Exploring the Cooperative Game Nature of a Two-agent Scheduling Problem

Bing Zhao[1], Yanhong Gu[1*], Yuan Ruan[2], Quanle Chen[3]

[1] Intelligent Computational Science Institute, College of Mathematics and Statistics, Shenzhen University, Shenzhen, Guangdong, 518060, China
[2] College of Computer Science & Software Engineering, Shenzhen University, Shenzhen, Guangdong, 518060, China
[3] The Logistics Institute-Asia Pacific, National University of Singapore, 117574, Singapore
*Corresponding author: Yanhong Gu (Associate Professor) Email: yhgu@szu.edu.cn

Abstract — **The multi-party (multi-agent) scheduling problems have received increasing attention for about ten years. Most of the research only focuses on the scheduling aspect while a distinct cooperative game model is originally constructed for a two-party setting in this literature. This game model is established to highlight the reasonable (cooperative) profit sharing, although the current multi-agent scheduling research interest lies in the balancing cost allocation. The two parties, each having his own set of jobs, cooperatively reserve a processing time slot offered from a single (third-party) processing resource and perform their own jobs together on that resource. All the jobs of these two parties have a common processing time. Each of those jobs is uninterruptable and available at the beginning of the time slot. One cooperation profit function is proposed for each party to evaluate various profit sharing schemes. The function consists of three terms and is defined as the first term minus the other two. Term one is the gross profit relying on the number of the party's jobs, term two is the total completion time of his jobs determined by the (processing) sequence of both parties' jobs, i.e., global sequence, and term three is his opportunity cost. It means the function still and only depends on the global sequence after each party's job number is given. Two approaches based on cooperative games are designed to yield the final profit sharing scheme (solution) accepted to each party, the related algorithms are developed, and the analysis of computational complexity is presented. Besides the abovementioned contributions, the literature also mentions a situation accruing non-cooperation, where two successive time slots are independently and serially reserved by two parties.**

Keywords — Scheduling; Multi-agent; Cooperative games; Cooperation profit function; Profit sharing

I. INTRODUCTION

Nowadays, in economic activities, if multiple companies share the same manufacturing/processing resources, it's better for them to be allied and negotiate a processing sequence [1] of all the jobs of those parties. In this paper, "resource" refers to the processing facility, which, in most cases, is called "machine". Such multi-agent scheduling (MAS) problems are proposed originally by Agnetis et al. [2]. Since then, most of the related researchers have mainly focused on the circumstance where two parties form a business coalition.

Baker et al. [3] starts the assumption that each party in a two-party coalition considers a certain regular cost (function) [1]. The common decision variable of such functions is the (processing) sequence of all the jobs of those two parties. Moreover, they first point out that the two studied (cost) functions can be different and propose an objective for two-agent scheduling (TAS), which is to minimize the weighted sum of the two functions. The three objective (cost) functions considered in their paper are $C_{\max}$, $L_{\max}$ and $\sum w_j C_j$ [3].

Since then, in terms of objective function, more typical functions used in the study of scheduling, such as $f_{\max}$, $\sum u_j$, $\sum T_j$, etc., are considered in the literature on TAS. From the perspective of problem parameter, various typical job-machine environments [1] are addressed for TAS. In terms of research purpose, the published articles on TAS are interested in the development of algorithms to reach the abovementioned objective and the analysis of computational complexity of those algorithms [4-11]. Such a research mode on objective function, problem parameter, and research purpose is likewise adopted for the following two other kinds of mathematical model on TAS.

The first kind of model is actually established to present another weighting pattern for the two objective functions of a coalition. Agnetis et al. [11] first define this pattern as minimizing one party's cost (function) with the restriction that the other party's cannot exceed a given upper bound. There are a lot of publications [10, 12-33] on this kind of model. One reason is that such a constrained discrete optimization problem is better at quantifying the two parties' different degrees of bargaining power than the first weighting pattern presented in [3]. Another reason is that it evidently offers more excellent potential for obtaining the Pareto optimal (PO) solution set (POSS) on the two objective functions [11] than Baker et al. [3] do.

The second kind of model is just developed to obtain the abovementioned POSS. The study on this model begins with a special case where each party has to process only one job [2]. Agnetis et al. [11] further investigate a general circumstance where each party undertakes the work including a set of jobs and he considers the objective function defined as the total completion time of his jobs. They show that the number of PO solutions can be exponential in regard to the instance size. Although there have been few published research results on this model [34] since then, a large number of new contributions are finally presented in [35].

In recent years, there have been two literature reviews of the three classes of mathematical models in terms of cooperative games [36, 37].

For the MAS circumstances where more than two parties form a coalition, there are only a small number of related contributions because such circumstances cause obviously serious research difficulties [38-42].

This paper considers a TAS circumstance. Differently, an algorithm is designed to obtain the POSS on two cooperation profit functions which are originally exploited to become their respective objective functions. The common cooperation profit function of the two parties is defined as the given gross profit, achieved by processing his own jobs, minus the related processing cost (function) on job-schedule and a given opportunity cost.

This paper assumes that there is a common processing time for all the jobs of those two parties, each of the jobs only needs one operation on a common single machine, and the common cost function of the two parties is defined as $\sum C_j$. The polynomial solvability for POSS is provided by the developed algorithm even in the worst case. It implies that the number of PO solutions for the studied TAS circumstance is at most polynomial with respect to the instance size. The two parties can negotiate a PO solution (i.e., cooperation profit distribution scheme), acceptable to each party, in the POSS. Finally another (game) solution concept, based on the theory of N-person cooperative games and called core, is proposed in order to show that the two solution concepts are both valuable for finding the final solution employed by the coalition.

II. PROBLEM DESCRIPTION

This paper investigates a TAS circumstance where the two parties (denoted by A and B) jointly process their respective jobs in a (machine) time slot of a third-party single machine. A has n_{A} jobs to be processed while B has n_{B} ones. The job set of A is denoted by $X_{\mathrm{A}}(=\{J_1^{\mathrm{A}}, J_2^{\mathrm{A}}, \cdots, J_{n_{\mathrm{A}}}^{\mathrm{A}}\})$, and the job set of B is denoted by $X_{\mathrm{B}}(=\{J_1^{\mathrm{B}}, J_2^{\mathrm{B}}, \cdots, J_{n_{\mathrm{B}}}^{\mathrm{B}}\})$, where J_i^{A} denotes the i^{th} job in X_{A}, and J_j^{B} denotes the j^{th} job in X_{B}. Then, the set of all the jobs of the two parties can be denoted by $X(=\{J_1^{\mathrm{A}}, J_2^{\mathrm{A}}, \cdots, J_{n_{\mathrm{A}}}^{\mathrm{A}}, J_1^{\mathrm{B}}, J_2^{\mathrm{B}}, \cdots, J_{n_{\mathrm{B}}}^{\mathrm{B}}\})$, and $X = X_{\mathrm{A}} \cup X_{\mathrm{B}}$. The common processing time of all the jobs in X is denoted by p. It is assumed that each job in X is non-preemptive and available at the beginning time point of the reserved time slot.

σ is used to denote a (processing) sequence of all the jobs in *X*. The completion time of J_i^{A} for a σ is denoted by $C_i^{\mathrm{A}}(\sigma)$ or concisely by C_i^{A}. Similarly, C_j^{B} (concisely) denotes the completion time of J_j^{B} for the σ. In this paper, A's cost function relying on σ is defined as $\sum_{i=1}^{n_{\mathrm{A}}} C_i^{\mathrm{A}}$, and B's is defined as $\sum_{j=1}^{n_{\mathrm{B}}} C_j^{\mathrm{B}}$. Because each job needs no ready time, it is a reasonable assumption that the time slot begins at time (point) zero. Thus, its end time (point) is $np = (n_{\mathrm{A}} + n_{\mathrm{B}})p$ (see the example in Fig. (1)) for any σ with no idle machine time.

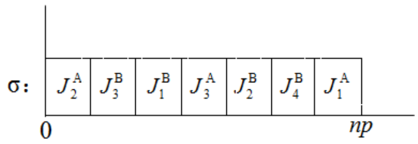

Fig.1 An Elementary Dominant Sequence

a_i denotes a given gross profit for J_i^{A} without any consideration for processing cost on job-sequence, $1 \le i \le n_{\mathrm{A}}$. It is assumed that $a_i = a$, $1 \le i \le n_{\mathrm{A}}$, in this paper. Additionally, b_j denotes the corresponding definition for J_j^{B}, and the corresponding assumption is that $b_j = b$, $1 \le j \le n_{\mathrm{B}}$. The given opportunity costs of the two parties are denoted by α_{A} and α_{B} respectively. Then, their two cooperation profit functions are respectively denoted by v_{A} and v_{B} and defined as

$$v_{\mathrm{A}} = v_{\mathrm{A}}\left(\sum_{i=1}^{n_{\mathrm{A}}} C_i^{\mathrm{A}}\right) = n_{\mathrm{A}} a - \sum_{i=1}^{n_{\mathrm{A}}} C_i^{\mathrm{A}} - \alpha_{\mathrm{A}}$$

and

$$v_{\mathrm{B}} = v_{\mathrm{B}}\left(\sum_{j=1}^{n_{\mathrm{B}}} C_j^{\mathrm{B}}\right) = n_{\mathrm{B}} b - \sum_{j=1}^{n_{\mathrm{B}}} C_j^{\mathrm{B}} - \alpha_{\mathrm{B}},$$

Where $n_{\mathrm{A}} a$ ($n_{\mathrm{B}} b$) is A's (B's) gross profit. Consequently, the coalition can only consider a set of elementary dominant (ED) sequences. Here, a sequence for X is called an ED one (also denoted by σ from now on) if and only if in the sequence there is no idle machine time from time zero (also see the example shown in Fig. (1)).

There are two scenarios, denoted by S1 and S2 respectively, addressed in this paper. In S1, the two parties have reserved their respective time slots independently. But they find that the two time slots are adjacent before they begin their respective work. This makes them assess the possibility of cooperation. In S2, the two parties want to reserve the time slot $[0,(n_A+n_B)p]$ together. If an ED sequence is acceptable to each party, they will jointly process their respective jobs in $[0,(n_A+n_B)p]$ by implementing that σ.

This paper analyzes the above two scenarios for the considered TAS circumstance in order to determine whether the two parties are willing to cooperate. If a scenario results in the cooperation between the two parties, an (ED sequence) σ^* and the related proper cooperation profit distribution scheme (v_A^*, v_B^*) will be derived by running the designed algorithm(s).

III. ANALYSIS AND SOLUTION

A. Scenario One (S1)

Consider σ_1 : $\{J_1^A \to J_2^A \to \cdots \to J_{n_A}^A \to J_1^B \to J_2^B \to \cdots \to J_{n_B}^B\}$. The corresponding (v_A, v_B) determined by the ED squence σ_1 can be computed as follows:

$$v_A = v_A\left(\sum_{i=1}^{n_A} C_i^A\right) = n_A a - (p + 2p + \cdots + n_A p) - \alpha_A = n_A a - \frac{n_A(n_A+1)}{2}p - \alpha_A,$$

$$v_B = v_B\left(\sum_{j=1}^{n_B} C_j^B\right) = n_B b - [(n_A p + p) + (n_A p + 2p) + \cdots + (n_A p + n_B p)] - \alpha_B = n_B b - (n_A + \frac{n_B+1}{2}) n_B p - \alpha_B.$$

Therefore,

$$\begin{aligned} v_A + v_B &= (n_A a + n_B b) - (\alpha_A + \alpha_B) - \left[\frac{n_A(n_A+1)}{2} + (n_A + \frac{n_B+1}{2}) n_B\right] p \\ &= (n_A a + n_B b) - (\alpha_A + \alpha_B) - \frac{n_A(n_A+1) + 2n_A n_B + n_B(n_B+1)}{2} p \\ &= (n_A a + n_B b) - (\alpha_A + \alpha_B) - \frac{(n_A + 1 + n_B)(n_A + n_B)}{2} p \\ &= (n_A a + n_B b) - (\alpha_A + \alpha_B) - \frac{(n+1)n}{2} p. \end{aligned}$$

Obviously, the same total cooperation profit $(v_A + v_B)$ can also be obtained if the two parties select σ_2 : $\{J_1^B \to J_2^B \to \cdots \to J_{n_B}^B \to J_1^A \to J_2^A \to \cdots \to J_{n_A}^A\}$. Furthermore, it is not difficult to know that any σ yields the above same $v_A + v_B$. In addition, if A (B) reserves (the time slot) $[0, n_A p]$ ($[0, n_B p]$) independently and then the two parties still cooperate in the time slot $[0,(n_A+n_B)p]$, any σ other than σ_1 (σ_2) not noly doesn't provide A (B) with more cooperation profit than

$$n_A a - \frac{n_A(n_A+1)}{2}p - \alpha_A$$

$$\left(n_B b - \frac{n_B(n_B+1)}{2}p - \alpha_B \right)$$

but also doesn't yield the total cooperation profit greater than

$$(n_A a + n_B b) - (\alpha_A + \alpha_B) - \frac{(n_A+1)n_A}{2}p - \frac{(n_B+1)n_B}{2}p$$

generated if they don't cooperate. Moreover,

$$(n_A a + n_B b) - (\alpha_A + \alpha_B) - \frac{(n_A+1)n_A}{2}p - \frac{(n_B+1)n_B}{2}p > (n_A a + n_B b) - (\alpha_A + \alpha_B) - \frac{(n+1)n}{2}p.$$

Hence, S1 does not result in the cooperation between A and B.

B. Scenario Two (S2)

Actually, Section 3.1 has shown that, if the two parties form a coalition to work together in $[0,(n_A+n_B)p]$ for their respective job-processing, any σ generates the same total completion time ($\frac{(n+1)n}{2}p$) and the same total cooperation profit ($(n_A a + n_B b) - (\alpha_A + \alpha_B) - \frac{(n+1)n}{2}p$).This implies that

Property 1 In S2, any (v_A, v_B), derived by a σ and satisfying v_A , $v_B > 0$, is a PO solution.

Clearly, it is still necessary to answer a key question: "Why are the two parties willing to cooperate?" The reason is that the cooperation may generate the possibility of reducing the price per machine time unit and therefore effectively making a and b greater than those two ones for the scenario where they independently reserve and use their respective time slots.

1) PO Solution Set (POSS)

This section develops an algorithm to derive the POSS for S2, analyzes the time complexity of the algorithm, and computes the tight upper bound of the number of PO solutions in a POSS.

If k_i^A (k_j^B) is used to denote the (processing) position number of J_i^A (J_j^B) in a σ (e.g., for J_3^A in Fig. (1), $k_3^A = 4$), then $C_i^A = k_i^A p$ ($C_j^B = k_j^B p$). Thus,

$$\sum_{i=1}^{n_A} C_i^A + \sum_{j=1}^{n_B} C_j^B$$
$$= p\left(\sum_{i=1}^{n_A} k_i^A + \sum_{j=1}^{n_B} k_j^B\right)$$
$$= (S_A + S_B)p = (1+2+\cdots+n-1+n)p$$
$$= \frac{n(n+1)}{2} p = Sp ,$$

where

$S_A = \sum_{i=1}^{n_A} k_i^A$, $S_B = \sum_{j=1}^{n_B} k_j^B$, $S = \frac{n(n+1)}{2}$, $S_A + S_B = S$.

Therefore,

$$S_A^{\min} = 1+2+\cdots+n_A = \frac{n_A(n_A+1)}{2} ,$$
$$S_A^{\max} = (n_B+1)+(n_B+2)+\cdots+(n_B+n_A-1)+(n_B+n_A)$$
$$= n_A \cdot n_B + \frac{n_A(n_A+1)}{2} ,$$

Because $p \ge 1$ and any integer from $S_A^{\min}$ to $S_A^{\max}$ can be assigned to S_A, the following property shows an essential characteristic of POSS.

Property 2 The tight upper bound of the number of PO solutions in a POSS is $n_A \cdot n_B + 1$.

Now, an algorithm to derive a POSS is developed as follows.

Algorithm 1

Step1. Set $m = n_A$, $\text{POSS} = \varnothing$.

Step2. Compute the (v_A, v_B) determined by the σ where the first n_A positions are arbitraily assigned to A's n_A jobs and the other n_B positions are arbitrarily assinged to B's n_B jobs. If v_A , $v_B > 0$, then set $\text{POSS} = \text{POSS} \bigcup \{(v_A, v_B)\}$.

Step3. Construct n_B ED sequence, $\sigma_1, \sigma_2, \cdots, \sigma_{n_B}$. In σ_k , $1 \le k \le n_B$, the first $m-1$ positions, the last $n_A - m$ positions, and the $(m+k)^{\text{th}}$ position are arbitraily assigned to A's n_A jobs, and the other n_B positions are arbitraily assinged to B's n_B jobs. Then, compute the n_B corresponding cooperation profit distribution schemes, $(v_A^{(1)}, v_B^{(1)})$, $(v_A^{(2)}, v_B^{(2)})$, $\cdots$, $(v_A^{(n_B)}, v_B^{(n_B)})$. $\forall k \in \{1,2,\cdots,n_B\}$, if $v_A^{(k)}$, $v_B^{(k)} > 0$, then set $\text{POSS} = \text{POSS} \bigcup \{(v_A^{(k)}, v_B^{(k)}\}$.

Step4. Set $m = m-1$.

Step5. If $m \ge 1$, go to Step3; otherwise, if $\text{POSS} = \varnothing$, then the cooperation is impossible; if $\text{POSS} \ne \varnothing$, then obtain the complete POSS.

It is clear that such an algorithm makes each PO solution obtained in $O(n)$ time. Hence,

Property 3 The time complexity of Algorithm 1 is $O(n_A n_B n)$.

The final solution (i.e., the final cooperation profit distribution scheme) (v_A^*, v_B^*) can be selected by A and B in the POSS, and the ED sequence determining the final solution is (denoted by) σ^* .

2) Core

Actually it is not difficult to know that the greater a and b (as analyzed in Section 3.2) can make the core of a two-party coalition non-empty. Thus, it is possible to design a core-based procedure to derive a final solution determined by a two-partition scheme of the total completion time. The following two instances are exploited to illustrate how to partition the total completion time and obtain the corresponding final solution.

Example 1 $p = 1$, $a = 20$, $b = 16$, $\alpha_A = \alpha_B = 0$, $n_A = n_B = 2$.

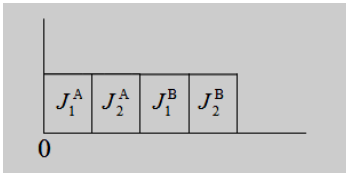

Fig.2 An elementary dominant sequence of Example 1

In this instance, the invariant total completion time is equal to 10, and the invariant total cooperation profit, obtained by the ED sequence σ^* (in Fig. (2)): $J_1^A \to J_2^A \to J_1^B \to J_2^B$, is equal to 62 $(= 20 \times 2 + 16 \times 2 - 0 - 0 - 10)$. The total completion cost is partitioned into two parts by the ratio of A's gross profit $(n_A \cdot a)$ to B's $(n_B \cdot b)$. Thus, the final solution

$$(v_A^*, v_B^*)$$
$$= (n_A \cdot a - \alpha_A - 10 \times \frac{5}{5+4}, n_B \cdot b - \alpha_B - 10 \times \frac{4}{5+4})$$
$$= (40 - 0 - 50/9, 32 - 0 - 40/9)$$
$$= (310/9, 248/9) .$$

Obviously, such a cost partition scheme is used to yield the final solution satisfying $v_A^* : v_B^* = (n_A \cdot a) : (n_B \cdot b)$.

Example 2 $p = 1$, $a = 20$, $b = 16$, $\alpha_A = 10$, $\alpha_B = 2$, $n_A = n_B = 2$.

Still use the σ^* in Fig. (2), and in this instance,

$$(n_A \cdot a - \alpha_A) : (n_B \cdot b - \alpha_B)$$
$$= (40 - 10) : (32 - 2)$$
$$= 1 : 1 .$$

Here, set

DOI 10.5013/IJSSST.a.17.27.18

ISSN: 1473-804x online, 1473-8031 print

$$(v_A^*, v_B^*)$$
$$= (40 - 10 \times \frac{1}{1+1} - 10, 32 - 10 \times \frac{1}{1+1} - 2)$$
$$= (25,25) .$$

Clearly, in this instance, there is the noticeable difference between the two opportunity costs. Hence, it is more reasonable that the scheme of cost partition should be in compliance with the ratio $(n_A \cdot a - \alpha_A):(n_B \cdot b - \alpha_B)$. By such a criterion, $(25,25)$ is yielded to become the final solution (v_A^*, v_B^*). It implies that the objective of solution selection is to make the ratio of v_A^* to v_B^* the same as $(n_A \cdot a - \alpha_A):(n_B \cdot b - \alpha_B)$.

Based on the above two instances, the following algorithm is developed to directly select a final solution from a non-empty core. Moreover, the objective of solution selection and the related criterion to partition the total completion cost, employed in Example 2, are necessarily modified for the more general application of this algorithm.

Algorithm 2

Step1. Consider $T_{B \to A}$ and $T_{A \to B}$, where

$$T_{B \to A} = n_A \cdot a - \alpha_A - C_{B \to A} ,$$
$$T_{A \to B} = n_B \cdot b - \alpha_B - C_{A \to B} ,$$
$$C_{B \to A} = n_A n_B p + \frac{n_A (n_A + 1)}{2} p ,$$
$$C_{A \to B} = n_B n_A p + \frac{n_B (n_B + 1)}{2} p .$$

If $T_{B \to A}$ or $T_{A \to B}$ is not positive, then the algorithm terminates and the cooperation doesn't seem to be the best. If both $T_{B \to A}$ and $T_{A \to B}$ are positive, then the core is not empty and the cooperation is valuable.

Step2. Construct σ^* : $J_1^A \to J_2^A \to \cdots \to J_{n_A - 1}^A \to J_{n_A}^A \to J_1^B \to J_2^B \to \cdots \to J_{n_B - 1}^B \to J_{n_B}^B$.

Step3. Compute the corresponding final solution (v_A^*, v_B^*).

If

$$\frac{n_A \cdot a - \alpha_A}{n_A \cdot a + n_B \cdot b - \alpha_A - \alpha_B} \cdot \frac{n(n+1)}{2} p \le C_{B \to A}$$

and

$$\frac{n_B \cdot b - \alpha_B}{n_A \cdot a + n_B \cdot b - \alpha_A - \alpha_B} \cdot \frac{n(n+1)}{2} p \le C_{A \to B} ,$$

then

$$(v_A^*, v_B^*)$$
$$= (n_A \cdot a - \alpha_A - \frac{n_A \cdot a - \alpha_A}{n_A \cdot a + n_B \cdot b - \alpha_A - \alpha_B} \cdot \frac{n(n+1)}{2} p,$$
$$n_B \cdot b - \alpha_B - \frac{n_B \cdot b - \alpha_B}{n_A \cdot a + n_B \cdot b - \alpha_A - \alpha_B} \cdot \frac{n(n+1)}{2} p) ;$$

If

$$\frac{n_A \cdot a - \alpha_A}{n_A \cdot a + n_B \cdot b - \alpha_A - \alpha_B} \cdot \frac{n(n+1)}{2} p > C_{B \to A} ,$$

then

$$(v_A^*, v_B^*)$$
$$= (n_A \cdot a - \alpha_A - C_{B \to A}, n_B \cdot b - \alpha_B - \frac{n_B (n_B + 1)}{2} p) ;$$

If

$$\frac{n_B \cdot b - \alpha_B}{n_A \cdot a + n_B \cdot b - \alpha_A - \alpha_B} \cdot \frac{n(n+1)}{n} p > C_{A \to B} ,$$

then

$$(v_A^*, v_B^*)$$
$$= (n_A \cdot a - \alpha_A - \frac{n_A (n_A + 1)}{2} p, n_B \cdot b - \alpha_B - C_{A \to B}) .$$

3) Some Criterions to Exploit POSS and Core

In terms of the definition mode of cooperation profit function, in order to generate the two-partition of the total cooperation profit which is satisfactory with each party and hence realize a potential two-party commercial coalition, it may be useful to jointly employ the two procedures, proposed respectively in Section 3.2.1 and Section 3.2.2, under the following two preconditions. One is that the time complexity of the algorithm to derive the POSS is low or moderate. The other is that the algorithm to implement the weighting pattern in [11] is substantially valuable for deriving the POSS [11]. On the other hand, the procedure, relying on the core (designed in Section 3.2.2) is obviously practical and effective if the POSS is empty or the algorithm to derive the POSS requires exponential time.

IV. CONCLUSION

This paper adopts a new research idea on multi-agent scheduling. From this new perspective, the distribution of processing cost has not been the final purpose, and furthermore, the distribution of processing profit is considered as the more global and reasonable intention in order to correctly judge whether the cooperation between two parties is more beneficial for each one. In addition, this paper proposes two approaches to the distribution task, and studies the relationship between the two ones.

In the future research, besides other job-machine environments, some more complicated definition frameworks on opportunity cost should be investigated preferentially.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflicts of interest.

