

A Generalized Multi-organization Scheduling on Unrelated Parallel Machines

February 13, 2009

Fukuhito Ooshita

Graduate School of Information Science
and Technology, Osaka University
Osaka 560-8531, Japan
f-oosita@ist.osaka-u.ac.jp

Tomoko Izumi

Center of Social Contribution and Collaboration,
Nagoya Institute of Technology
Aichi 466-8555, Japan
izumi.tomoko@nitech.ac.jp

Taisuke Izumi

Graduate School of Engineering,
Nagoya Institute of Technology
Aichi 466-8555, Japan
t-izumi@nitech.ac.jp

Abstract We consider the parallel computing environment where multiple organizations provide machines and several jobs to be executed. While cooperation of organizations is required to minimize the global makespan, each organization also expects the faster completion of its own jobs primarily and thus it is not necessarily cooperative.

To handle the above situations, we formulate the α -cooperative multi-organization scheduling problem (α -MOSP), where $\alpha \geq 1$ is a parameter representing the degree of cooperativeness. α -MOSP minimizes the makespan under the *cooperation constraint* that each organization does not allow the completion time of its own jobs to be delayed α times of that in the case where those jobs are executed by itself.

In this paper, we aim to reveal the relation between the makespan and the degree of cooperativeness. First, we investigate the relation between α and the quality of the global makespan. For $\alpha = 1$ (i.e., the completely uncooperative case), we show an instance where the cooperation constraint degrades the optimal makespan by m times. In contrast, for $\alpha > 1$, we can construct an algorithm transforming any unconstrained schedule to one satisfying the cooperation constraint. This algorithm bounds the degradation ratio by $\alpha/(\alpha - 1)$, which implies that weak cooperation improves the makespan dramatically. Second, we study the complexity of α -MOSP. We show its strongly NP-hardness and inapproximability for the approximation factor less than $\max\{(\alpha + 1)/\alpha, 3/2\}$. We also show the hardness of transformation: Even if an optimal schedule under no cooperation constraint is given, no polynomial-time algorithm finds an optimal schedule for α -MOSP. This result is a witness for inexistence of general polynomial-time transformation algorithms that preserve the approximation ratio.

1 Introduction

Background and Motivation. Cooperative computing has become important with an increase in the demand for considerable computational power. In typical cooperative computing, several independent organizations provide their computational resources and form a huge parallel computing environment by concentrating all the resources. Each organization can obtain the benefit by executing its jobs on the environment. However, we have some concern that the benefit of each organization would be decreased when multiple organizations utilize the environment at the same time. In fact, if the environment maximizes the global benefit (such as the global makespan), it may decrease the local benefit of each organization (such as the completion time of its jobs) [13]. If an organization

suffers a significant decrease in its local benefit, the organization has no reason to participate in the cooperative computing. Thus, to keep the organizations into the cooperative computing, it is necessary to maximize the global benefit without sacrificing the local benefit of each organization. However, in general, the environment must decrease the global benefit to increase the local benefit because it must maximize the global benefit in more constrained situations. In other words, there exists the tradeoff between the global and local benefit. Then, it is a challenging problem to clarify the inherent difficulty caused by such tradeoff.

To consider the above situation, the *Multi-Organization Scheduling Problem* (abbreviated by MOSP) is proposed by Pascual et al. in [13]. MOSP considers a parallel computing environment formed by several organizations, where each organization provides a multiprocessor machine and independent jobs to be executed. While the goal of MOSP is to minimize the global makespan (i.e., the time at which all jobs are finished), each organization is primarily interested in its completion time (i.e., the time at which all jobs provided by the organization are finished). If the completion time is later than its local execution, the organization receives no merit. Thus, MOSP aims to minimize the global makespan under the constraint that all organizations receive benefit (i.e., the completion time of each organization is not delayed compared to its local execution). Pascual et al. modeled a parallel computing environment as a set of identical multiprocessor machines and jobs as rigid parallel jobs [12]. In this setting, they proposed a 3-approximation algorithm¹ in a special case and a 4-approximation algorithm in the general case.

Our Contribution. In this paper, we consider a more generalized form of MOSP: We define it as the α -*Cooperative Multi-Organization Scheduling Problem* (abbreviated by α -MOSP)² by introducing parameter $\alpha \geq 1$, which represents the degree of cooperativeness. α -MOSP minimizes the global makespan under the constraint (called the *cooperation constraint*) that the completion time of each organization is bounded by α times of that in its local execution. When $\alpha > 1$, each organization is cooperative in the sense that it allows its completion time to be sacrificed for the global benefit (i.e., the shorter global makespan). Our objective is to reveal how the degree of cooperativeness α affects the global makespan of α -MOSP.

We model a parallel computing environment as unrelated parallel machines [12], instead of identical multiprocessor machines in [13]. This is because we want to treat the heterogeneity among computational resources in the environment. Actually, it is much more likely that organizations provide machines with different computational powers. In unrelated parallel machines, the time required for a machine to execute a job depends on both the job and the machine.

In the first part of this paper, we study the relation between the degree of cooperativeness α and the global makespan of α -MOSP. Let T^α be the minimum global makespan of α -MOSP, and T^* be the minimum global makespan under no cooperation constraint. First, we show that there exists an instance satisfying $T^\alpha/T^* \geq \max \left\{ \sum_{k=1}^m 1/\alpha^k, (\alpha+1)/\alpha \right\} - \epsilon$, where m is the number of organizations and ϵ is an arbitrary small positive constant. Note that, when $\alpha = 1$, we have $T^\alpha/T^* \geq m - \epsilon$. This implies that, when each organization does not allow its completion time to be delayed (i.e., the completely uncooperative case), it is impossible to bound reasonably the degradation of the global makespan. In contrast, when $\alpha > 1$, we still have possibility that we can construct the schedule which incurs only a small amount of degradation. As the second contribution, we show that such construction is actually possible. More precisely, we propose the algorithm which transforms a schedule S under no cooperation constraint into a schedule S' for α -MOSP. This algorithm guarantees

¹Let U be a minimization problem, and $cost$ be a cost function of U . An algorithm A for U is an α -approximation algorithm if $cost(A(x)) \leq \alpha \cdot cost(opt(x))$ holds for any instance x , where $A(x)$ is the output of the algorithm A for x and $opt(x)$ is an optimal solution for x [7].

² α -MOSP was introduced informally as a possible future work in [13]

that the global makespan of S' is at most $1 + \sum_{k=1}^m 1/\alpha^k$ times longer than that of S . This implies that weak cooperation improves the global makespan dramatically since $1 + \sum_{k=1}^m 1/\alpha^k \leq \alpha/(\alpha - 1)$ holds for $\alpha > 1$. Note that minimizing the global makespan under no cooperation constraint is equivalent to the traditional job scheduling problem $R||C_{\max}$.³ Thus, we can obtain approximation algorithms for α -MOSP by combining approximation algorithms for $R||C_{\max}$ with our transformation algorithm.

In the second part of this paper, we investigate the complexity of α -MOSP. First, we prove that α -MOSP is strongly NP-hard for any $\alpha \geq 1$. The proof is based on the reduction from $R||C_{\max}$. Since this reduction preserves the approximation ratio, we can easily obtain the inapproximability of α -MOSP for the approximation ratio less than $3/2$ from the inapproximability of $R||C_{\max}$. In addition, we can improve this lower bound for the case of $\alpha < 2$. We prove that, for any $\rho < (\alpha + 1)/\alpha$, there is no ρ -approximation algorithm of α -MOSP unless $P = NP$. Finally, we consider the complexity of transformation from unconstrained schedules to constrained ones. We show that, even if the optimal unconstrained solution is given, no polynomial-time algorithm can compute the optimal constrained solution for $\alpha < 2$ unless $P = NP$. This result is a witness for inexistence of general transformation algorithms that preserve the approximation ratio.

Related Work. In this paper, we aim to minimize the global makespan under the realistic assumption that each agent (organization) participates in the parallel computing unless it incurs a large increase of its completion time. In this sense, our results show the attainable global makespan when a certain kind of selfishness is introduced into agents. In what follows, we briefly summarize the literature about the makespan scheduling under selfishness.

Rzadca et al. [14] consider the cooperative computing formed by multiple organizations. In their cooperative computing, each organization, at some intervals, makes a decision whether it joins the cooperative computing or not. If an organization decides to join, the organization can use computational resources provided by other organizations but its own jobs may sometimes be delayed by other jobs. If an organization decides not to join, the organization has to compute its jobs by its local computational resources. Rzadca et al. construct a game-theoretic model to analyze the situation that each organization selfishly makes the decision and propose an algorithm that promotes cooperation among organizations.

Some researches model the scheduling problem as another game-theoretic model. In the model, there are some agents and some machines. Each agent produces a job and assigns it to some machine. Each machine schedules assigned jobs locally and executes them. Then, each agent knows the scheduling rule and how other agents assign jobs to machines. This means that each agent can selfishly select a machine which minimizes its completion time. Researches using this model focus on the Nash equilibrium, in which no agent can improve its completion time by unilaterally changing the machine. The main research topic is the coordination mechanism [6], which specifies how each machine schedules assigned jobs locally to attain the minimum makespan on the Nash equilibrium. To measure the quality of coordination mechanisms, the Price of Anarchy (PoA) [10] and the Price of Stability (PoS) [15, 3] are used. The PoA and the PoS are defined as the ratios of the global makespan on the worst and the best Nash equilibrium to that on the optimal situation, respectively. Note that the PoA guarantees the performance in the worst situation and the PoS shows the lower bound of the attainable performance. The PoA and the PoS of several methods are analyzed in [6, 8, 4] and in [1], respectively.

Other researches consider the model where agents can expand processing time of its job (by appending dummy computation) to minimize their completion time. For example, let us consider the LPT (Longest Processing Time first) rule, which is efficient for $P||C_{\max}$ and $Q||C_{\max}$. Since jobs

³The notations $P||C_{\max}$, $Q||C_{\max}$, and $R||C_{\max}$ denote the independent job scheduling problem on identical, uniform, and unrelated parallel machines, respectively [12].

with long processing time are scheduled earlier under the LPT rule, agents will expand processing time of their jobs. This selfish behavior of agents makes the LPT rule inefficient. In [2, 5], the authors propose truthful algorithms, which have the property that expanding a job never improves its completion time. From the property, truthful algorithms do not give agents the motivation to expand their jobs.

In the literature of selfish routing, some researches consider a problem similar to that in this paper. It is well known that, in networks with multiple users, a global-optimal flow may assign some users to considerably long routes. To overcome this dilemma, Jahn et al. [9] propose an algorithm to find a global-optimal flow under the user constraints that no user is assigned to a considerably long route. Schulz et al. [16] consider the same problem and give the theoretical bounds on the inefficiency and unfairness of a flow under the user constraints.

2 Preliminaries

2.1 Notations and the system model

We consider a computing environment formed by m independent organizations $\mathcal{O} = \{O_1, \dots, O_m\}$. Each organization O_k owns a machine M_k . Consequently, the computing environment contains m machines. The set of all machines is denoted by \mathcal{M} . Each organization O_k produces a set of jobs $\mathcal{J}_k = \{J_{k,1}, J_{k,2}, \dots\}$. We define $\mathcal{J} = \mathcal{J}_1 \cup \dots \cup \mathcal{J}_m$. The computing environment is modeled as unrelated parallel machines: The time required for M_k to execute $J_{i,j}$ is denoted by $t_{i,j}(M_k)$. We assume that all jobs are nonpreemptive.

A schedule S is defined as a set of job lists $L_S(M_k)$ ($1 \leq k \leq m$): Each job list $L_S(M_k) = (J_{i_1,j_1}, J_{i_2,j_2}, \dots)$ represents the sequence of jobs executed on M_k and its processing order.

For each $J_{i,j} \in \mathcal{J}$, we define $\rho_S(J_{i,j})$ as the machine that executes $J_{i,j}$ in a schedule S , and $\tau_S(J_{i,j})$ as the time at which the execution of $J_{i,j}$ is started in a schedule S . Note that $\tau_S(J_{i_\ell,j_\ell}) = \sum_{h < \ell} t_{i_h,j_h}(M_k)$ holds for any $L_S(M_k) = (J_{i_1,j_1}, J_{i_2,j_2}, \dots)$. We define $C_{\max}^S(O_k)$ as the time at which all jobs produced by O_k are finished, i.e., $C_{\max}^S(O_k) = \max\{\tau_S(J_{k,j}) + t_{k,j}(\rho_S(J_{k,j})) | J_{k,j} \in \mathcal{J}_k\}$. The time $C_{\max}^S(O_k)$ is called the completion time of O_k in a schedule S . We define $T_{\max}^S(M_k)$ as the time at which M_k finishes all the allocated jobs, i.e., $T_{\max}^S(M_k) = \max\{\tau_S(J_{i,j}) + t_{i,j}(M_k) | \rho_S(J_{i,j}) = M_k\}$. The (global) makespan of a schedule S is defined as $T_{\max}^S = \max\{T_{\max}^S(M_k) | M_k \in \mathcal{M}\}$.

2.2 Problem definition

The α -Cooperative Multi-Organization Scheduling Problem (abbreviated by α -MOSP) is the problem to minimize the makespan under the constraint that, for any $O_k \in \mathcal{O}$, the completion time of O_k is not delayed more than α times the local execution. To define α -MOSP formally, we introduce the local schedule loc , in which each O_k computes all jobs in \mathcal{J}_k on its own machine M_k . Precisely, the local schedule loc is defined as follows: $L_{loc}(M_k) = (J_{k,1}, J_{k,2}, \dots, J_{k,|\mathcal{J}_k|})$ holds for each $M_k \in \mathcal{M}$. Using the local schedule, we define feasible schedules for α -MOSP.

Definition 1 [α -feasible schedule] For a constant $\alpha \geq 1$, a schedule S is α -feasible if S satisfies the following conditions:

- Each job in \mathcal{J} is executed once in S .
- (Cooperation Constraint) For any O_k , $C_{\max}^S(O_k) \leq \alpha \cdot C_{\max}^{loc}(O_k)$ holds.

We say an α -feasible schedule S is α -optimal iff $T_{\max}^S \leq T_{\max}^{S'}$ for any α -feasible schedule S' . We define S_{opt}^α as an α -optimal schedule and T^α as the makespan of S_{opt}^α .

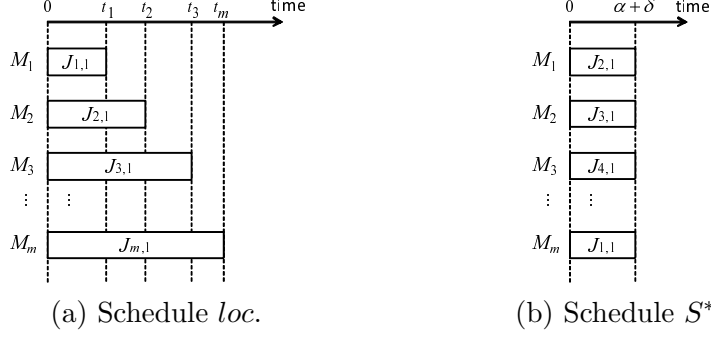


Figure 1: Schedules *loc* and S^* in Lemma 1.

Definition 2 [α -MOSP] α -MOSP is one to find an α -optimal schedule S_{opt}^α .

We define S^* as the ∞ -optimal schedule, that is, $S^* = S_{opt}^\infty$. We also define T^* as the makespan of S^* .

Note that the parameter α represents the degree of cooperativeness. When $\alpha = 1$, each organization is not cooperative in the sense that each organization does not allow its completion time to be delayed compared to the local schedule. When $\alpha = \infty$, each organization is unlimitedly cooperative, i.e., each organization aims to maximize the global benefit.

3 Effect of the Cooperativeness

In this section, we investigate how cooperativeness of organizations inherently affects the global makespan. As a result, we show that, for any $\alpha \geq 1$, there exists an instance satisfying $T^\alpha/T^* \geq \max\{A_m, (\alpha + 1)/\alpha\} - \epsilon$, where $A_m = \sum_{\ell=1}^m 1/\alpha^\ell$ and ϵ is an arbitrary small positive constant. When $\alpha = 1$, it follows that $T^\alpha/T^* \geq m - \epsilon$. This means that the global makespan in the completely uncooperative case can become m times longer than that under the unlimited cooperation.

In the rest, we show the above claim by two instances.

Lemma 1 For any α such that $1 \leq \alpha < 2$, there exists an instance satisfying $T^\alpha/T^* \geq A_m - \epsilon$ if $A_m > 1$ holds, where $A_m = \sum_{\ell=1}^m 1/\alpha^\ell$ and ϵ is an arbitrary small positive constant.

Proof Consider the following instance. For each O_k , $\mathcal{J}_k = \{J_{k,1}\}$. We define

$$t_{i,1}(M_k) = \begin{cases} t_k & \text{if } k = i \\ \alpha + \delta & \text{if } k \bmod m = i - 1 \\ \infty & \text{otherwise,} \end{cases}$$

where $t_k = \sum_{\ell=0}^{k-1} 1/\alpha^\ell$ and δ is an arbitrary small positive constant satisfying $0 < \delta < t_m - \alpha$.

We show schedules *loc* and S^* in Fig. 1 (a) and (b), respectively (Schedule S^* is clearly ∞ -optimal since the processing time of $J_{m,1}$ is at least $\alpha + \delta$). For this instance, we show $S_{opt}^\alpha = loc$: Focus on the job $J_{1,1}$. Since $C_{\max}^{loc}(O_1) = t_1 = 1$ and $t_{1,1}(M_k) > \alpha$ for $k \neq 1$, $J_{1,1}$ has to be executed on M_1 in α -feasible schedules. Next, focus on the job $J_{2,1}$. Since $C_{\max}^{loc}(O_2) = t_2 = 1 + 1/\alpha$, $J_{2,1}$ has to be executed on M_1 or M_2 in α -feasible schedules. However, since M_1 has to execute $J_{1,1}$ first, it cannot execute $J_{2,1}$ (if it does, the completion time of $J_{2,1}$ becomes $1 + \alpha + \delta > \alpha \cdot C_{\max}^{loc}(O_2)$). Thus, $J_{2,1}$ has to be executed on M_2 . Similarly, we can show that each job $J_{k,1}$ has to be executed on M_k , which

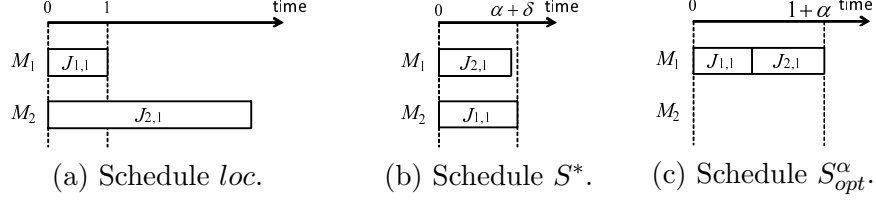


Figure 2: Schedules *loc*, S^* and S_{opt}^α in Lemma 2.

implies $S_{opt}^\alpha = loc$. From $T_{\max}^{S_{opt}^\alpha} = t_m = \alpha \cdot A_m$ and $T_{\max}^{S^*} = \alpha + \delta$, we obtain the lemma by using sufficiently small δ . \square

Lemma 2 For any $\alpha > 1$, there exists an instance satisfying $T^\alpha/T^* \geq (\alpha + 1)/\alpha - \epsilon$, where ϵ is an arbitrary small positive constant.

Proof Consider the instance such that $m = 2$, $\mathcal{J}_1 = \{J_{1,1}\}$, $\mathcal{J}_2 = \{J_{2,1}\}$, $t_{1,1}(M_1) = 1$, $t_{1,1}(M_2) = \alpha + \delta$, $t_{2,1}(M_1) = \alpha$, and $t_{2,1}(M_2) = \infty$, where δ is an arbitrary small positive constant. We can easily check $T_{\max}^{S_{opt}^\alpha} = \alpha + 1$ and $T_{\max}^{S^*} = \alpha + \delta$ (See Fig. 2). Consequently, we obtain the lemma by using sufficiently small δ . \square

Since $A_m < (\alpha + 1)/\alpha$ holds for any $\alpha \geq 2$, we have the following theorem from Lemma 1 and Lemma 2.

Theorem 1 There exists an instance satisfying $T^\alpha/T^* \geq \max\{A_m, (\alpha + 1)/\alpha\} - \epsilon$ for any $\alpha \geq 1$, where $A_m = \sum_{\ell=1}^m 1/\alpha^\ell$ and ϵ is an arbitrary small positive constant. \square

4 A transformation algorithm for α -MOSP

The instances proposed in the previous section demonstrate that it is impossible to bound the performance degradation caused by the cooperation constraint reasonably if each organization is completely uncooperative. However, it also remains the possibility that we can construct the schedule which satisfies the cooperation constraint but incurs only a small amount of performance degradation if each organization allows weak cooperation. Interestingly, we can show such construction is actually possible.

In this section, we present an algorithm, called TOMOS, which transforms a ∞ -feasible schedule S into a α -feasible schedule S' satisfying $T_{\max}^{S'} \leq B_m \cdot T_{\max}^S$, where $B_m = 1 + \sum_{k=1}^m 1/\alpha^k$. From algorithm TOMOS, we have $T^\alpha/T^* \leq B_m$. Note that, when $\alpha > 1$, it follows from $B_m < \alpha/(\alpha - 1)$ that $T^\alpha/T^* < \alpha/(\alpha - 1)$ holds. Remind that, when $\alpha = 1$, there exists an instance satisfying $T^\alpha/T^* \geq m - \epsilon$, where ϵ is a small positive constant. These results show that the global makespan is dramatically improved by weak cooperation.

By using algorithm TOMOS, if we have a ∞ -feasible schedule S satisfying $T_{\max}^S \leq \beta \cdot T^*$, we can obtain an α -feasible schedule S' satisfying $T_{\max}^{S'} \leq \beta B_m \cdot T^*$. Remind that the problem to find S^* (i.e., a ∞ -feasible schedule with the minimum makespan) is equivalent to $R||C_{\max}$. This problem was studied by many research groups, and several approximation algorithms were proposed [11, 17]. For example, Shchepin et al. [17] proposed an approximation algorithm which constructs a feasible schedule S satisfying $T_{\max}^S \leq (2 - 1/m) \cdot T^*$. Therefore, by combining Shchepin's algorithm with algorithm TOMOS, we can construct an $(2 - 1/m)B_m$ -approximation algorithm for α -MOSP, which outputs an α -feasible schedule S' satisfying $T_{\max}^{S'} \leq (2 - 1/m)B_m \cdot T^*$.

```

1: algorithm ToMOS( $S$ ) begin
2:    $\ell \leftarrow 0$ .
3:    $S_0 \leftarrow S$ .
4:   while there exists  $O_k$  such that  $C_{\max}^{S_\ell}(O_k) > \alpha \cdot C_{\max}^{loc}(O_k)$  do begin
5:     // Create  $S_{\ell+1}$  in which  $O_k$  first executes all its own jobs.
6:      $S_{\ell+1} \leftarrow S_\ell$ .
7:     For all  $M_{k'} \in \mathcal{M}$ , remove all jobs in  $\mathcal{J}_k$  from  $L_{S_{\ell+1}}(M_{k'})$ .
8:      $L_{S_{\ell+1}}(M_k) \leftarrow (J_{k,1}, J_{k,2}, \dots, J_{k,|\mathcal{J}_k|}) \bullet L_{S_{\ell+1}}(M_k)$ .
9:      $\ell \leftarrow \ell + 1$ .
10:  end
11:  output  $S_\ell$ .
12: end

