Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 197123, 7 pages
http://dx.doi.org/10.1155/2015/197123

Research Article

Multiple-Machine Scheduling with Learning Effects and Cooperative Games

Yiyuan Zhou[1,2] and Qiang Zhang[1]

[1] *School of Management and Economics, Beijing Institute of Technology, Beijing 10081, China*
[2] *College of Science, China Three Gorges University, Yichang, Hubei 443002, China*

Correspondence should be addressed to Yiyuan Zhou; zhou_yiyuan@126.com

Received 13 February 2015; Revised 28 April 2015; Accepted 28 April 2015

Academic Editor: Chin-Chia Wu

Multiple-machine scheduling problems with position-based learning effects are studied in this paper. There is an initial schedule in this scheduling problem. The optimal schedule minimizes the sum of the weighted completion times; the difference between the initial total weighted completion time and the minimal total weighted completion time is the cost savings. A multiple-machine sequencing game is introduced to allocate the cost savings. The game is balanced if the normal processing times of jobs that are on the same machine are equal and an equal number of jobs are scheduled on each machine initially.

1. Introduction

In a single-machine scheduling problem, a finite number of jobs are processed on a machine. Jobs are characterized by normal processing times, weights, due-dates, release times, and so on. The objective is to minimize a given function, such as the sum of the completion times, the sum of the weighted completion times, the maximal lateness, and the makespan.

Scheduling problems with learning effects were first introduced by Biskup [1], who described the actual processing time of a job as a decreasing power function of its position. Since then, several classes of learning effects in scheduling problems have been studied, namely, time-dependent learning effects [2–6], position-based learning effects [7–13], combined effects of position-based effects and time-dependent effects [14–17]. In a scheduling problem with time-dependent learning effects, the actual processing time of a job depends on the total processing times of jobs already processed, while, in a scheduling problem with position-based learning effects, the actual processing time of a job is a function of its position. In a scheduling problem with learning effects, the later a job starts, the shorter its actual processing time is. Learning effects have also been introduced to multiple-machine scheduling problems [8–10, 12, 15]. The main aims of the above works are optimal schedules, algorithms, and complexities of the algorithms. We refer to [18] for a review of scheduling with learning effects.

Cooperative games arising from scheduling or sequencing situations are called sequencing games. Curiel et al. [19] considered a single-machine sequencing situation, in which each job belongs to an agent (a player) and there is an initial order of jobs. The weight of a job is the cost per unit time of the agent. The optimal order minimizes the total cost which is the sum of the weighted completion times. The difference between the initial total cost and the minimal total cost can be seen as cost savings of all agents. This begs the question: How to allocate the cost savings among the agents fairly? Curiel et al. tackled this problem by introducing sequencing games. Sequencing games have been extended by many researchers. The extensions focus on the allocation rules [20, 21], the number of jobs [22, 23], the admissible rearrangements of jobs [24, 25], the initial order of jobs [26, 27], the family or batch sequencing situations [28, 29], the ready times [30], and the due-dates [31]. Convexity and balancedness of the corresponding sequencing games are the main goals of these studies as convex games and balanced games have nice properties.

Studies on the sequencing games with multiple machines are also found in the literatures. Hamers et al. [32] and Slikker [33] proposed a multiple-machine scheduling in which each player has one job and each job is processed on one machine only; the corresponding sequencing game is balanced if the normal processing times are equal or the weights are equal. Calleja et al. [34] considered a two-machine scheduling in which each player has two jobs and each job must be processed on each machine; it is shown that the corresponding sequencing game is balanced. Slikker [35] extended the model in [34] to sequencing games with m machines and n jobs.

In all the above mentioned models, the processing times are constant. van Velzen [36] studied a sequencing game with controllable processing times, where the processing times can be reduced to crashed processing times; the corresponding sequencing games are balanced but need not be convex.

In this paper, a multiple-machine scheduling problem with position-based learning effects is studied. In our model, each player has one job to be processed on one machine only, there is an initial schedule of the jobs, and the learning index depends on the machine. The actual processing time of a job is not a constant but a power function of its position. To achieve the optimal schedule, the jobs should be rearranged, and a rearrangement results in some changes of the processing times. The processing time of a job decreases if the job moves to the back of its queue; it increases if the job moves to the front of its queue. A multiple-machine sequencing game is introduced to allocate the cost savings. The game is balanced if and only if a related machine game is balanced. If in the initial schedule the normal processing times of jobs that are on the same machine are equal and each machine has an equal number of jobs, then the related machine game is balanced; furthermore, the multiple-machine sequencing game is balanced. To the best of our understanding, sequencing games have not been studied so far.

The rest of the paper is organized as follows. In Section 2 some preliminaries are recalled. In Section 3 a multiple-machine scheduling problem with learning effects is considered, and a cooperative game is defined on this scheduling problem. In Section 4 it is shown that a simple multiple-machine sequencing game with learning effects is balanced. Some concluding remarks are given in Section 5.

2. Preliminaries

A cooperative game is denoted by (N, v), where $N = \{1, 2, \ldots, n\}$ is the set of players and $v : 2^N \rightarrow \mathbb{R}$ is the characteristic function such that $v(\emptyset) = 0$.

Let (N, v) be a cooperative game and σ an order of the players. A coalition $S \subseteq N$ is connected if for all $i, j \in S$ and $l \in N$ with $\sigma(i) < \sigma(l) < \sigma(j)$ it holds that $l \in S$. The game (N, v) is a σ-component additive game if

(1) $v(\{i\}) = 0$ for all $i \in N$,

(2) $v(S) + v(T) \leq v(S \cup T)$ for all $S, T \subseteq N$ with $S \cap T = \emptyset$; that is, (N, v) is superadditive,

(3) $v(S) = \sum_{T \in S \setminus \sigma} v(T)$ for all $S \subseteq N$, where $S \setminus \sigma$ is the set of maximally connected components of S with respect to σ. A coalition $T \subseteq S$ is maximally connected if T is connected and $T \cup \{i\}$ is not connected for all $i \in S \setminus T$.

The core $C(v)$ of a cooperative game (N, v) is defined by

$$C(v) = \{x \in R^n \mid x(N) = v(N),\ x(S) \geq v(S)\ \forall S \subseteq N\}, \tag{1}$$

where $x(S) = \sum_{i \in S} x_i$.

For a cooperative game (N, v) and a coalition $S \subseteq N \setminus \{\emptyset\}$, the subgame $(S, v_{|S})$ is defined by $v_{|S}(T) = v(T)$ for all $T \subseteq S$.

Let $Z = (z_{ij})_{n \times n}$ be a matrix. A permutation game (N, u) is defined by

$$u(S) = \max_{\pi \in \Pi(S)} \sum_{i \in S} [z_{ii} - z_{i\pi(i)}] \tag{2}$$

for all $S \subseteq N$, where $z_{ij} \in Z$ and $\Pi(S)$ is the set of all permutations of S.

It is well known that the component additive games and the permutation games are balanced [37, 38]; thus their cores are not empty.

3. Multiple-Machine Scheduling with Learning Effects and Sequencing Games

In a multiple-machine scheduling problem, there are m machines and n players. Each player has one job to be processed on one of the machines, every job can be processed by any machine, and there is an initial schedule of jobs. Let $M = \{1, 2, \ldots, m\}$ denote the set of machines and let $N = \{1, 2, \ldots, n\}$ denote the set of players. With an abuse of notation we denote the job of player i by i itself. The normal processing time of job i is denoted by $p_i \geq 0$. The cost c_i of player i depends linearly on the completion time of his job; that is, $c_i(t) = w_i t$, where $w_i \geq 0$ is the weight of job i. The objective is to minimize the total cost which is the sum of the weighted completion times of the n jobs.

The initial schedule of jobs is denoted by $b^0 : N \rightarrow M \times N$, and $b^0(i) = (k, r)$ indicates that job i is processed in position r on machine k. The notation $b^{0^{-1}}$ is the inverse mapping of b^0.

We assume that all the machines have position-based learning effects. The learning index of machine k is denoted by a_k, where $a_k \leq 0$. Let p_{irk} be the actual processing time of job i if it is scheduled in position r on machine k; then

$$p_{irk} = p_i r^{a_k}. \tag{3}$$

A multiple-machine scheduling with learning effects as described above is denoted by (M, N, b^0, p, w, a), where vectors $p = (p_i)_{i \in N}$, $w = (w_i)_{i \in N}$, and $a = (a_k)_{k \in M}$.

The ordering $\prec$ on $M \times N$ is defined as follows. For $i, j \in N$, $b^0(j) \prec b^0(i)$ if and only if $b_1^0(j) = b_1^0(i)$ and $b_2^0(j) < b_2^0(i)$. It indicates that jobs i and j are on the same machine and i follows j. Note that $b_1^0(i)$ and $b_2^0(i)$ are the first component and the second component of $b^0(i)$, respectively. Let $P(b^0, i) = \{j \in N \mid b^0(j) \prec b^0(i)\}$ and $F(b^0, i) = \{j \in N \mid b^0(j) \succ b^0(i)\}$.

For each $k \in M$, the set $N_k(b^0) = \{j \in N \mid b_1^0(j) = k\}$ represents the jobs that are on machine k with respect to b^0, and $n_k(b^0)$ is the number of jobs on machine

k; that is, $n_k(b^0) = |N_k(b^0)|$. In addition, $\sigma_k^0 : N_k(b^0) \to \{1, 2, \ldots, n_k(b^0)\}$ denotes the initial processing order of the jobs that are on machine k; that is, $\sigma_k^0(i) = r$ implies that job i is in position r on machine k.

