Efficiency and Equity in the Multi Organization Scheduling Problem.

Martin Durand and Fanny Pascual

Sorbonne Université, CNRS, LIP6, 4 place Jussieu, 75005 Paris, France.
martin.durand@lip6.fr, fanny.pascual@lip6.fr

Abstract

We consider the multi organization scheduling problem (MOSP) [20]: given N organizations owning, each of them, one set of tasks and machines, the aim is to compute a schedule which gathers all the tasks on all the machines, and such that the makespan is minimized. A rationality constraint must be fulfilled: no organization should increases its makespan (the completion time of its last task) compared to the case where it schedules its own tasks (and only its own tasks) on its own machines. We show that cooperation (sharing machines and tasks) can benefit to all the organizations simultaneously, since they may decrease their makespans by a factor of N. We present an algorithm which is $(1+\epsilon)$-approximate, while the makespan of each organization is increased by a factor at most $(1+\epsilon)$. We also study to which extent the rationality constraint (or a relaxed constraint) increases the makespan, compared to problem $(P||C_{\max})$ where there is no such a constraint. Finally, we introduce a new problem, which focus on equity: the aim is to return a schedule which fulfills the rationality constraint and which maximizes the factor by which each organization has decreased its makespan. We give an optimal algorithm for this problem in a particular case, and show that it is NP-hard and hard to approximate in the general case. We complete this paper by an efficient heuristic for this problem.

1 Introduction

Cost constraints, as well as environmental issues, make the sharing of machines between independent organizations (such as laboratories or universities) a very interesting solution. Sharing machines allow organizations which need to execute tasks to use the machines of organizations which do not need machines at this time, decreasing the completion time of the tasks without having to invest in new machines. But cooperation is even more than sharing unused machines with organizations who need to schedule tasks: cooperation can benefit simultaneously several organizations which all have tasks to compute, by allowing a better placement of the tasks, as we will see in the sequel. The Multi Organization Scheduling Problem (MOSP) [20] deals with several organizations which each owns both a set of identical parallel machines and a set of sequential tasks to execute. The objective is to minimize the completion time of the last task completed on the machines shared by the organizations, given that no organization should increase the completion time of its tasks in the shared system, compared to the case where it executes its own tasks on its own machines. This last constraint is called the *rationality constraint*, and ensures that all the organizations have incentive to share their machines.

Besides analyzing the best possible benefit that organizations can mutually have by sharing their machines, our aim is to focus on the *efficiency* of algorithms (where the efficiency is thought in term of makespan – the date at which all the tasks have been computed), and on the *equity* of algorithms for MOSP (it is not suitable that, even if the returned schedule fulfills the rationality constraint, the machines which are free are used only for the tasks of a single organization while some tasks of the other organizations are waiting). These two aspects may be antagonist, and our aim is to see to which extent, since what we want would be a schedule with a small makespan and in which machines are shared with equity. We will start by reviewing existing work on MOSP, and continue by presenting our results and the map of this paper.

Related work.

The Multi Organization Scheduling Problem [20,21] has been introduced with parallel rigid tasks (tasks that need to be executed in parallel on several machines) and has mainly been studied from an approximation viewpoint. The best approximate algorithm is a 3-approximation algorithm when the organizations schedule locally the tasks in decreasing order of their heights (the height of a task is the number of machines needed to execute the task), or a 4-approximation algorithm in the general case [10]. For sequential tasks (tasks that need to be executed on one machine only), the best known algorithm is a 2-approximate algorithm [6] (in the sequel, all the papers – as well as our results – deal with sequential tasks). Note that all these bounds are not only approximation ratios, since they are in fact upper bounds of the ratio, in the worst instance, $\frac{C_{\max}(\mathscr{S})}{OPT^{\bar{r}}}$, where $C_{\max}(\mathscr{S})$ is the makespan in a solution returned by the algorithm and $OPT^{\bar{r}}$ is the smallest possible makespan that can be obtained by scheduling the same set of tasks on the same set of machines (this last schedule does not necessarily fulfill the rationality constraint). Lower bounds on such a ratio have also been given: it has been proved that there is no algorithm with a ratio smaller than 2 when the tasks are parallel [21], or when the tasks are sequential and when it is required that, in the returned schedule, the machines of each organization schedule their own tasks before scheduling the tasks of other organizations [6,8]. A lower bound of $\frac{3}{2}$ is known in the general case for sequential tasks [6,20] – we will improve this bound in the sequel.

Several variants have been studied: for example, organizations choose themselves on which machines to schedule their tasks knowing that each machine schedule the tasks of its organization first [8]. Despite most works deal with minimizing the overall makespan while each organization wishes to minimizes its own makespan, other objectives have also been studied: the aim can be to minimize the average completion time of tasks [6,8], or the energy needed to schedule the tasks [5]. These papers usually show that the problem is NP-hard and then give approximation algorithms or heuristics.

Some papers also consider a relaxed version of MOSP: it is assumed that the organizations tolerate a bounded degradation on the makespan of their own tasks, and the aim is to minimize the global makespan. This problem is denoted by $(1+\alpha)$-MOSP [17] when it is assumed that each organization accepts to increase the maximum completion time of its tasks by a factor at most $(1+\alpha)$. A $\frac{3}{2}$-approximate algorithm for 2-MOSP has been given [9]. Other work include additional constraints on the machines [4]. The closest work in spirit to what we will do in Section 4 is a study of $(1+\alpha)$-MOSP on unrelated machines [17,18]. In this setting, Ooshita et al. show that, when there is no cooperation ($\alpha = 0$), the makespan can be m times higher than in the optimal makespan without the rationality constraint. When $\alpha > 0$, the authors also give a $(2+\frac{2}{\alpha})$-approximate algorithm for $(1+\alpha)$-MOSP.

There is few work on fairness issues when some organizations own tasks and machines. In an experimental work, Cohen et al. [8] look at the fairness (using stretch and Jain Index) of schedules returned by some algorithms, and they show that the best results are obtained by algorithm

ILBA [21]. In another work, Skowron and Rzadca [22] model the fair scheduling problem as a cooperative game and use the Shapley value to determine an ideal fair schedule. To calculate the contribution of an organization, they determine how the presence of this organization influences the performance of other organizations. For unit-size tasks they give a fully polynomial-time randomized approximation scheme, and they show this problem is NP-hard and hard to approximate in the general case. Other work about fairness in scheduling are mainly about how to schedule tasks of different users on a set of shared machines (tasks belong to different users, but the machines are not owned by the users) [1], or when different users have different preferences on the order in which shared tasks will be executed on a same machine [19].

Our results, and map of the paper.

In this paper, we consider that N organizations $O_1, \dots, O_N$ share m machines, and that each organization O_i has its own set of tasks $\mathscr{T}_i$. Each organization O_i wishes to minimize its makespan, i.e. the date at which all its tasks (the tasks of $\mathscr{T}_i$) have been completed. If each organization O_i schedules its own tasks (and only its owns tasks) on its own machines, these tasks are completed at a date which will be called the *local makespan* of O_i. We have two objectives, which can be antagonists. First, we would like to return a schedule which is as efficient as possible, and thus which minimizes the global makespan, i.e. the date at which all the tasks have been scheduled. Second, we would like to return a fair schedule. Our results are as follows.

In Section 3 we show that cooperation can permit to decrease the makespan of each organization by a factor N (but no more). This shows that cooperation can benefit to all the organizations simultaneously, and not only to some organizations which own many tasks or few machines. In this section, we also give a polynomial time algorithm with resource augmentation: for a fixed $\epsilon > 0$, and a fixed number of organizations, it returns a solution $(1+\epsilon)$-approximate in which each organization has a makespan at most $(1+\epsilon)$ times its local makespan.

In Section 4, we relax the rationality constraint by considering $(1+\alpha)$-MOSP: we assume that each organization agrees to complete its last task at a date at most $(1+\alpha)$ times its local makespan. We are interested by the tradeoff between the value of α and the value of the (global) makespan. We first show that an algorithm which returns schedules which minimize the makespan can have to increase a local makespan by a factor $m-1$, which is certainly unacceptable for the agents. We then focus on the ratio than can be obtained for the global makespan, for a fixed α: we give a lower bound of the necessary increase of the makespan in $(1+\alpha)$-MOSP with respect to the optimal makespan without the rationality constraint. If $\alpha = 0$, $(1+\alpha)$-MOSP corresponds to MOSP (no organization should get a makespan higher than its local makespan). In this case, the obtained lower bound shows that it is not possible to get algorithm which outputs 2-approximate schedules for the makespan and which fulfills the rationality constraint. This improves the lower bound of $\frac{3}{2}$ given in [6, 20].

In Section 5, we define the gain of an organization as the ratio between its local makespan over its makespan in the schedule returned. Since we want to fulfill the rationality constraint, this gain will be at least 1, but the higher this gain is, the higher an organization will be satisfied by the schedule returned. We are interested by getting fair schedules: we introduce the problem which consists in returning schedules which maximize the minimal gain of an organization. For the unit tasks case, we give a polynomial time optimal algorithm for this problem. For the general case, we show that this problem is NP-complete, and even hard to approximate, and we give an heuristic which outputs, in practice, schedules close to the optimal ones.

We conclude this work by giving a few research direction in Section 6. