Two-person cooperative games on minimizing the makespan

Lili Liu, Yanhong Gu, Han Ren, Guochun Tang

1Department of Applied Mathematics, Shanghai Second Polytechnic University, Shanghai 201209, China
2Department of Applied Mathematics, Shenzhen University, Shenzhen 518060, China
3Department of Mathematics, East China Normal University, Shanghai 200062, China
4Institute of Management Engineering, Shanghai Second Polytechnic University, Shanghai 201209, China
03902707@163.com

Abstract. This paper considers the problem of two-person cooperative games on minimizing the makespan. A special case of this problem is proved NP-hard, and a dynamic programming algorithm is presented for the general case, which runs in pseudo-polynomial time and indicates that this problem is binary NP-hard. This paper also adopts List Scheduling (LS) and Longest Processing Time (LPT) algorithms for classical parallel machines scheduling problem to cooperative game problem and develops the performance ratios of the algorithms.

1. Introduction
This paper studies the two-person cooperative games on scheduling problem. In reality, the machines may belong to different persons, and in order to maximize the overall profit, they need to cooperate and negotiate with each other to divide the jobs into two subsets.

Nash studied cooperative game problems in [1-2], and a lot of papers have been published since then, while most of them dealt with continuous utility set case. This paper considers the two-person cooperative games on scheduling problem. Each person offers one machine, so they need to cooperate with each other and divide the jobs into two disjoint subsets in order to obtain a maximum cooperation profit.

The problem considered in this paper can be described as follows. Given \( n \) jobs that will be processed by two persons, and person \( i \) has machine \( i \) (\( i = 1, 2 \)). The processing time of job \( J_j \) (\( j = 1, 2, \ldots, n \)) is denoted by \( p_j \). Given a solution of the two-person cooperative games on scheduling problem, let \( C_j \) denote the completion time of job \( J_j \). This paper considers the two-person cooperative games with makespan objective involved, which is denoted by \( C_{\text{max}} \). Given a solution, for \( i = 1, 2 \), person \( i \) can earn \( b_i \) by processing per unit processing time, the profit of person \( i \) can be formulated by \( v_i \), by processing per unit processing time, the profit of person \( i \) can be formulated by \( v_i = b_i \sum_{j \in X} p_j - f_{j \in X}(i) \), where \( X \) represents the set of job indices assigned to person \( i \), \( f_{j \in X}(i) \) represents the processing cost (scheduling objective function) contributed by the jobs with indices in set \( X \). The cooperative profit of the two persons is formulated by product of \( v_1v_2 \), and the objective considered in this paper is to maximize product \( v_1v_2 \).

Following the three-field notation [3], the problem under study is denoted as \( G2 \parallel v_1v_2 / C_{\text{max}} \) [5]. The first field \( G2 \) represents two-person cooperative games with scheduling objective problem and
there are two persons cooperating with each other to process the jobs. The second field represents jobs and game characteristics. The last field describes the optimized objective with the makespan as processing cost. Gu et al. [4] first study two-person cooperative games on scheduling problems. They prove that problem \( G^2 \| v_1 v_2 \mid L_{\text{max}} \) is binary NP-hard and develop a polynomial time algorithm for problem \( G^2 \| v_1 v_2 \mid L_{\text{max}} \) with identical job processing time.

For problem \( G^2 \| v_1 v_2 \mid C_{\text{max}} \), the objective value can be formulated by

\[
\left( b_1 \sum_{j \in X} p_j - \sum_{j \in Y} p_j \right) = \left( b_1 - 1 \right) \sum_{j \in X} p_j \sum_{j \in Y} p_j ,
\]

where \( X \) and \( Y \) are the sets of job indices assigned to person 1 and 2, respectively.

A polynomial time approximation algorithm for a maximization problem is called a \( \rho \) approximation algorithm if it can produce a feasible solution for every instance whose expected objective value is not smaller than a factor of \( \rho (0 < \rho < 1) \) of the optimal value, and \( \rho \) is also called the performance ratio (or performance guarantee) of this algorithm. If \( \rho \) is the largest possible performance ratio, it is also called the worst case performance ratio of this algorithm.

This paper first shows that problem \( G^2 \| v_1 v_2 \mid C_{\text{max}} \) is NP-hard even if \( b_1 = b_2 \), and develops a dynamic programming algorithm for the general case, which runs in pseudo-polynomial time and indicates that this problem is binary NP-hard. Then it explores the performance ratio of LS algorithm and illustrates that the worst case performance ratio of LS algorithm is \( \frac{1}{2} \), and the performance ratio of LPT algorithm is also developed.

2. Problem of Two-person Cooperative Games on Minimizing the Makespan

By reducing from the NP-hard Partition problem, problem \( G^2 \| v_1 v_2 \mid C_{\text{max}} \) will be proved NP-hard even if \( b_1 = b_2 \).

Partition: Given \( m \) positive integers \( a_1, \ldots, a_m \) satisfies \( \sum_{j=1}^{m} a_j = 2A \), is there a partition of the indices set \{1, \ldots, m\} into two disjoint subsets \( X \) and \( Y \) such that \( \sum_{j \in X} a_j = \sum_{j \in Y} a_j = A \).

**Theorem 1.** Problem \( G^2 \| v_1 v_2 \mid C_{\text{max}} \) is NP-hard even if \( b_1 = b_2 \).

**Proof.** Given an instance of Partition, the following instance of problem \( G^2 \| v_1 v_2 \mid C_{\text{max}} \) with \( n = m \) is constructed. Define a job \( J_j \) with processing time \( p_j = a_j \) for \( j = 1, \ldots, m \). Let \( b_1 = b_2 = 2 \).

Obviously, the construction of the instance takes polynomial time under binary encoding. It is easy to show that the constructed two-person cooperative games instance has a solution such that \( v_1 v_2 = A^2 \) if and only if Partition instance has a solution.

In a feasible solution of problem \( G^2 \| v_1 v_2 \mid C_{\text{max}} \), let \( X \) denote the set of indices such that the corresponding jobs are assigned to person 1 and \( Y \) denote the set of indices such that the corresponding jobs are assigned to person 2. It follows that

\[
v_1 = b_1 \sum_{j \in X} p_j - \sum_{j \in X} p_j = \sum_{j \in X} a_j ;
\]

\[
v_2 = b_2 \sum_{j \in Y} p_j - \sum_{j \in Y} p_j = \sum_{j \in Y} a_j .
\]
Denote $\sum_{j \in X} a_j = A + \beta, \sum_{j \in Y} a_j = A - \beta$, where $\beta$ is a positive integer satisfying $0 \leq \beta < A$. Hence $v_1 v_2 = (A + \beta)(A - \beta) = A^2 - \beta^2$. If and only if $\beta = 0$, Partition instance has a solution, thus $v_1 v_2 = A^2$.

For the general case of problem $G2 \parallel v_1 v_2 / C_{\text{max}}$, the following pseudo-polynomial time dynamic programming algorithm is proposed, and therefore show that this problem is binary NP-hard. First order the jobs in increasing order of their processing times. Denote $F_k(l_1, l_2)$ as the maximum profit of jobs $J_1, \ldots, J_k$ subject to the condition that the completion time of person $i$ is $l_i$. The initialization of the dynamic programming algorithm is

$$F_0(l_1, l_2) = \begin{cases} 0, & \text{if } l_1 = l_2 = 0, \\ -\infty, & \text{otherwise.} \end{cases}$$

For $k = 1, \ldots, n$ and $l_1, l_2 = 0, \ldots, \sum_{j=1}^n p_j$, the recursion equations are

$$F_k(l_1, l_2) = \max \left\{ F_{k-1}(l_1 - p_k, l_2) + (b_1 - 1) p_k (b_2 - 1) l_1, \\ F_{k-1}(l_1, l_2 - p_k) + (b_2 - 1) p_k (b_1 - 1) l_2 \right\}.$$

The optimal value of problem $G2 \parallel v_1 v_2 / C_{\text{max}}$ is equal to $\max_{0 \leq l_1, l_2 \leq \sum_{j=1}^n p_j} F_n(l_1, l_2)$ and the corresponding optimal solution can be obtained by backtracking. The dynamic programming algorithm runs in $O\left(n \left(\sum_{j=1}^n p_j\right)^2\right)$ time, which indicates that it is pseudo-polynomial time.

3. LS Algorithm

From the above discussion, it follows that problem $G2 \parallel v_1 v_2 / C_{\text{max}}$ is NP-hard, and the objective value is equal to $(b_1 - 1)(b_2 - 1) \sum_{j \in X} p_j \sum_{j \in Y} p_j$, where $b_1, b_2 > 1$. Adopt the LS (List Scheduling) algorithm for the classical parallel machines scheduling problem $P \parallel C_{\text{max}}$ to problem $G2 \parallel v_1 v_2 / C_{\text{max}}$: whenever a person becomes free, assign any unprocessed job to him.

**Theorem 2.** Let $V$ denote the profit obtained by applying LS algorithm to problem $G2 \parallel v_1 v_2 / C_{\text{max}}$, and $V'$ denote the maximum profit. Then $V \geq \frac{1}{2} V'$, and this bound is tight.

**Proof.** Let $P$ denote the sum of the processing times of the jobs, i.e., $P = \sum_{j=1}^n p_j$. Assign the jobs by LS to two persons. The processing time of the last job to finish is denoted as $p_l$. Consider the following two cases.

Suppose that $p_l \leq \frac{P}{2}$. Let $D_i$ be the completion time of the last job assigned to person $i$ in the solution constructed by LS, assume that $D_1 \geq D_2$. Then

$$D_1 + D_2 = P, \quad D_1 - D_2 \leq p_l, \quad (1)$$

From (1),

$$D_1^2 + D_2^2 + 2D_1D_2 = P^2, \quad (3)$$
From (2),
\[ D_1^2 + D_2^2 - 2D_1D_2 \leq p_i^2. \] \hspace{1cm} (4)

Combine (3) and (4), it follows that
\[ D_1D_2 \geq \frac{p^2 - p_i^2}{4}. \]

According to the NP-hardness proof of Theorem 1, it is easy to see that
\[ V^* \leq (b_1 - 1)(b_2 - 1) \frac{P^2}{4}. \] It follows that
\[
\frac{V}{V^*} \geq \frac{(b_1 - 1)(b_2 - 1)D_1(b_2 - 1)D_2}{(b_1 - 1)(b_2 - 1)p_i(P - p_i)} \geq \frac{P^2 - p_i^2}{4} \geq \frac{3}{4} \frac{P^2}{4} = \frac{3}{4} > \frac{1}{2}.
\]

Suppose that \( p_i > \frac{P}{2} \). Note that in the optimal solution the job with processing time \( p_i \) must be assigned to one person and the other jobs must be assigned to the other person, hence
\[ V^* = (b_1 - 1)(b_2 - 1)p_i(P - p_i). \] Suppose that the job with processing time \( p_i \) is assigned to person 1 in the solution constructed by LS. Denote the start time of the job with processing time \( p_i \) as \( t_1 \) and the completion time of the last job assigned to person 2 as \( t_2 \). According to the procedure of LS algorithm, \( t_1 \leq t_2 \). Hence
\[
\frac{V}{V^*} = \frac{(b_1 - 1)(t_1 + p_i)(b_2 - 1)t_2}{(b_1 - 1)(b_2 - 1)p_i(P - p_i)} = \frac{P^2 - p_i^2}{4} \geq \frac{3}{4} \frac{P^2}{4} = \frac{3}{4} > \frac{1}{2}.
\]

**Example 1.** Consider an instance of problem \( G2 | e_1 = e_2 = 0 | v_1v_2 / C_{\text{max}} \) with the following data:
\[ p_1 = p_2 = 1, p_3 = M (M \gg 2). \]

It is easy to see that the maximum profit \( V^* = (b_1 - 1)(b_2 - 1)2M \). LS will assign jobs with processing times 1 and \( M \) to one person and a job with processing time 1 to the other person, hence
\[ V = (b_1 - 1)(b_2 - 1)(1 + M). \] Thus
\[
\frac{V}{V^*} = \frac{(b_1 - 1)(b_2 - 1)(1 + M)}{(b_1 - 1)(b_2 - 1)2M} = \frac{1 + M}{2M},
\]
where \( \frac{1 + M}{2M} \rightarrow \frac{1}{2} \) when \( M \rightarrow +\infty \).

### 4. LPT Algorithm

This section extends the LPT (Longest Processing Time) algorithm for classical parallel machines scheduling problem \( P \| C_{\text{max}} \) to problem \( G2 \| v_1v_2 / C_{\text{max}} \).