Suppose that job i is on machine k; then the starting time $t(b^0, i)$ of job i with respect to b^0 is given by

$$t\left(b^0, i\right) = \sum_{j \in P(b^0, i)} p_j \left[b_2^0(j)\right]^{a_k}, \tag{4}$$

and the completion time $T(b^0, i)$ of job i is the sum of its processing time and the waiting time; that is,

$$T\left(b^0, i\right) = p_i \left[b_2^0(i)\right]^{a_k} + \sum_{j \in P(b^0, i)} p_j \left[b_2^0(j)\right]^{a_k}. \tag{5}$$

We assume that, in the initial schedule, the starting time of the last job of each machine is less than or equal to the completion times of the last jobs of other machines. Formally, for each $k \in M$, the initial schedule b^0 satisfies

$$t\left(b^0, l_k\right) \le T\left(b^0, l_h\right) \tag{6}$$

for all $h \in M$, where l_k is the last job of machine k with respect to b^0. This assumption implies that in the initial schedule the last job of a machine cannot make any profit by joining the end of a queue of any other machine.

The cost of job i with respect to schedule b^0 is denoted by $C(b^0, i)$. For each coalition $S \subseteq N$, the total cost of S is the sum of the costs of jobs contained in S; that is,

$$C\left(b^0, S\right) = \sum_{i \in S} C\left(b^0, i\right) = \sum_{i \in S} w_i T\left(b^0, i\right). \tag{7}$$

Clearly, there is an optimal schedule b^* such that the total cost $C(b^*, N)$ is minimal. The maximal cost savings $C(b^0, N) - C(b^*, N)$ can be seen as a profit of all players.

The maximal cost savings of a coalition S depend on the set of admissible schedules of this coalition. A schedule $b : N \to M \times N$ is admissible for S with respect to b^0 if it satisfies the following conditions.

(1) For $i, j \in S$, jobs i and j which are on the same machine can be switched only if the jobs that are between i and j belong to S.

(2) For $i, j \in S$, jobs i and j that are on different machines can be switched only if both the jobs in $F(b^0, i)$ and the jobs in $F(b^0, j)$ are contained in S.

The set of all admissible schedules for S is denoted by $A(S)$.

For multiple-machine scheduling problems with learning effects (M, N, b^0, p, w, a), a corresponding m-sequencing game (N, v) is defined by

$$\begin{aligned} v(S) &= \max_{b \in A(S)} \left[C\left(b^0, S\right) - C(b, S)\right] \\ &= C\left(b^0, S\right) - C\left(b_S^*, S\right), \end{aligned} \tag{8}$$

where b_S^* is the optimal schedule for coalition S. Obviously, $v(\{i\}) = 0$ for all $i \in N$.

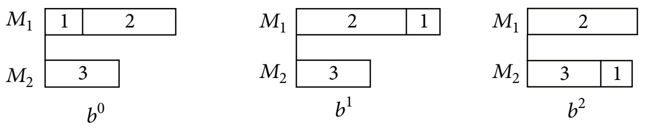

FIGURE 1: The schedules of b^0, b^1, and b^2.

Remark 1. Let $S \subseteq N_k(b^0)$ and the last job $l_k \in S$. To achieve the optimal schedule b_S^*, the queue of machine k should be rearranged. After the rearrangement, the last job $\bar{l}_k$ (it may be different from l_k) can make a profit by joining the end of a queue of another machine, although this will not occur in the initial schedule. The reason is that the starting time of the last job $\bar{l}_k$ of the new queue may increase. The following example illustrates this.

Example 2. Let (M, N, b^0, p, w, a) be a two-machine scheduling problem with learning effects, where $M = \{1, 2\}$, $N = \{1, 2, 3\}$, $p = (1, 3, 2)$, $w = (1, 3, 20)$, and $a = (-0.2, -0.3)$. The initial schedule b^0 is $b^0(1) = (1, 1)$, $b^0(2) = (1, 2)$, and $b^0(3) = (2, 1)$; it satisfies $t(b^0, 2) < T(b^0, 3)$ and $t(b^0, 3) < T(b^0, 2)$. Consider the coalition $S = \{1, 2\}$; we have $C(b^0, S) = 11.8350$.

Schedule b^1 is given by $b^1(1) = (1, 2)$, $b^1(2) = (1, 1)$, and $b^1(3) = (2, 1)$; thus $C(b^1, S) = 12.8706$.

Schedule b^2 is $b^2(1) = (2, 2)$, $b^2(2) = (1, 1)$, and $b^2(3) = (2, 1)$; then $C(b^2, S) = 11.8123$. The schedules b^0, b^1, and b^2 are depicted in Figure 1. Obviously, the optimal schedule for $\{1, 2\}$ is b^2. The cost $C(b^0, S)$ increases if the initial schedule is changed to b^1, but there is a profit if the last job of machine 1 in schedule b^1 joins the end of the queue of machine 2.

4. Balancedness of Simple m-Sequencing Games with Learning Effects

In this section, we restrict attention to simple multiple-machine scheduling problem with learning effects. A multiple-machine scheduling problem with learning effects is simple if the normal processing times of the jobs that are on the same machine are equal and an equal number of jobs are scheduled on each machine initially. Without loss of generality, we assume that, for all $k \in M$, $n_k(b^0) = q$ and $p_i = \lambda_k$ if $i \in N_k(b^0)$. The set of such simple multiple-machine scheduling problems is denoted by $SMS^{\lambda,q}$.

If job i is in position r on machine k, then the actual processing time of job i is $\lambda_k r^{a_k}$, the starting time is $\lambda_k \sum_{j=1}^{r-1} j^{a_k}$, and the completion time, denoted by $T(b, i)$, is given by

$$T(b, i) = \lambda_k \sum_{j=1}^{r} j^{a_k}. \tag{9}$$

For each $K \subseteq M$, let $S(K) = \cup_{k \in K} N_k(b^0)$ and let $b_{S(K)}^*$ denote the optimal schedule for coalition $S(K)$.

Lemma 3. *If $(M, N, b^0, p, w, a) \in SMS^{\lambda,q}$, then $n_k(b^*_{S(K)}) = q$ for all $k \in K$.*

Proof. In the initial schedule b^0, the starting time of the last job l_k of machine k is $\lambda_k \sum_{j=1}^{q-1} j^{a_k}$, and the completion time of the last job l_s of machine s is $\lambda_s \sum_{j=1}^{q} j^{a_s}$ since each machine has q jobs. According to assumption (6), we have $\lambda_k \sum_{r=1}^{q-1} r^{a_k} \le \lambda_s \sum_{r=1}^{q} r^{a_s}$ for all $s \in M$.

In the following, we will show that there is a contradiction if $n_k(b^*_{S(K)}) \ne q$.

Assume that $n_k(b^*_{S(K)}) < q$. Then there is a machine $h \in M$ such that $n_h(b^*_{S(K)}) \ge q + 1$. According to the admissible rearrangement, the last job l^*_h of machine h with respect to $b^*_{S(K)}$ is in coalition $S(K)$. In $b^*_{S(K)}$, the completion time $T(b^*_{S(K)}, l^*_k)$ of the last job l^*_k of machine k satisfies $T(b^*_{S(K)}, l^*_k) \le \lambda_k \sum_{r=1}^{q-1} r^{a_k}$ since machine k has at most $q - 1$ jobs, and the starting time $t(b^*_{S(K)}, l^*_h)$ of the last job l^*_h of machine h satisfies $t(b^*_{S(K)}, l^*_h) \ge \lambda_h \sum_{r=1}^{q} r^{a_h}$ as machine h has at least $q + 1$ jobs. If l^*_h joins the end of the queue of machine k, the cost of l^*_h will decrease since $t(b^*_{S(K)}, l^*_h) \ge \lambda_h \sum_{r=1}^{q} r^{a_h} \ge \lambda_k \sum_{r=1}^{q-1} r^{a_k} \ge T(b^*_{S(K)}, l^*_k)$; furthermore, the cost $C(b^*_{S(K)}, S(K))$ will decrease. It is contrary to the fact that $C(b^*_{S(K)}, S(K))$ is minimal.

Thus $n_k(b^*_{S(K)}) = q$ for all $k \in K$. □

Lemma 4. *Let $(M, N, b^0, p, w, a) \in SMS^{\lambda,q}$ and let (N, v) be the corresponding m-sequencing game. Then, for each $k \in M$, the subgame $(N_k(b^0), v_{|N_k(b^0)})$ is a σ^0_k-component additive game.*

Proof. Obviously, $(N_k(b^0), v_{|N_k(b^0)})$ is superadditive, and $v_{|N_k(b^0)}(\{i\}) = 0$ for all $i \in N_k(b^0)$. It is necessary to prove that $v_{|N_k(b^0)}(S) = \sum_{T \in S\setminus\sigma^0_k} v_{|N_k(b^0)}(T)$.

It follows from Lemma 3 that, in the optimal schedule $b^*_{N_k(b^0)}$, the jobs in $N_k(b^0)$ are not allowed to join the ends of other queues; thus $N_k(b^0) = N_k(b^*_{N_K(b^0)})$ and the subgame $(N_k(b^0), v_{|N_k(b^0)})$ is a single-machine sequencing game.

For each $i \in N_k(b^0)$, the cost of job i with respect to σ^0_k is denoted by $C(\sigma^0_k, i)$ and is given by $C(\sigma^0_k, i) = w_i T(\sigma^0_k, i)$. Following from (9), it holds that

$$C(\sigma^0_k, i) = w_i \lambda_k \sum_{r=1}^{\sigma^0_k(i)} r^{a_k}. \qquad (10)$$

For any coalition $S \subseteq N_k(b^0)$,

$$v_{|N_k(b^0)}(S) = \sum_{i \in S} \left[C(\sigma^0_k, i) - C(\sigma^*_{k,S}, i) \right], \qquad (11)$$

where $\sigma^*_{k,S}$ is the optimal order for S.

Suppose $S \setminus \sigma^0_k = \{T_1, T_2\}$; then T_1 and T_2 are connected, $T_1 \cap T_2 = \emptyset$, and $T_1 \cup T_2$ are not connected. The optimal orders of T_1 and T_2 are denoted by σ^*_{k,T_1} and σ^*_{k,T_2}, respectively. Thus

$$\begin{aligned} v_{|N_k(b^0)}(S) &= \sum_{i \in S} \left[C(\sigma^0_k, i) - C(\sigma^*_{k,S}, i) \right] \\ &= \sum_{i \in T_1} \left[C(\sigma^0_k, i) - C(\sigma^*_{k,S}, i) \right] \\ &\quad + \sum_{i \in T_2} \left[C(\sigma^0_k, i) - C(\sigma^*_{k,S}, i) \right] \\ &= \sum_{i \in T_1} \sum_{r=1}^{\sigma^0_k(i)} w_i \lambda_k r^{a_k} - \sum_{i \in T_1} \sum_{r=1}^{\sigma^*_{k,S}(i)} w_i \lambda_k r^{a_k} \\ &\quad + \sum_{i \in T_2} \sum_{r=1}^{\sigma^0_k(i)} w_i \lambda_k r^{a_k} - \sum_{i \in T_2} \sum_{r=1}^{\sigma^*_{k,S}(i)} w_i \lambda_k r^{a_k}, \end{aligned} \qquad (12)$$

where the third equality follows from (10). Since jobs in T_1 are not allowed to join the coalition T_2 and vice versa, we have $\sigma^*_{k,S}(i) = \sigma^*_{k,T_1}(i)$ if $i \in T_1$ and $\sigma^*_{k,S}(j) = \sigma^*_{k,T_2}(j)$ if $j \in T_2$. Hence

$$\begin{aligned} v_{|N_k(b^0)}(S) &= \sum_{i \in T_1} \sum_{r=1}^{\sigma^0_k(i)} w_i \lambda_k r^{a_k} - \sum_{i \in T_1} \sum_{r=1}^{\sigma^*_{k,T_1}(i)} w_i \lambda_k r^{a_k} \\ &\quad + \sum_{i \in T_2} \sum_{r=1}^{\sigma^0_k(i)} w_i \lambda_k r^{a_k} - \sum_{i \in T_2} \sum_{r=1}^{\sigma^*_{k,T_2}(i)} w_i \lambda_k r^{a_k} \\ &= C(\sigma_0, T_1) - C(\sigma^*_{k,T_1}, T_1) + C(\sigma_0, T_2) \\ &\quad - C(\sigma^*_{k,T_2}, T_2) \\ &= v_{|N_k(b^0)}(T_1) + v_{|N_k(b^0)}(T_2). \end{aligned} \qquad (13)$$

The proof follows exactly in the same way for $S \setminus \sigma^0_k = \{T_1, T_2, \ldots, T_l\}$. □

Remark 5. The subgame $(N_k(b^0), v_{|N_k(b^0)})$ need not be a σ^0_k-component additive game if there is an $i \in N_k(b^0)$ such that $p_i \ne \lambda_k$.

Example 6. Let $N_k(b^0) = \{1, 2, 3, 4\}$, $p = (3, 1, 1, 1)$, $w = (1, 1, 5, 3)$, $a_k = -0.2$, and $\lambda_k = 1$, and the initial order σ^0_k

is given by $\sigma_k^0(i) = i$ for all $i \in N_k(b^0)$. Take the coalition $S = \{1,2,4\}$ such that $S \setminus \sigma_k^0 = \{\{1,2\},\{4\}\}$. It follows that

$$\begin{aligned} v_{|N_k(b^0)}(\{1,2\}) &= w_1 p_1 + w_2 (p_1 + p_2 2^{a_k}) - w_2 p_2 \\ &\quad - w_1 (p_2 + p_1 2^{a_k}) = 2.2589, \\ v_{|N_k(b^0)}(S) &= w_1 p_1 + w_2 (p_1 + p_2 2^{a_k}) \\ &\quad + w_4 (p_1 + p_2 2^{a_k} + p_3 3^{a_k} + p_4 4^{a_k}) \\ &\quad - w_2 p_2 - w_1 (p_2 + p_1 2^{a_k}) \\ &\quad - w_4 (p_2 + p_1 2^{a_k} + p_3 3^{a_k} + p_4 4^{a_k}) \\ &= 3.0356, \end{aligned} \tag{14}$$

and $v_{|N_k(b^0)}(\{4\}) = 0$. Obviously, $(N_k(b^0), v_{|N_k(b^0)})$ is not a σ_k^0-component additive game since $v_{N_k(b^0)}(\{1,2,4\}) \neq v_{N_k(b^0)}(\{1,2\}) + v_{N_k(b^0)}(\{4\})$.

Although the position of job 4 is unchanged, its completion time decreases if jobs 2 and 3 are switched. In other words, switching of jobs 2 and 3 results in an "extra" profit for job 4.

Before we prove the balancedness of the corresponding m-sequencing game, a machine game is defined.