ACKNOWLEDGMENT

The authors are grateful to the editor and the anonymous referees for their constructive comments and suggestions. This work is supported by the Natural Science Foundation of China (Grant No.61273220).

REFERENCES

[1] K. Baker, Introduction to sequencing and scheduling, New York (USA): John Wiley & Sons, 1974.

[2] A. Agnetis, P.B. Mirchandani, D. Pacciarelli, et al., “Nondominated schedules for a job-shop with two competing users,” Computational and Mathematical Organization Theory, vol. 6, pp. 191-217.2000

[3] K. Baker, J. Smith, “A multiple-criterion model for machine scheduling,” Journal of Scheduling, vol. 6, 2003, pp. 7-16.

[4] S. Gawiejnowicz, C. Suwalski, “Scheduling linearly deteriorating jobs by two agents to minimize the weighted sum of two criteria,” Computers & Operations Research, vol. 52, Part A, pp. 135-146.2014

[5] D.C. Li, P.H. Hsu, “Solving a two-agent single-machine scheduling problem considering learning effect,” Computers & Operations Research, vol. 39, pp. 1644-1651.2012

[6] Y. Yin, S.R. Cheng, T.C.E. Cheng, et al., “Two-agent single-machine scheduling with assignable due dates,” Applied Mathematics and Computation, vol. 219, pp. 1674-1685.2012

[7] Y. Yin, C.C. Wu, W.H. Wu, et al., “A branch-and-bound procedure for a single-machine earliness scheduling problem with two agents,” Applied Soft Computing, vol. 13, pp. 1042-1054.2013

[8] Q.Q. Nong, T.C.E. Cheng, C.T. Ng, “Two-agent scheduling to minimize the total cost,” European Journal of Operational Research, vol. 215,pp. 39-44.2011

[9] J.J. Yuan, W.P. Shang, Q. Feng, “A note on the scheduling with two families of jobs,” Journal of Scheduling, vol. 8,, pp. 537-542.2005

[10] N.H. Tuong, A. Soukhal, J.C. Billaut, “Single-machine multi-agent scheduling problems with a global objective function,” Journal of Scheduling, vol. 15,pp. 311-321.2012

[11] A. Agnetis, P.B. Mirchandani, D. Pacciarelli, et al., “Scheduling problems with two competing agents,” Operations Research, vol. 52, pp. 229-242.2004

[12] P. Liu, N. Yi, X. Zhou, et al., “Scheduling two agents with sum-of-processing-times-based deterioration on a single machine,” Applied Mathematics and Computation, vol. 219, pp. 8848-8855.2013

[13] W.C. Lee, S.K. Chen, C.C. Wu, “Branch-and-bound and simulated annealing algorithms for a two-agent scheduling problem,” Expert Systems with Applications, vol. 37, pp. 6594-6601.2010

[14] Y. Yin, W.H. Wu, S.R. Cheng, et al., “An investigation on a two-agent single-machine scheduling problem with unequal release dates,” Computers & Operations Research, vol. 39, pp. 3062-3073.2012

[15] T.C.E. Cheng, Y.H. Chung, S.C. Liao, et al., “Two-agent single-machine scheduling with release times to minimize the total weighted completion time,” Computers & Operations Research, vol. 40, pp. 353-361.2013

[16] W.C. Lee, Y.H. Chung, M.C. Hu, “Genetic algorithms for a two-agent single-machine problem with release time,” Applied Soft Computing, vol. 12, pp. 3580-3589.2012

[17] A. Agnetis, G. Pascale, D. Pacciarelli, “A Lagrangian approach to single-machine scheduling problems with two competing agents,” Journal of Scheduling, vol. 12, pp. 401-415.2009

[18] K. Zhao, X. Lu, “Approximation schemes for two-agent scheduling on parallel machines,” Theoretical Computer Science, vol. 468, pp. 114-121.2013

[19] C.C. Wu, W.H. Wu, J.C. Chen, et al., “A study of the single-machine two-agent scheduling problem with release times,” Applied Soft Computing, vol. 13, pp. 998-1006.2013

[20] E. Gerstl, G. Mosheiov, “Single machine just-in-time scheduling problems with two competing agents,” Naval Research Logistics, vol. 61, pp. 1-16.2014

[21] E. Gerstl, G. Mosheiov, “Scheduling problems with two competing agents to minimized weighted earliness-tardiness,” Computers & Operations Research, vol. 40, pp. 109-116.2013

