Research Article

Two-Agent Single-Machine Scheduling with Resource-Dependent Starting Times

Peng Liu and Xiaoyu Tian

School of Management, Shenyang University of Technology, Shenyang 110870, China

Correspondence should be addressed to Peng Liu; liup7802@163.com

Received 22 May 2013; Revised 15 July 2013; Accepted 15 July 2013

Academic Editor: Yunqiang Yin

Copyright © 2013 P. Liu and X. Tian. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We consider several two-agent scheduling problems with resource consumption on a single machine, where each of the agents wants to minimize a measure dependent on its own jobs. The starting time of each job of the first agent is related to the amount of resource consumed. The objective is to minimize the total amount of resource consumption of the first agent with the restriction that the makespan or the total completion time of the second agent cannot exceed a given bound $U$. The optimal properties and the optimal polynomial time algorithms are proposed to solve the scheduling problems.

1. Introduction

Recently, there has been a growing interest in the literature to study multiagent scheduling problems, in which different agents share a common processing machine, and each agent wants to minimize a cost function dependent on its own jobs. Scheduling with multiple agents is firstly introduced by Baker and Smith [1] and Agnetis et al. [2]. Baker and Smith [1] manage to find a schedule that minimizes a combination of multiple agents’ objective functions. Agnetis et al. [2] study several scheduling problems with two agents that have requirements of their own objective functions. Some researchers pay their attention to the different machine environments and job characteristics of the scheduling problems with multiagent. Yuan et al. [3] show that two dynamic programming recursions in Baker and Smith [1] are incorrect and give a polynomial time algorithm for the same problems. Cheng et al. [4] extend one of the problems of Agnetis et al. [2] to the multiagent scheduling problem with minimizing the total weighted number of tardy jobs on a single machine. They show that the problem is strongly NP-complete in general, and they study a special case in which the number of agents is fixed. Cheng et al. [5] consider the multiagent scheduling problem on a single machine, where the agent’s objective functions are of the max form. They study the feasibility model and the minimality model. Leung et al. [6] study a two-agent scheduling environment with $m$ identical parallel machines and generalize the results of Baker and Smith [1] and Agnetis et al. [2] by including the total tardiness objective, allowing for preemptions, and considering jobs with different release dates. Gawiejnowicz et al. [7] consider a single-machine two-agent scheduling problem with proportionally deteriorating job processing times. Yin et al. [8] consider several two-agent scheduling problems with assignable due dates on a single machine. Gerstl and Mosheiov [9] consider the scheduling problems with two competing agents to minimize the weighted earliness-tardiness. Cheng et al. [10] study a two-agent single-machine scheduling problem with release times to minimize the total weighted completion time. Zhao and Lu [11] consider two models of two-agent scheduling problems on identical machines. Yin et al. [12] provide a branch-and-bound procedure for a single-machine earliness scheduling problem with two agents. Yu et al. [13] investigate several single-machine two-synergetic-agent scheduling problems. For other recent scheduling problems with two agents, the reader is referred to Lee et al. [14], Wu et al. [15], Feng et al. [16], and Wu et al. [17].

On the other hand, the scheduling problems with resource consumption have received a considerable attention in the last two decades. The classical scheduling problems have traditionally been considered under the assumptions that job processing times and release times are constant parameters and that the resource allocation issue is ignored. Cheng and Janiak [18] study the resource optimal control

However, to the best of our knowledge, the multiagent scheduling problem with the consideration of resource-dependent starting times has hardly been studied in the literature. These two categories of scheduling problems have been extensively and separately researched over the last two decades. In this paper, we study the two-agent scheduling problems on a single machine with resource-dependent starting times, where the goal is to find a schedule that minimizes the objective function of one agent with the restriction that the objective function of the other agent cannot exceed a given bound. The problems under consideration fall into the category of scheduling problems with resource consumption and multiple agents.