Let $(M, N, b^0, p, w, a) \in SMS^{\lambda,q}$ and let (N, v) be the corresponding m-sequencing game. For each $K \subseteq M$, a machine game (M, w) is defined by

$$w(K) = v\left(\bigcup_{k\in K} N_k(b^0)\right) - \sum_{k\in K} v(N_k(b^0)). \tag{15}$$

The machine game (M, w) is defined on the set of machines. The value $w(K)$ is the cost savings that the machines in K can attain if they cooperate.

Since subgames $(N_k(b^0), v_{|N_k(b^0)})$ are σ_k^0-component additive games for all $k \in M$, following Theorem 3.1 in [32] and Theorem 4.1 in [33], we have the following.

Theorem 7. *Let $(M, N, b^0, p, w, a) \in SMS^{\lambda,q}$, and let (N, v) and (M, w) be the corresponding m-sequencing game and machine game, respectively. Then (N, v) is balanced if and only if (M, w) is balanced.*

The following lemma shows that the coalition value $v(S(K))$ can be rewritten as a value of a permutation game.

Lemma 8. *If $(M, N, b^0, p, w, a) \in SMS^{\lambda,q}$ and (N, v) is the corresponding m-sequencing game, then there is a permutation game (N, u) such that*

$$v(S(K)) = u(S(K)) \tag{16}$$

for each $K \subseteq M$.

Proof. For convenience, renumber the machines in K such that $K = \{1, 2, \ldots, |K|\}$. Note that

$$\begin{aligned} C(b, S(K)) &= \sum_{k\in K}\sum_{r=1}^{q} w_{b^{-1}(k,r)} \lambda_k \sum_{\beta=1}^{r} \beta^{a_k}, \\ v(S(K)) &= \max_{b\in A(S(K))} \left[C(b^0, S(K)) - C(b, S(K))\right] \\ &= \max_{b\in A(S(K))} \left[\sum_{k\in K}\sum_{r=1}^{q} w_{b^{0^{-1}}(k,r)} \lambda_k \sum_{\beta=1}^{r} \beta^{a_k} \right. \\ &\quad \left. - \sum_{k\in K}\sum_{r=1}^{q} w_{b^{-1}(k,r)} \lambda_k \sum_{\beta=1}^{r} \beta^{a_k}\right]. \end{aligned} \tag{17}$$

For $i, j \in \{1, 2, \ldots, |K|q\}$, let $z_{ij} = w_i \lambda_k \sum_{\mu=1}^{j-(k-1)q} \mu^{a_k}$ if $(k-1)q + 1 \le j \le kq$; then we have

$$\begin{aligned} w_{b^{-1}(k,r)} \lambda_k \sum_{\beta=1}^{r} \beta^{a_k} &= z_{b^{-1}(k,r),(k-1)q+r}, \\ v(S(K)) &= \max_{b\in A(S(K))} \left[\sum_{k\in K}\sum_{r=1}^{q} z_{b^{0^{-1}}(k,r),(k-1)q+r} \right. \\ &\quad \left. - \sum_{k\in K}\sum_{r=1}^{q} z_{b^{-1}(k,r),(k-1)q+r}\right]. \end{aligned} \tag{18}$$

For each schedule $b \in A(S(K))$, there is a permutation $\pi^b \in \Pi(S(K))$ such that

$$\pi^b(i) = [b_1(i) - 1]\, q + b_2(i) \tag{19}$$

for all $i \in N$. For each permutation $\pi \in \Pi(S(K))$, there is a schedule $b^\pi \in A(S(K))$ satisfying

$$b^\pi(i) = (k, \pi(i) - (k-1)q), \tag{20}$$

where k is such that $(k-1)q + 1 \le \pi(i) \le kq$. Thus

$$\begin{aligned} v(S(K)) &= \max_{b\in A(S(K))} \left[\sum_{i\in S(K)} z_{i,\pi^{b^0}(i)} - \sum_{i\in S(K)} z_{i,\pi^b(i)}\right] \\ &= \max_{\pi\in\Pi(S(K))} \left[\sum_{i\in S(K)} z_{i,i} - \sum_{i\in S(K)} z_{i,\pi(i)}\right] \\ &= u(S(K)), \end{aligned} \tag{21}$$

where the first equality follows from (19) and the second equality follows under the assumption $\pi^{b^0}(i) = i$. □

Arising from $S(K) = \bigcup_{k\in K} N_k(b^0)$, we have

$$v\left(\bigcup_{k\in K} N_k(b^0)\right) = u\left(\bigcup_{k\in K} N_k(b^0)\right). \tag{22}$$

Theorem 9. *If $(M, N, b^0, p, w, a) \in SMS^{\lambda,q}$, then the corresponding m-sequencing game (N, v) is balanced.*

Proof. Following from (22), we have

$$\begin{aligned} w(K) &= v\left(\bigcup_{k\in K} N_k\left(b^0\right)\right) - \sum_{k\in K} v\left(N_k\left(b^0\right)\right) \\ &= u\left(\bigcup_{k\in K} N_k\left(b^0\right)\right) - \sum_{k\in K} v\left(N_k\left(b^0\right)\right). \end{aligned} \quad (23)$$