[22] T.C.E. Cheng, S.R. Cheng, W.H. Wu, et al., “A two-agent single-machine scheduling problem with truncated sum-of-processing-times-based learning considerations,” Computers & Industrial Engineering, vol. 60, pp. 534-541.2011

[23] P. Liu, N. Yi, X. Zhou, “Two-agent single-machine scheduling problems under increasing linear deterioration,” Applied Mathematical Modelling, vol. 35, pp. 2290-2296.2011

[24] P. Liu, X. Zhou, L. Tang, “Two-agent single-machine scheduling with position-dependent processing times,” The International Journal of Advanced Manufacturing Technology, vol. 48, pp. 325-331.2010

[25] P. Liu, L. Tang, X. Zhou, “Two-agent group scheduling with deteriorating jobs on a single machine,” The International Journal of Advanced Manufacturing Technology, vol. 47, pp. 657-664.2010

[26] B.Q. Fan, T.C.E. Cheng, S.S. Li, et al., “Bounded parallel-batching scheduling with two competing agents,” Journal of Scheduling, vol. 16, 2013, pp. 261-271.

[27] S. Li, J. Yuan, “Unbounded parallel-batching scheduling with two competitive agents,” Journal of Scheduling, vol. 15, 2012, pp. 629-640.

[28] C.T. Ng, T.C.E. Cheng, J.J. Yuan, “A note on the complexity of the problem of two-agent scheduling on a single machine,” Journal of Combinatorial Optimization, vol. 12, pp. 387-394.2006

[29] J. Leung, M. Pinedo, G. Wan, “Competitive two-agent scheduling and its applications,” Operations Research, vol. 58, pp. 485-469.2010

[30] W.C. Lee, S.K. Chen, C.W. Chen, et al., “A two-machine flowshop problem with two agents,” Computers & Operations Research, vol. 38, pp. 98-104.2011

[31] B. Mor, G. Mosheiov, “Scheduling problems with two competing agents to minimize minmax and minsum earliness measures,” European Journal of Operational Research, vol. 206, pp. 540-546.2010

[32] B. Mor, G. Mosheiov, “Single machine batch scheduling with two competing agents to minimize total flowtime,” European Journal of Operational Research, vol. 215, pp. 524-531.2011

[33] G. Wan, S.R. Vakati, J. Leung, et al., “Scheduling two agents with controllable processing times,” European Journal of Operational Research, vol. 205, pp. 528-539.2010

[34] D. Elvikis, H.W. Hamacher, V. T’kindt, “Scheduling two agents on uniform parallel machines with makespan and cost functions,” Journal of Scheduling, vol. 14, pp. 471-481.2011

[35] A. Agnetis, J.C. Billaut, S. Gawiejnowicz, et al., Multiagent scheduling: models and algorithms, Berlin (German): Springer-Verlag Press, 2014.

[36] Y.H. Gu, G.C. Tang, “Scheduling game: new development of Nash bargaining solution,” Journal of Chongqing Normal University (Natural Science), vol. 29, pp. 1-6.2012

[37] G.C. Tang, B.Q. Fan, L.L. Liu, “Classifications, advances and prospects of scheduling games,” Journal of Chongqing Normal University(Natural Science), vol. 31, pp. 6-14.2014

[38] K. Lee, B.C. Choi, J. Leung, et al., “Approximation algorithms for multi-agent scheduling to minimize total weighted completion time,” Information Processing Letters, vol. 109, pp. 913-917.2009

[39] A. Agnetis, D. Pacciarelli, A. Pacifici, “Multi-agent single machine scheduling,” Annals Operations Research, vol. 150, pp. 3-15.2007

[40] T.C.E. Cheng, C.T. Ng, J.J. Yuan, "Multi-agent scheduling on a single machine with max-form criteria," European Journal of Operational Research, vol. 188, pp. 603-609.2008

[41] W.C. Lee, J.Y. Wang, "A scheduling problem with three competing agents," Computers & Operations Research, vol. 51, pp. 208-217.2014

[42] T.C.E. Cheng, C.T. Ng, J.J. Yuan, "Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs," Theoretical Computer Science, vol. 362, pp. 273-281.2006

[43] J.M. Bilbao, Cooperative games on combinatorial structures, Boston(USA): Kluwer Academic, 2000.