```

Figure 3: Algorithm ToMOS. We denote by “ \bullet ” the concatenation of lists, i.e., for any $A = (a_1, \dots, a_{|A|})$ and $B = (b_1, \dots, b_{|B|})$, we have $A \bullet B = (a_1, \dots, a_{|A|}, b_1, \dots, b_{|B|})$.

In the following, we explain the detail of algorithm ToMOS. The behavior of algorithm ToMOS is very simple: If there exists an organization O_k whose completion time exceeds $\alpha \cdot C_{\max}^{loc}(O_k)$, the algorithm collects all jobs in \mathcal{J}_k to M_k . The algorithm repeats this process until the resultant schedule becomes α -feasible. We show the pseudo-code of algorithm ToMOS in Fig. 3.

Theorem 2 For any ∞ -feasible schedule S , algorithm ToMOS constructs S' such that $T_{\max}^{S'} \leq B_m \cdot T_{\max}^S$, where $B_m = 1 + \sum_{k=1}^m 1/\alpha^k$.

Proof First, we consider how many times the while-loop is repeated. Let us assume O_k is selected in the ℓ -th while-loop. Then, all jobs in \mathcal{J}_k are collected to M_k and first executed on M_k . Thus, for all $\ell' \geq \ell$, $C_{\max}^{S_{\ell'}}(O_k) = C_{\max}^{loc}(O_k)$ holds (Note that $S_{\ell'}$ is the schedule at the end of the ℓ' -th while-loop). This implies that each O_k is selected in the fourth line at most once. Consequently, the while-loop is repeated at most m times.

In the following, we assume algorithm ToMOS repeats the while-loop h times ($h \leq m$) and outputs $S' = S_h$. Clearly, the schedule S' is α -feasible. In the rest of the proof, we show $T_{\max}^{S'} \leq B_m \cdot T_{\max}^S$.

We prove the following proposition by induction.

(Proposition A) For any ℓ such that $0 \leq \ell \leq h$, $T_{\max}^{S_\ell} \leq B_\ell \cdot T_{\max}^S$ holds.

In the case that $\ell = 0$, Proposition A clearly holds. We assume that Proposition A holds in the case $\ell = \ell'$ ($0 \leq \ell' < \ell$). Let O_k be the organization selected in $(\ell' + 1)$ -th while-loop. Then, algorithm ToMOS collects all jobs in \mathcal{J}_k to M_k . While such collection does not increase $T_{\max}^{S_{\ell'}}(M_{k'})$ for any $k' \neq k$, it increases $T_{\max}^{S_{\ell'}}(M_k)$ by at most $C_{\max}^{loc}(O_k)$. Furthermore, no increase of assigned jobs occurs at M_k by the end of ℓ' -th while-loop because it can appear at most once only when O_k collects its own jobs. Thus, we have $T_{\max}^{S_{\ell'}}(M_k) \leq T_{\max}^S(M_k) \leq T_{\max}^S$. Consequently, it follows that $T_{\max}^{S_{\ell'+1}} \leq T_{\max}^S + C_{\max}^{loc}(O_k)$. Since O_k is selected in the fourth line of the algorithm, we obtain $C_{\max}^{S_{\ell'}}(O_k) > \alpha \cdot C_{\max}^{loc}(O_k)$, which implies $C_{\max}^{loc}(O_k) < (1/\alpha) \cdot T_{\max}^{S_{\ell'}} < (1/\alpha) \cdot B_{\ell'} \cdot T_{\max}^S$. Thus, we have $T_{\max}^{S_{\ell'+1}} < T_{\max}^S + (1/\alpha) \cdot B_{\ell'} \cdot T_{\max}^S = B_{\ell'+1} \cdot T_{\max}^S$. Therefore, Proposition A holds for $\ell = \ell' + 1$.

From Proposition A, $T_{\max}^{S_h} \leq B_h \cdot T_{\max}^S \leq B_m \cdot T_{\max}^S$ holds.

5 Complexity of α -MOSP

In this section, we present several results related to the complexity of α -MOSP.

5.1 Strongly NP-Hardness of α -MOSP

In this subsection, we prove that α -MOSP is strongly NP-hard for any $\alpha \geq 1$. When $\alpha = \infty$, α -MOSP is equivalent to $R||C_{\max}$. Thus, since $R||C_{\max}$ is strongly NP-hard [12], ∞ -MOSP is strongly NP-hard. In the following, we prove that α -MOSP is strongly NP-hard for any α ($\alpha \geq 1$). In the proof, we show the reduction from ∞ -MOSP (i.e., $R||C_{\max}$) to α -MOSP. For lack of space, the proof is omitted and shown in the appendix.