Since the permutation game (N, u) is balanced, the core $C(u)$ is not empty, and there is a vector $x \in C(u)$. Let $y_k = x(N_k(b^0)) - v(N_k(b^0))$ for all $k \in M$; then

$$\begin{aligned} \sum_{k\in K} y_k &= \sum_{k\in K} x\left(N_k\left(b^0\right)\right) - \sum_{k\in K} v\left(N_k\left(b^0\right)\right) \\ &\geq u\left(\bigcup_{k\in K} N_k\left(b^0\right)\right) - \sum_{k\in K} v\left(N_k\left(b^0\right)\right) = w(K), \end{aligned} \quad (24)$$

where the inequality follows from the fact that x is a core element of the game (N, u). If $K = M$, then $\sum_{k\in M} y_k = w(M)$. Thus $y \in C(w)$, the core of the game (M, w) is not empty, and (M, w) is balanced.

The balancedness of the m-sequencing game (N, v) follows from Theorem 7 directly. □

5. Conclusions

Cooperative games based on multiple-machine scheduling problems with learning effects are investigated, where the processing times are not constants. A necessary and sufficient condition for the balancedness of the corresponding sequencing games is that the related machine games are balanced. Furthermore, the sequencing games are balanced if normal processing times of the jobs that are on the same machine are equal and each machine has an equal number of jobs to be processed.

If the normal processing times of jobs that are on the same machine are different, then the corresponding subgames which are defined on one machine are not component additive games, and Theorem 7 does not hold. The balancedness of general sequencing games with learning effects in which the normal processing times are different will be studied in the future.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Science Foundation of China (71201089, 71371030, and 71401003), Specialized Research Fund for the Doctoral Program of Higher Education (20111101110036), and Yichang Science and Technology Research and Development Project (A201230225).

References

[1] D. Biskup, "Single-machine scheduling with learning considerations," *European Journal of Operational Research*, vol. 115, no. 1, pp. 173–178, 1999.

[2] W.-H. Kuo and D.-L. Yang, "Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect," *European Journal of Operational Research*, vol. 174, no. 2, pp. 1184–1190, 2006.

[3] C. Koulamas and G. J. Kyparisis, "Single-machine and two-machine flowshop scheduling with general learning functions," *European Journal of Operational Research*, vol. 178, no. 2, pp. 402–407, 2007.

[4] J. B. Wang, C. T. Ng, T. C. E. Cheng, and L. L. Liu, "Single-machine scheduling with a time-dependent learning effect," *International Journal of Production Economics*, vol. 111, no. 2, pp. 802–811, 2008.

[5] J.-B. Wang, "A note on single-machine scheduling with decreasing time-dependent job processing times," *Applied Mathematical Modelling*, vol. 34, no. 2, pp. 294–300, 2010.

[6] C.-C. Wu, P.-H. Hsu, J.-C. Chen, and N.-S. Wang, "Genetic algorithm for minimizing the total weighted completion time scheduling problem with learning and release times," *Computers and Operations Research*, vol. 38, no. 7, pp. 1025–1034, 2011.

[7] A. Bachman and A. Janiak, "Scheduling jobs with position-dependentprocessing times," *Journal of the Operational Research Society*, vol. 55, no. 3, pp. 257–264, 2004.

[8] G. Mosheiov, "Parallel machine scheduling with a learning effect," *Journal of the Operational Research Society*, vol. 52, no. 10, pp. 1165–1169, 2001.

[9] G. Mosheiov and J. B. Sidney, "Scheduling with general job-dependent learning curves," *European Journal of Operational Research*, vol. 147, no. 3, pp. 665–670, 2003.

[10] C.-C. Wu, W.-C. Lee, and W.-C. Wang, "A two-machine flowshop maximum tardiness scheduling problem with a learning effect," *International Journal of Advanced Manufacturing Technology*, vol. 31, no. 7-8, pp. 743–750, 2007.

[11] Y. Q. Yin, W.-H. Wu, W.-H. Wu, and C.-C. Wu, "A branch-and-bound algorithm for a single machine sequencing to minimize the total tardiness with arbitrary release dates and position-dependent learning effects," *Information Sciences*, vol. 256, pp. 91–108, 2014.

[12] C. L. Zhao, Q. L. Zhang, and H. Y. Tang, "Machine scheduling problems with a learning effect," *Dynamics of Continuous, Discrete and Impulsive Systems, Series A: Mathematical Analysis*, vol. 11, no. 5-6, pp. 741–750, 2004.

[13] C. Zhao, Y. Yin, T. C. Cheng, and C.-C. Wu, "Single-machine scheduling and due date assignment with rejection and position-dependent processing times," *Journal of Industrial and Management Optimization*, vol. 10, no. 3, pp. 691–700, 2014.

[14] T. C. E. Cheng, C.-C. Wu, and W.-C. Lee, "Some scheduling problems with sum-of-processing-times-based and job-position-based learning effects," *Information Sciences*, vol. 178, no. 11, pp. 2476–2487, 2008.