**Algorithm LPT**

Sequence the jobs in decreasing order of their processing times, once a person is free, assign the job with the largest processing time to him among the unassigned jobs.

**Theorem 3.** Let \( V \) denote the profit obtained by applying LPT algorithm to problem \( G2 \| v_1v_2 / C_{\text{max}} \), and \( V^* \) denote the maximum profit. Then \( \frac{24}{25} V^* \geq V \).
**Proof.** Let $P = \sum_{j=1}^{n} p_j$. Assign the jobs by LPT to two persons. Denote the completion time of the last job assigned to person $i$ as $D_i$, suppose that $D_1 \geq D_2$, and the processing time of the last job to finish as $p_j$. From the discussion in Theorem 2, it follows that $D_1 D_2 \geq \frac{p^2 - p_j^2}{4}$. Consider the following two cases.

Suppose that $p_i \leq \frac{P}{5}$. Hence

$$\frac{V}{V'} \geq \frac{(b_1 - 1)D_1(b_2 - 1)D_2}{(b_1 - 1)(b_2 - 1)\frac{P^2}{4}} = \frac{D_1 D_2}{\frac{P^2}{4}} \geq \frac{\frac{p^2 - p_j^2}{4}}{\frac{p^2}{4}} \geq \frac{24}{100} = \frac{24}{25}.$$ 

Suppose that $p_i > \frac{P}{5}$. This would imply that there are at most four jobs with processing times no smaller than $p_j$ and the sum of processing times of the other jobs is less than $p_j$. Denote the number of the jobs with processing times no smaller than $p_j$ as $t$. If $t = 1$, then in the optimal solution the job with processing time $p_i$ is assigned to one person and the other jobs are assigned to the other person, this is exactly the solution that LPT would provide. If $t = 2$, note that the job with processing time $p_j$ is finished last in the solution constructed by LPT, hence there are exactly two jobs with processing times $p_j$. It is easy to see that LPT returns the optimal solution. If $t = 3$, sequence the jobs in decreasing order of their processing times such that $p_1 \geq p_2 \geq p_3 (p_i) \cdots$. In this case, the optimal solution assigns the jobs with processing times $p_2$ and $p_j$ to one person and the other jobs to the other person. This can be shown to be optimal by a simple interchange procedure. This solution is also exactly the solution that LPT would construct. If $t = 4$, sequence the jobs in decreasing order of their processing times such that $p_1 \geq p_2 \geq p_3 \geq p_4 (p_i) \cdots$. Thus $\frac{P}{5} < p_i < \frac{2P}{5}$, otherwise, $P = p_1 + p_2 + p_3 + p_4 + \cdots > \frac{2P}{5} + \frac{3P}{5} = P$, it's a contradiction. Hence, LPT constructs the following solution: jobs with processing times $p_1$ and $p_4 (p_i)$ are assigned to one person and the other jobs are assigned to the other person. This can also be shown to be optimal by a simple interchange procedure.

**5. Conclusion**

This paper explores two-person cooperative games with makespan as the scheduling objective. The problem is proved NP-hard even for a special case of this problem. A dynamic programming algorithm is proposed for the general case, which runs in pseudo-polynomial time and indicates that this problem is binary NP-hard. The performance ratios of two approximation algorithms are also developed. This paper cannot prove that the performance ratio of LPT shown in Theorem 3 is tight, to find an instance with performance ratio $\frac{24}{25}$ or to propose a larger bound is a very challenging open question for future research.
Acknowledgments
This research is supported by the National Natural Science Foundation of China (No. 11601316) and discipline “Applied Mathematics” of Shanghai Second Polytechnic University (No. XXKPY1604).

References
[1] Nash John. The bargaining problem [J], Econometrica, 1950, 18, 155-162.
[2] Nash John. Two person cooperative games [J], Econometrica, 1953, 21, 128-140.
[3] Graham Ronald, Lawler Eugene et al. Optimization and approximation in deterministic sequencing and scheduling: A survey [J], ADM. 1979, 5, 287-326.
[4] Gu Yanhong, Fan Jing et al. Maximum lateness scheduling problem on two-person cooperative games [J], Journal of Combinatorial Optimization, 2013, 26, 71-81.
[5] Jin Ji, Gu Yanhong et al. Two-person cooperative games on makespan scheduling [J], Journal of Shanghai Second Polytechnic University, 2011, 28, 14-17.