Theorem 3 α -MOSP is strongly NP-hard for any α ($\alpha \geq 1$). □

5.2 Inapproximability of α -MOSP

We can easily show that α -MOSP has the same lower bound for ∞ -MOSP because the reduction used in the proof of the theorem 3 preserves the approximation ratio. Since it is known that no polynomial-time algorithm for ∞ -MOSP can attain the approximation ratio less than $3/2$ under the assumption of $P \neq NP$, α -MOSP has the same lower bound for any α . In this subsection, we can improve this lower bound for α less than 2. More precisely, the following argument gives the lower bound for the approximation ratio $(\alpha + 1)/\alpha$ for any α .

The lower bound is proved by using a technique similar with [11], which is based on the reduction from the 3-dimensional matching problem.

Definition 3 Consider three disjoint sets $A = \{a_1, \dots, a_n\}$, $B = \{b_1, \dots, b_n\}$, $C = \{c_1, \dots, c_n\}$, and a family $F = \{T_1, \dots, T_m\}$ of triples with $|T_i \cap A| = |T_i \cap B| = |T_i \cap C| = 1$ for any i ($1 \leq i \leq m$). 3-dimensional matching problem is one to decide whether there exists a 3-dimensional matching F' , i.e., a subfamily $F' \subseteq F$ satisfying $|F'| = n$ and $\bigcup_{T_i \in F'} T_i = A \cup B \cup C$ or not.

It is well-known that the 3-dimensional matching is NP-complete. It is easy to decide the existence of 3-dimensional matching for any instance with $n \geq m$. Thus, in the following argument, we only consider the instances satisfying $n < m$. For any instance of the 3-dimensional matching problem, we construct an instance of α -MOSP with $2(n + m)$ machines. All machines are divided into three groups: the *triple group* $\mathcal{M}_F = \{M_1, \dots, M_m\}$, the *element group* $\mathcal{M}_{ABC} = \{M_{m+1}, \dots, M_{3n+m}\}$, and the *dummy group* $\mathcal{M}_D = \{M_{3n+m+1}, \dots, M_{2(n+m)}\}$. Each machine M_i in \mathcal{M}_F corresponds to the triple $T_i = (a_j, b_k, c_l)$. We say “the elements a_j , b_k and c_l are assigned to M_i ” if the triple corresponding to M_i is (a_j, b_k, c_l) . Each machine M_i in \mathcal{M}_{ABC} provides one *element job*, which has one-to-one correspondence to an element in $A \cup B \cup C$. The element corresponding to M_i ’s element job is denoted by $e(i)$. The processing time of M_i ’s element job is defined as follows:

$$t_{i,1}(M_k) = \begin{cases} \alpha/3 & \text{if } M_k \in \mathcal{M}_F \text{ and } e(i) \text{ is assigned to } M_k \\ \alpha + 1 & \text{if } M_k \in \mathcal{M}_F \text{ and } e(i) \text{ is not assigned to } M_k \\ 1 & \text{if } M_k \in \mathcal{M}_{ABC} \\ \alpha + 1 & \text{if } M_k \in \mathcal{M}_D, \end{cases}$$

Machines in \mathcal{M}_F and \mathcal{M}_D also provides *dummy jobs*. The triple group \mathcal{M}_F provides $2n + m$ jobs in total (the provider for each job is arbitrary), each of which have the following processing time:

$$t_{i,j}(M_k) = \begin{cases} \alpha + 1 & \text{if } M_k \in \mathcal{M}_F \\ \alpha & \text{otherwise,} \end{cases}$$

Each machine M_i in the dummy group provides one job with the following processing time:

$$t_{i,1}(M_k) = \begin{cases} \alpha & \text{if } M_k \in \mathcal{M}_F \\ \alpha + 1 & \text{if } M_k \in \mathcal{M}_{ABC} \\ 1 & \text{if } M_k \in \mathcal{M}_D \end{cases}$$

To distinguish the dummy jobs provided by \mathcal{M}_F and \mathcal{M}_D , we call the formers *dummy1 jobs* and the latters *dummy2 jobs*. For any instance I of the 3-dimensional matching problem, $R(I)$ denotes its corresponding instance of α -MOSP. Intuitively, the choice of a triple T_i in I implies that M_i processes all of three jobs corresponding to its assigned elements.

Lemma 3 An instance I of 3-dimensional matching problem is 'yes' instance $\iff R(I)$ has an α -feasible schedule with makespan α .

Proof (The proof of \implies) Suppose there exists a 3-dimensional matching F' for the given instance I . Then, following F' , we can dispatch all element jobs to at most n machines in \mathcal{M}_F . Then, every assigned machine takes exactly three element jobs. Since every element job is processed with $\alpha/3$ time units on the machines in \mathcal{M}_F , all of element jobs are completed by time α . This implies that all machines in \mathcal{M}_{ABC} satisfy the cooperation constraint because their processing times in the local schedule are one. After the assignment of the element jobs, exactly $m - n$ machines in \mathcal{M}_F receives no assignment of jobs. To those machines, we assign all of $m - n$ dummy2 jobs. Then, every dummy2 job is processed with α , and thus all machines in \mathcal{M}_D satisfy the cooperation constraint. Finally, each machine in $\mathcal{M}_{ABC} \cup \mathcal{M}_D$ takes one dummy1 job (remind that the number of dummy1 jobs is equal to $|\mathcal{M}_{ABC} \cup \mathcal{M}_D|$), which is also completed by time α . The providers of dummy1 jobs (that is, the machines in \mathcal{M}_F) need $\alpha + 1$ time units to process dummy1 jobs, they clearly satisfy the cooperation constraint. Consequently, the resultant schedule is α -feasible schedule with makespan α .

(The proof of \impliedby) Consider an α -feasible schedule S with makespan α . Then, only the machines in $\mathcal{M}_D \cup \mathcal{M}_{ABC}$ can process dummy1 jobs with time α , and thus every machine in $\mathcal{M}_D \cup \mathcal{M}_{ABC}$ has only one dummy1 job in S . This implies that all of element jobs and dummy2 jobs are processed on the machines in \mathcal{M}_F . Since the sum of processing times on the triple group \mathcal{M}_F for all element jobs and dummy2 jobs is αm , every machine in \mathcal{M}_F executes jobs until α in the schedule S . This implies that exactly n machines take three element jobs. The triples corresponding to such the machines are a 3-dimensional matching. \square

Theorem 4 Under the assumption of $P \neq NP$, there is no ρ -approximation algorithm of α -MOSP for any $\rho < (\alpha + 1)/\alpha$.

Proof Suppose a ρ -approximation algorithm \mathcal{A} for contradiction. From Lemma 3, we can prove the theorem by showing that \mathcal{A} returns the schedule with makespan α for the instance $R(I)$ reduced from any 'yes' instance I of the 3-dimensional matching problem. Then, the 3-dimensional matching problem is decidable in polynomial time, which contradicts $P \neq NP$.

Let $R(I)$ be the reduced instance, and S be the α -feasible schedule \mathcal{A} returns for the instance $R(I)$. By Lemma 3, $R(I)$ has an α -feasible schedule with makespan α . It implies that the makespan of S is lower than $\alpha + 1$. If $\alpha > 3$, the minimum granularity of job processing time in $R(I)$ is one. Thus, any

schedule with makespan more than α has the makespan more than or equal to $\alpha + 1$, which implies \mathcal{A} necessarily returns the schedule with makespan α . On the other hand, if $\alpha < 3$, the minimum granularity is $\alpha/3$. Thus, the makespan of S can be $4\alpha/3$ or $5\alpha/3$. Suppose for contradiction that S has makespan $4\alpha/3$ ($5\alpha/3$). Then, some machine $M_i \in \mathcal{M}_F$ takes one (two) element job(s) and exactly one dummy2 job. Let J be the last job processed by M_i . Then, the organization providing J cannot satisfy the cooperation constraint, which contradicts the α -feasibility of S , and thus S necessarily has the makespan α . The theorem is proved. \square

5.3 NP-completeness of Transformation

In this subsection, we show the hardness of the transformation from an ∞ -optimal schedule to an α -optimal schedule. In other words, α -MOSP is NP-complete even if the optimal solution in the unconstrained case is given.

The proof is based on the reduction from the 2-partition problem.

Definition 4 Consider the set of n nonzero values $C = \{c_1, \dots, c_n\}$. The 2-partition problem is one to decide whether a subset C' of C satisfying $\sum_{c_k \in C'} c_k = \sum_{c_k \notin C'} c_k$ or not.

The key idea of the proof is to reduce any instance I of the 2-partition problem to an instance $R(I)$ of α -MOSP for which the optimal solution can be computed in polynomial time if we omit the cooperation constraint. Then, if there exists a polynomial transformation algorithm, the 2-partition problem can be solved in polynomial time, and it contradicts $P \neq NP$.

In what follows, we explain the construction of $R(I)$ from the instance I of the 2-partition problem. Without loss of generality, we assume that I satisfies $\sum_{k=1}^n c_k = 2(4 + \alpha)/3$ (i.e., the instance is scaled by $2(4 + \alpha)/3$). Two machines M_1 and M_2 exist in $R(I)$. The machine M_1 provides one dummy job, whose processing time is one on M_1 and $2(\alpha + 1)/3$ on M_2 . The machine M_2 provides n jobs $\{J_{2,1}, \dots, J_{2,n}\}$, each of which corresponds to one value in C . We define their processing times as $t_{2,k}(M_2) = c_k$ and $t_{1,k}(M_1) = c_k(1 + \alpha)/(4 + \alpha)$. For instance $R(I)$, we can show the following lemmas. For lack of space, the proof is omitted and shown in the appendix.

Lemma 4 For any $\alpha < 2$, an instance I of the 2-partition problem is an 'yes' instance $\iff R(I)$ has the α -feasible schedule with makespan $(\alpha + 4)/3$.

Lemma 5 For any $\alpha < 2$, an ∞ -optimal schedule for $R(I)$ can be computed in polynomial time.

The above two lemmas imply the following theorem:

Theorem 5 Under the assumption of $P \neq NP$, there is no polynomial-time algorithm that transforms an ∞ -optimal solution to an α -optimal solution for any $\alpha < 2$. \square

6 Conclusion

This paper has investigated the tradeoff between the global and local benefit in multi-organization cooperative computing. To analyze the tradeoff we have introduced α -MOSP, which minimizes the global makespan under the cooperative constraint that the completion time of each organization is bounded by α times of that in its local execution. Note that the parameter α represents the degree of cooperativeness. In the first part of this paper, we have analyzed the lower and upper bound of the global makespan for each cooperativeness α . As a result, we have shown that it is impossible to bound reasonably the degradation of the global makespan for no cooperation case (i.e., $\alpha = 1$),

however, in contrast, it is possible to do so for weak cooperation (i.e., $\alpha > 1$). In the second part, we have shown some complexity results of α -MOSP. These results are witnesses for inexistence of general polynomial-time transformation algorithms that preserve the approximation ratio.

We have some directions in our future works. In the theoretical direction, we would like to study the tighter lower and upper bounds. It is also an interesting problem to find the lower and upper bounds on more restrictive computation models such as uniform parallel machines. In the practical direction, we would like to propose an algorithm which constructs a schedule in decentralized and dynamic manner. Such an algorithm is indispensable to apply α -MOSP to massive-scale cooperative computing such as grid computing.

