to Deadlines and Resource Dependent Processing Times, *Eur. J. Oper. Res.* **94**, 284 – 291.
- [6] C-Y. Lee (2004), Machine Scheduling with Availability Constraints. In J. Y-T. Leung (ed.) *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, CRC Press, Boca Raton, FL.
- [7] J. Y-T. Leung (2004), Minimizing Total Weighted Error for Imprecise Computation Tasks and Related Problems, In J. Y-T. Leung (ed.) *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, CRC Press, Boca Raton, FL.
- [8] J. Y-T. Leung, M. L. Pinedo and G. Wan (2006), Competitive Agent Scheduling and Its Applications. Submitted.

- [9] J. Y-T. Leung, V. K. M. Yu and W.-D. Wei (1994), Minimizing the Weighted Number of Tardy Task Units, *Discrete Appl. Math.* **51**, 307 – 316.
- [10] E. Nowicki and S. Zdrzalka (1990), A Survey of Results for Sequencing Problems with Controllable Processing Times, *Discrete Appl. Math.* **26**, 271 – 287.
- [11] E. Nowicki and S. Zdrzalka (1995), A Bicriterion Approach to Preemptive Scheduling of Parallel Machines with Controllable Job Processing Times, *Discrete Appl. Math.* **63**, 271 – 287.
- [12] N. V. Shakhlevich and V. A. Strusevich (2005), Preemptive Scheduling Problems with Controllable Processing Times, *Journal of Scheduling* **8**, 233 – 253.
- [13] L. N. van Wassenhove and K. R. Baker (1982), A Bicriterion Approach to Time/Cost Tradeoffs in Sequencing, *Eur. J. Oper. Res.* **11**, 48 – 54.
- [14] R. G. Vickson (1980a), Choosing the Job Sequence and Processing Times to Minimize Total Processing Plus Flow Cost on a Single Machine, *Oper. Res.* **28**, 1155 – 1167.
- [15] R. G. Vickson (1980b), Two Single Machine Sequencing Problems Involving Controllable Job Processing Times, *AIIE Trans.* **12**, 258 – 262.
- [16] G. Wan, B. P. C. Yen and C. L. Li (2001), Single Machine Scheduling to Minimize Total Compression Plus Weighted Flow Cost is NP-hard, *Inform. Process. Lett.* **79**, 273 – 280.