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

Bing Zhao¹, Yanhong Gu¹*, Yuan Ruan², Quanle Chen³

¹Intelligent Computational Science Institute, College of Mathematics and Statistics, Shenzhen University, Shenzhen, Guangdong, 518060, China
²College of Computer Science & Software Engineering, Shenzhen University, Shenzhen, Guangdong, 518060, China
³The 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
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 $X_A = \{J^A_1, J^A_2, \ldots, J^A_{n_A}\}$, and the job set of B is denoted by $X_B = \{J^B_1, J^B_2, \ldots, J^B_{n_B}\}$, where $J^A_i$ denotes the $i^{th}$ job in $X_A$, and $J^B_j$ denotes the $j^{th}$ job in $X_B$. Then, the set of all the jobs of the two parties can be denoted by $X = X_A \cup X_B$. 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.

$\sigma$ is used to denote a (processing) sequence of all the jobs in $X$. The completion time of $J^A_i$ for a $\sigma$ is denoted by $C^A_i(\sigma)$ or concisely by $C^A_i$. Similarly, $C^B_j$ (concisely) denotes the completion time of $J^B_j$ for the $\sigma$. In this paper, $A$’s cost function relying on $\sigma$ is defined as $\sum_{i=1}^{n_A} C^A_i$, and $B$’s is defined as $\sum_{j=1}^{n_B} C^B_j$. 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 $np = (n_A + n_B)p$ (see the example in Fig. (1)) for any $\sigma$ with no idle machine time.

$$\sigma: \begin{align*}
0 &< J^A_1 < J^A_2 < \ldots < J^A_{n_A} < J^B_1 < J^B_2 < \ldots < J^B_{n_B} \leq np
\end{align*}$$

Fig.1 An Elementary Dominant Sequence

$a_i$ denotes a given gross profit for $J^A_i$ without any consideration for processing cost on job-sequence, $1 \leq i \leq n_A$. It is assumed that $a_i = a$, $1 \leq i \leq n_A$, in this paper. Additionally, $b_j$ denotes the corresponding definition for $J^B_j$, and the corresponding assumption is that $b_j = b$, $1 \leq j \leq n_B$. The given opportunity costs of the two parties are denoted by $\alpha_A$ and $\alpha_B$ respectively. Then, their two cooperation profit functions are respectively by $v_A$ and $v_B$ and defined as

$$v_A = v_A\left(\sum_{i=1}^{n_A} C^A_i\right) = n_Aa - \sum_{i=1}^{n_A} C^A_i - \alpha_A$$
and

$$v_B = v_B\left(\sum_{j=1}^{n_B} C^B_j\right) = n_Bb - \sum_{j=1}^{n_B} C^B_j - \alpha_B,$$

Where $n_Aa$ ($n_Bb$) 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 $\sigma$ 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 : \( J_1^A \rightarrow J_2^A \rightarrow \cdots \rightarrow J_n^A \rightarrow J_1^B \rightarrow J_2^B \rightarrow \cdots \rightarrow J_n^B \). The corresponding (v_A, v_B) determined by the ED sequence σ_1 can be computed as follows:

\[
\begin{align*}
v_A &= v_A \left( \sum_{j=1}^{n_A} c_j^A \right) \\
&= n_A a - \left( p + 2p + \cdots + n_A p \right) - \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 - \left[ (n_A p + p) + (n_A p + 2p) + \cdots + (n_A p + n_B p) \right] - \alpha_B \\
&= n_B b - \left( n_A + \frac{n_B + 1}{2} \right) n_B p - \alpha_B.
\end{align*}
\]

Therefore,

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

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 \( k_1^A (k_1^B) \) is used to denote the (processing) position number of \( J_1^A (J_1^B) \) in a σ (e.g., for \( J_2^A \) in Fig. 1, \( k_2^A = 4 \)), then \( C_i^A = k_i^A p \, (C_i^B = k_i^B p) \). Thus,
The total completion cost is partitioned into two parts: 

\[ \sum_{i=1}^{n_A} a_i + \sum_{j=1}^{n_B} b_j \]

and 

\[ p \left( \sum_{i=1}^{n_A} k_i^A + \sum_{j=1}^{n_B} k_j^B \right) \]

\[ = (S_A + S_B)p = (1 + 2 + \cdots + n - 1 + n)p \]

\[ = n(n+1)/2 \]

Equation \( p = Sp \), where 

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

Therefore,

\[ S_A^{\min} = 1 + 2 + \cdots + n_A = n_A(n_A + 1)/2, \]

\[ S_A^{\max} = (n_B + 1) + (n_B + 2) + \cdots + (n_B + n_A - 1) + (n_B + n_A) \]

\[ = n_B \cdot n_B + n_A(n_A + 1)/2. \]

Because \( p \geq 1 \) 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

Step 1. Set \( m = n_A \), POSS = \( \emptyset \).

Step 2. Compute the \((v_A, v_B)\) determined by the \( \sigma \) where the first \( n_A \) positions are arbitrarily assigned to A’s \( n_A \) jobs and the other \( n_B \) positions are arbitrarily assigned to B’s \( n_B \) jobs. If \( v_A, v_B > 0 \), then set POSS = POSS \( \cup \{ (v_A, v_B) \} \).

Step 3. Construct \( n_B \) ED sequence, \( \sigma_1, \sigma_2, \cdots, \sigma_{n_B} \). In \( \sigma_1, 1 \leq k \leq n_B \), the first \( m - 1 \) positions, the last \( n_A - m \) positions, and the \( (m + k)\)th position are arbitrarily assigned to A’s \( n_A \) jobs, and the other \( n_B \) positions are arbitrarily assigned 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 POSS = POSS \( \cup \{ (v_A^{(k)}, v_B^{(k)}) \} \).

Step 4. Set \( m = m - 1 \).

Step 5. If \( m \geq 1 \), go to Step 3; otherwise, if \( \text{POSS} = \emptyset \), then the cooperation is impossible; if \( \text{POSS} \neq \emptyset \), 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 \( O(n \cdot n_B \cdot n) \).

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) \( \sigma^* \).

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](image)

In this instance, the invariant total completion time is equal to 10, and the invariant total cooperation profit, obtained by the ED sequence \( \sigma^\prime \) (in Fig. (2)): \( J^A_1 \rightarrow J^B_1 \rightarrow J^A_2 \rightarrow J^B_2 \), is equal to 62 \((= 20 \times 2 + 16 \times 2 - 0 - 10)\). 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

\[ (v_A^*, v_B^*) = \left(\frac{n_A \cdot a - \alpha_A}{5 + 4} \cdot n_B \cdot b - \alpha_B - 10 \times \frac{4}{5 + 4}\right) \]

\[ = \left(40 - 50/9, 32 - 0 - 40/9\right) \]

\[ = \left(310/9, 248/9\right). \]

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 \( \sigma^\prime \) in Fig. (2), and in this instance,

\[ (n_A \cdot a - \alpha_A) : (n_B \cdot b - \alpha_B) \]

\[ = \left(40 - 10 : 32 - 2\right) \]

\[ = 1:1. \]

Here, set
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 difference between the two opportunity costs. Hence, it is necessarily modified for the more general application of this algorithm.

**Algorithm 2**

Step1. Consider \( T_{B \rightarrow A} \) and \( T_{A \rightarrow B} \), where

\[
T_{B \rightarrow A} = n_A \cdot a - \alpha_A - C_{B \rightarrow A},
\]

\[
T_{A \rightarrow B} = n_B \cdot b - \alpha_B - C_{A \rightarrow B},
\]

\[
C_{B \rightarrow A} = n_A \cdot n_B \cdot p + \frac{n_A (n_A + 1)}{2} p,
\]

\[
C_{A \rightarrow B} = n_B \cdot n_A \cdot p + \frac{n_B (n_B + 1)}{2} p.
\]

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

Step2. Construct \( \sigma^* : J_1^A \rightarrow J_2^A \rightarrow \cdots \rightarrow J_{n_A}^A \rightarrow J_1^B \rightarrow J_2^B \rightarrow \cdots \rightarrow J_{n_B}^B \rightarrow J_{n_B}^B \).

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

If

\[
\frac{n_A \cdot a - \alpha_A}{n_A \cdot a + n_B \cdot b - \alpha_A - \alpha_B} \cdot \frac{n(n+1)}{2} p \leq C_{B \rightarrow A}
\]

and

\[
\frac{n_B \cdot b - \alpha_B}{n_A \cdot a + n_B \cdot b - \alpha_A - \alpha_B} \cdot \frac{n(n+1)}{2} p \leq C_{A \rightarrow B},
\]

then

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

If

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

then

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

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

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


DOI 10.5013/IJSSST.a.17.27.18

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