The remainder of this paper is organized as follows. In Section 2, we describe the proposed problems. In Sections 3 and 4, we develop the optimal polynomial time algorithms for the two-agent single-machine scheduling problems. Section 5 gives some concluding remarks.

2. Problem Description

We now describe our problems formally. There are two families of independent and nonpreemptive jobs $J^A = \{j^A_1, j^A_2, \ldots, j^A_n\}$ and $J^B = \{j^B_1, j^B_2, \ldots, j^B_n\}$ to be processed on a common single machine. All jobs are available for processing at time zero. The jobs in $J^A$ and $J^B$ are called $A$-agent's jobs and $B$-agent's jobs, respectively. Associated with each job $j^A_h$, let $P^A_h$ denote the processing time, $h = 1, 2, \ldots, n_A$. The starting time $s^A_h$ is related to the amount of the resource $f^A_h$ consumed on job $j^A_h$. We assume that $f^A_h = f(s^A_h)$, where $f : R^+ \to R^+$, called the resource consumption function, is a strictly increasing continuous function to $A$-agent's jobs. Associated with each job $j^B_h$, let $P^B_h$ denote the processing time, $k = 1, 2, \ldots, n_B$. Let $r$ indicate a feasible schedule of the $r = n_A + n_B$ jobs. Let $C^B_r(\pi)$ denote the completion time of $B$-agent's job $j^B_r$ under schedule $\pi$. The objective function of agent $A$ is to minimize the total amount of resource consumption $\sum_{h=1}^{n_A} f(s^A_h)$. The objective function of agent $B$ is to minimize the makespan $C^B_{\max} = \max_{k=1,2,\ldots,n_B} C^B_k(\pi)$ or the total completion time $\sum_{h=1}^{n_B} C^B_h(\pi)$.

The goal is to minimize the total amount of resource consumption $\sum_{h=1}^{n_A} f(s^A_h)$ of agent $A$ with the restriction that the makespan $C^B_{\max}$ or the total completion time $\sum_{h=1}^{n_B} C^B_h$ of agent $B$ cannot exceed a given bound $U$. If the value $U$ is too small, an instance of the scheduling problem may not have feasible solutions. If there is at least one feasible solution, we say that the instance is feasible. According to the three-field notation $\pi_1|\pi_2|\pi_3$ of Graham et al. [31], the two scheduling problems are denoted as $1\sum_{h=1}^{n_A} f(s^A_h) : C^B_{\max} \leq U$ and $1\sum_{h=1}^{n_B} f(s^A_h) : \sum_{h=1}^{n_B} C^B_h \leq U$, respectively.

3. Problem

In this section, we develop an optimal polynomial time algorithm to solve the problem $1\sum_{h=1}^{n_A} f(s^A_h) : C^B_{\max} \leq U$. The following Lemmas 1 and 2 are easily obtained, and we omit the details of the proof.

**Lemma 1.** Given a sequence $\pi = \{j^A_1, j^A_2, \ldots, j^A_n, j^B_1, j^B_2, \ldots, j^B_n\}$ and a constant $U$, define $C^B_\pi = \sum_{h=1}^{n_B} P^B_h$. Then, if $U < C^B_\pi$, the sequence $\pi$ corresponds to an infeasible schedule.

**Lemma 2.** Given a sequence $\pi = \{j^A_1, j^A_2, \ldots, j^A_n, j^B_1, j^B_2, \ldots, j^B_n\}$ and a constant $U$, define $C^B_\pi = \sum_{h=1}^{n_B} P^B_h + \sum_{h=1}^{n_B} P^B_h$. Then, if $U \geq C^B_\pi$, the schedule corresponding to the sequence $\pi$ is feasible.

Now we can define bounds for the constraint $U$. Define $U = \sum_{h=1}^{n_A} P^A_h$ and $U = \sum_{h=1}^{n_B} P^B_h + \sum_{h=1}^{n_B} P^B_h$.

**Lemma 3.** If $U < U = \sum_{h=1}^{n_A} P^A_h$, there are no feasible schedules.

**Lemma 4.** An optimal schedule exists in which the $A$-agent's jobs are processed in the nondecreasing order of processing times $P^A_h$.

**Proof.** The resource consumption function $f$ is a strictly increasing continuous function to $A$-agent's jobs. Since
starting $A$-agent’s jobs later consumes more resources, $A$-
agent’s jobs should be processed as early as possible. Hence, $A$-
agent’s jobs should be processed in a nondecreasing order of $p^A_h$.

**Lemma 5.** An optimal schedule exists in which the $B$-agent’s jobs are consecutively processed.

*Proof.* The makespan of agent $B$ is the maximum completion time of $B$-agent’s jobs on the single machine; that is, the makespan of agent $B$ is the completion time of $B$-agent’s last job. Using a pairwise job interchange argument, we can consecutively process $B$-agent’s jobs and consolidate all $B$-
agent’s jobs into a block.

Next, an algorithm to determine an optimal schedule of the problem $1 \parallel \sum_{h=1}^{n_A} f(s^B_h) : C^B_{\text{max}} \leq U$ is developed as follows.

**Algorithm 6.**

Step 1. Set $i \leftarrow 1$.

Step 2. Arrange the $A$-agent’s jobs as $\{J^A_1, J^A_2, \ldots, J^A_{n_A}\}$ according to the nondecreasing order of $p^A_h$ and denote all $B$-agent’s jobs as a dummy job $B1$.

Step 3. Define sequence $S = \{J^A_1, J^A_2, \ldots, J^A_{n_A-1}, J^A_{n_a}, B1\}$ and calculate the makespan $C^B_{\text{max}}$ for agent $B$. If $C^B_{\text{max}} \leq U$, then the sequence $S$ is an optimal schedule.

Step 4. Select the dummy job $B1$ to the $(n_A - i)$th position in the sequence $\{J^A_1, J^A_2, \ldots, J^A_{n_A-1}, J^A_{n_a}, J^A_{n_a+1}, \ldots, J^A_{n_a}, B1\}$ and calculate the makespan $C^B_{\text{max}}$ for agent $B$. If $C^B_{\text{max}} \leq U$, then the sequence $S_i = \{J^A_1, J^A_2, \ldots, J^A_{n_a-i}, B1, J^A_{n_a+i+1}, \ldots, J^A_{n_a}, J^A_{n_a}\}$ is an optimal schedule, and stop. Otherwise, go to Step 5.

Step 5. If $i < n_A$ and $C^B_{\text{max}} > U$, then $i \leftarrow i + 1$, and go to Step 4. Otherwise, stop.

**Theorem 7.** Algorithm 6 generates an optimal schedule for the problem $1 \parallel \sum_{h=1}^{n_A} f(s^B_h) : C^B_{\text{max}} \leq U$ in $O(n_A \log n_A + n_B)$ time.

*Proof.* The proof of optimality is straightforward from the results of Lemmas 1–5. We now turn to time complexity. The time to sequence the jobs of set $J^A$ according to the nondecreasing order of $p^A_h$ is $O(n_A \log n_A)$. Creating dummy job $B1$ incurs $O(n_B)$ operations. The solution requires finding locations for the dummy job $B1$ among the $A$-agent’s jobs sequenced by the nondecreasing order of $p^B_k$. So, the overall computational complexity of Algorithm 6 is bounded by $O(n_A \log n_A + n_B)$. This completes the proof.

### 4. Problem $1 \parallel \sum_{h=1}^{n_A} f(s^A_h) : \sum_{k=1}^{n_B} C^B_k \leq U$

For the problem $1 \parallel \sum_{k=1}^{n_B} f(s^B_k) : \sum_{k=1}^{n_B} C^B_k \leq U$, we have the following lemmas that are similar to Lemmas 1, 2, and 3.

**Lemma 8.** Given a sequence $\pi = \{J^A_1, J^A_2, \ldots, J^A_{n_A}, J^A_{n_a}, J^B_1, J^B_2, \ldots, J^B_{n_B}\}$ and a constant $U$, define $C^B_{\pi} = \sum_{k=1}^{n_B} (n_B - k + 1) p^B_k$. Then, if $U < C^B_{\pi}$, the sequence $\pi$ corresponds to an infeasible schedule.

*Proof.* If $U < C^B_{\pi}$, then

\[
\sum_{k=1}^{n_B} C^B_k = \sum_{k=1}^{n_B} \left( s^B_k + p^B_k \right) = \sum_{k=1}^{n_B} \left( s^B_k + \sum_{i=1}^{k-1} p^B_i + p^B_k \right) = \sum_{k=1}^{n_B} \left( s^B_k + \sum_{i=1}^{k} p^B_i \right) = \sum_{k=1}^{n_B} s^B_k + \sum_{k=1}^{n_B} \sum_{i=1}^{k} p^B_i = n_B s^B_{\max} + \sum_{k=1}^{n_B} \left( n_B - k + 1 \right) p^B_k = C^B_{\pi} > U,
\]

which implies that the sequence $\pi$ is infeasible.

**Lemma 9.** Given a sequence $\pi = \{J^A_1, J^A_2, \ldots, J^A_{n_A}, J^A_{n_a}, J^B_1, J^B_2, \ldots, J^B_{n_B}\}$ and a constant $U$, define $C^B_{\pi} = n_B \sum_{h=1}^{n_B} p^B_h + \sum_{k=1}^{n_B} (n_B - k + 1) p^B_k$. Then, if $U \geq C^B_{\pi}$, the schedule corresponding to the sequence $\pi$ is feasible.

*Proof.* If $U \geq C^B_{\pi}$, then

\[
\sum_{k=1}^{n_B} C^B_k = \sum_{k=1}^{n_B} \left( s^B_k + p^B_k \right) = \sum_{k=1}^{n_B} \left( s^B_k + \sum_{i=1}^{k-1} p^B_i + p^B_k \right) = \sum_{k=1}^{n_B} \left( s^B_k + \sum_{i=1}^{k} p^B_i \right) = \sum_{k=1}^{n_B} s^B_k + \sum_{k=1}^{n_B} \sum_{i=1}^{k} p^B_i = n_B s^B_{\max} + \sum_{k=1}^{n_B} \left( n_B - k + 1 \right) p^B_k = C^B_{\pi} \leq U,
\]

which implies that the sequence $\pi$ is feasible.

Now we can define bounds for the constraint $U$. Let the processing times of the $B$-agent’s jobs be ordered in the nondecreasing order of $p^B_1 \leq p^B_2 \leq \cdots \leq p^B_{n_B}$. Then, define $U = \sum_{h=1}^{n_B} (n_B - k + 1) p^B_k$ and $\bar{U} = n_B \sum_{h=1}^{n_B} p^B_h + \sum_{k=1}^{n_B} k p^B_k$. These constants play an important role in this problem.

**Lemma 10.** If $U < \bar{U} = \sum_{k=1}^{n_B} (n_B - k + 1) p^B_k$, there are no feasible schedules. If $U \geq \bar{U} = n_B \sum_{h=1}^{n_B} p^B_h + \sum_{k=1}^{n_B} k p^B_k$, there are feasible schedules for all possible sequences.

In view of these properties, the analysis in the following section will be confined to the case in which $\bar{U} \leq U \leq \bar{U}$.
Lemma 11. An optimal schedule exists in which the A-agent’s jobs are processed in the nondecreasing order of processing times \( p_A^k \).

Proof. The proof is same to Lemma 4, and we omit the details. \( \square \)

Lemma 12. An optimal schedule exists in which the B-agent’s jobs are consecutively processed in the nondecreasing order of processing times \( p_B^k \).

Proof. Using a pairwise job interchange argument, we can consecutively process B-agent’s jobs in the nondecreasing order of processing times \( p_B^k \).

Next, an algorithm to determine an optimal schedule of the problem \( \sum_{k=1}^{n_B} f(s_k^B) : \sum_{k=1}^{n_B} C_k^B \leq U \) is developed as follows.

Algorithm 13.

Step 1. Set \( i \leftarrow 1 \).

Step 2. Arrange the A-agent’s jobs as \( \{ j_1^A, j_2^A, \ldots, j_{n_A}^A \} \) according to the nondecreasing order of \( p_A^k \), and denote all B-agent’s jobs sequenced by the nondecreasing order of \( p_B^k \) as a dummy job \( B_1 \).

Step 3. Define sequence \( S = \{ j_1^A, j_2^A, \ldots, j_{n_A}^A, j_1^B, j_2^B, \ldots, j_{n_B}^B, B_1 \} \) and calculate the total completion time \( \sum_{k=1}^{n_A} C_k^B \) for agent B. If \( \sum_{k=1}^{n_A} C_k^B \leq U \), then the sequence \( S \) is an optimal schedule.

Step 4. Select the dummy job \( B_1 \) to the \( (n_A - i) \)th position in the sequence \( \{ j_1^A, j_2^A, \ldots, j_{n_A-i}^A, j_{n_A-i+1}^B, j_{n_A-i+2}^B, \ldots, j_{n_B}^B, B_1 \} \) and calculate \( \sum_{k=1}^{n_A} C_k^B \) for agent B. If \( \sum_{k=1}^{n_A} C_k^B \leq U \), then the sequence \( S_i = \{ j_1^A, j_2^A, \ldots, j_{n_A-i}^A, j_{n_A-i+1}^B, j_{n_A-i+2}^B, \ldots, j_{n_B}^B \} \) is an optimal schedule, and stop. Otherwise, go to Step 5.

Step 5. If \( i < n_A \) and \( \sum_{k=1}^{n_A} C_k^B > U \), then \( i \leftarrow i + 1 \), and go to Step 4. Otherwise, stop.

Theorem 14. Algorithm 13 generates an optimal schedule for the problem \( \sum_{k=1}^{n_B} f(s_k^B) : \sum_{k=1}^{n_B} C_k^B \leq U \) in \( O(n_A \log n_A + n_B \log n_B) \) time.

Proof. The proof of optimality is straightforward from the results of Lemmas 8–12. We now turn to time complexity. The times to sequence the jobs of sets \( j_A^k \) and \( j_B^k \) according to the nondecreasing orders of \( p_A^k \) and \( p_B^k \) are \( O(n_A \log n_A) \) and \( O(n_B \log n_B) \), respectively. Creating dummy job \( B_1 \) incurs \( O(n_B) \) operations. So, the overall computational complexity of Algorithm 13 is bounded by \( O(n_A \log n_A + n_B \log n_B) \). This completes the proof. \( \square \)

5. Conclusions

In this paper, we introduce a new scheduling model in which both resource-dependent starting times and two agents exist simultaneously. The objective is to minimize the total amount of resource consumption of the first agent with the restriction that the makespan or the total completion time of the second agent does not exceed a given bound. We propose the optimal properties and the optimal polynomial time algorithms for the considered scheduling problems.

Future research may be directed to analyze the problems with other objective functions such as minimizing the number of late jobs, the total weighted completion time, and tardiness. An interesting research topic is also to consider other resource consumption functions. Finally, the future research direction is to analyze the scheduling problem with more than two agents or in other machine environments.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant no. 71001074), the Program for Liaoning Excellent Talents in University (Grant no. WJQ2013003), and the Science Research Foundation, Department of Education, Liaoning Province of China (Grant no. W2010302).

References