[15] Y. Yin, D. Xu, K. Sun, and H. Li, "Some scheduling problems with general position-dependent and time-dependent learning effects," *Information Sciences*, vol. 179, no. 14, pp. 2416–2425, 2009.

[16] Y. Q. Yin, D. H. Xu, and J. Y. Wang, "Single-machine scheduling with a general sum-of-actual-processing-times-based and job-position-based learning effect," *Applied Mathematical Modelling*, vol. 34, no. 11, pp. 3623–3630, 2010.

[17] C.-C. Wu and W.-C. Lee, “Single-machine scheduling problems with a learning effect,” *Applied Mathematical Modelling*, vol. 32, no. 7, pp. 1191–1197, 2008.

[18] D. Biskup, “A state-of-the-art review on scheduling with learning effects,” *European Journal of Operational Research*, vol. 188, no. 2, pp. 315–329, 2008.

[19] I. Curiel, G. Pederzoli, and S. Tijs, “Sequencing games,” *European Journal of Operational Research*, vol. 40, no. 3, pp. 344–351, 1989.

[20] H. Hamers, J. Suijs, S. Tijs, and P. Borm, “The split core for sequencing games,” *Games and Economic Behavior*, vol. 15, no. 2, pp. 165–176, 1996.

[21] C. Fernández, P. Borm, R. Hendrickx, and S. Tijs, “Drop out monotonic rules for sequencing situations,” *Mathematical Methods of Operations Research*, vol. 61, no. 3, pp. 501–504, 2005.

[22] P. Calleja, A. Estévez-Fernández, P. Borm, and H. Hamers, “Job scheduling, cooperation, and control,” *Operations Research Letters*, vol. 34, no. 1, pp. 22–28, 2006.

[23] V. Fragnelli, N. Llorca, J. Sanchez-Soriano, S. Tijs, and R. Branzei, “Convex games with an infinite number of players and sequencing situations,” *Journal of Mathematical Analysis and Applications*, vol. 362, no. 1, pp. 200–209, 2010.

[24] B. van Velzen and H. Hamers, “On the balancedness of relaxed sequencing games,” *Mathematical Methods of Operations Research*, vol. 57, no. 2, pp. 287–297, 2003.

[25] M. Slikker, “Relaxed sequencing games have a nonempty core,” *Naval Research Logistics*, vol. 53, no. 4, pp. 235–242, 2006.

[26] F. Klijn and E. Sanchez, “Sequencing games without initial order,” *Mathematical Methods of Operations Research*, vol. 63, no. 1, pp. 53–62, 2006.

[27] H. Hamers, F. Klijn, and B. van Velzen, “On the convexity of precedence sequencing games,” *Annals of Operations Research*, vol. 137, no. 1, pp. 161–175, 2005.

[28] B. Çiftçi, P. Borm, H. Hamers, and M. Slikker, “Batch sequencing and cooperation,” *Journal of Scheduling*, vol. 16, no. 4, pp. 405–415, 2013.

[29] S. Grundel, B. Çiftçi, P. Borm, and H. Hamers, “Family sequencing and cooperation,” *European Journal of Operational Research*, vol. 226, no. 3, pp. 414–424, 2013.

[30] H. Hamers, P. Borm, and S. Tijs, “On games corresponding to sequencing situations with ready times,” *Mathematical Programming*, vol. 69, no. 3, pp. 471–483, 1995.

[31] P. Borm, G. Fiestras-Janeiro, H. Hamers, E. Sanchez, and M. Voorneveld, “On the convexity of games corresponding to sequencing situations with due dates,” *European Journal of Operational Research*, vol. 136, no. 3, pp. 616–634, 2002.

[32] H. Hamers, F. Klijn, and J. Suijs, “On the balancedness of multiple machine sequencing games,” *European Journal of Operational Research*, vol. 119, no. 3, pp. 678–691, 1999.

[33] M. Slikker, “Balancedness of multiple machine sequencing games revisited,” *European Journal of Operational Research*, vol. 174, no. 3, pp. 1944–1949, 2006.

[34] P. Calleja, P. Borm, H. Hamers, F. Klijn, and M. Slikker, “On a new class of parallel sequencing situations and related games,” *Annals of Operations Research*, vol. 109, no. 1–4, pp. 265–277, 2002.

[35] M. Slikker, “Balancedness of sequencing games with multiple parallel machines,” *Annals of Operations Research*, vol. 137, no. 1, pp. 177–189, 2005.

[36] B. van Velzen, “Sequencing games with controllable processing times,” *European Journal of Operational Research*, vol. 172, no. 1, pp. 64–85, 2006.

[37] I. Curiel, J. Potters, R. Prasad, S. Tijs, and B. Veltman, “Sequencing and cooperation,” *Operations Research*, vol. 42, no. 3, pp. 566–568, 1994.

[38] S. H. Tijs, T. Parthasarathy, J. A. M. Potters, and V. R. Prasad, “Permutation games: another class of totally balanced games,” *OR Spektrum*, vol. 6, no. 2, pp. 119–123, 1984.