References

- [1] E. Angel, E. Bampis, and F. Pascual. The price of approximate stability for scheduling selfish tasks on two links. In *Proceedings of the 12th International Euro-Par Conference*, pages 157–166, 2006.
- [2] E. Angel, E. Bampis, and F. Pascual. Truthful algorithms for scheduling selfish tasks on parallel machines. *Theoretical Computer Science*, 369:157–168, 2006.
- [3] E. Anshelevich, A. Dasgupta, J. Kleinberg, É. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 295–304, 2004.
- [4] Y. Azar, K. Jain, and V. Mirrokni. (almost) optimal coordination mechanisms for unrelated machine scheduling. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 323–332, 2008.
- [5] G. Christodoulou, L. Gourvès, and F. Pascual. Scheduling selfish tasks: About the performance of truthful algorithms. In *Proceedings of the 13th Annual International Computing and Combinatorics Conference*, pages 187–197, 2007.
- [6] G. Christodoulou, E. Koutsoupias, and A. Nanavati. Coordination mechanisms. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming*, pages 345–357, 2004.
- [7] J. Hromkovič and W. M. Oliva. *Algorithmics for Hard Problems*. Springer-Verlag New York, Inc., 2002.
- [8] N. Immorlica, L. Li, V. Mirrokni, and A. Schulz. Coordination mechanisms for selfish scheduling. In *Proceedings of the 1st Workshop on Internet and Network Economics*, pages 55–69, 2005.
- [9] O. Jahn, R. H. Möhring, A. S. Schulz, and N. E. Stier-Moses. System-optimal routing of traffic flows with user constraints in networks with congestion. *Operations Research*, 53:600–616, 2005.
- [10] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413, 1999.
- [11] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46(3):259–271, 1990.
- [12] J. Y-T. Leung, editor. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, 2004.

- [13] F. Pascual, K. Rzdca, and D. Trystram. Cooperation in multi-organization scheduling. In *Proceedings of the 13th International Euro-Par Conference*, pages 224–233, 2007.
- [14] K. Rzdca and D. Trystram. Promoting cooperation in selfish computational grids. *European Journal of Operational Research*, in Press.
- [15] A. S. Schulz and N. Stier. On the performance of user equilibria in traffic networks. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 86–87, 2003.
- [16] A. S. Schulz and N. E. Stier-Moses. Efficiency and fairness of system-optimal routing with user constraints. *Networks*, 48:223–234, 2006.
- [17] E. V. Shchepin and N. Vakhania. An optimal rounding gives a better approximation for scheduling unrelated machines. *Operations Research Letters*, 33:127–133, 2005.

A Appendix: Proof of Lemmas and Theorems

A.1 Proof of Theorem 3

Proof We show the reduction from ∞ -MOSP to α -MOSP. Consider an instance I of ∞ -MOSP. In I , we consider m organizations $\mathcal{O} = \{O_1, \dots, O_m\}$, and each O_k provides jobs $\mathcal{J}_k = \{J_{k,1}, J_{k,2}, \dots\}$. Let $t_{\min} = \min\{t_{i,j,k} \mid J_{i,j} \in \mathcal{J}, M_k \in \mathcal{M}\}$, where $t_{i,j,k}$ is the time required for M_k to execute job $J_{i,j}$ in I .

From the instance I of ∞ -MOSP, we construct an instance $R(I)$ of α -MOSP. In $R(I)$, we consider $m+1$ organizations $\mathcal{O} = \{O_1, \dots, O_{m+1}\}$. In $R(I)$, O_{m+1} provides all jobs in I , and each machine M_k ($1 \leq k \leq m$) executes the jobs at the same speed as I . We assign ∞ to the time required for M_{m+1} to execute the jobs. Each organization O_k ($1 \leq k \leq m$) provides a job J_k , whose processing time is t_{\min}/m on M_{m+1} and ∞ on other machines.

By the setting, in efficient schedules of $R(I)$, each job provided by O_{m+1} has to be executed on some M_k ($1 \leq k \leq m$), and each job provided by O_1, \dots, O_m has to be executed on M_{m+1} . Note that such schedules do not violate the cooperation constraint of α -MOSP. The makespan of such schedules depends on the schedule for each machine M_k ($1 \leq k \leq m$), which is the same as the schedule for M_k in I . Thus, the problem to find an ∞ -optimal schedule in I is equivalent to the problem to find an α -optimal schedule in $R(I)$. Therefore, we have the theorem. \square

A.2 Proof of Lemma 4

Proof (The proof of \implies) Suppose a solution C' of the instance I . Then, we consider the α -feasible schedule S where the dummy job and the jobs whose corresponding values are in C' are executed on M_1 (other jobs are executed on M_2). Then, from the fact that C' is a 2-partition, we can easily see that both M_1 and M_2 finishes the assigned jobs at time $(\alpha+4)/3$ in S .

(The proof of \impliedby) Suppose an α -feasible schedule S with makespan $(\alpha+4)/3$. Since $2(\alpha+1)/3 > \alpha$ holds, the execution of the dummy job on M_2 violates the cooperation constraint. Thus, the dummy job is executed on M_1 in S . Then, we can show that M_1 finishes all assigned jobs exactly at time $(\alpha+4)/3$: If M_1 finishes at time $(\alpha+4)/3$ or earlier, M_2 takes more than $(\alpha+4)/3$ time units to complete all assigned jobs. This implies that the set of values corresponding the jobs assigned to M_1 is a 2-partition of C . \square

A.3 Proof of Lemma 5

Proof We show that the schedule S where all jobs are swapped between M_1 and M_2 is the ∞ -optimal schedule. In S , both M_1 and M_2 finish the computation exactly at time $2(\alpha+1)/3$. Thus, at the schedule S , any transfer of jobs from M_1 to M_2 does not improve the makespan. This implies that S has the shortest makespan of all schedules where the dummy job is executed on M_2 . By the same argument as the proof of Lemma 4, we can also show that the makespan becomes larger than or equal to $(\alpha+4)/3$ as long as the dummy job is executed on M_1 . This implies that S is the optimal because $2(\alpha+1)/3 < (\alpha+4)/3$ holds for any $\alpha < 2$. \square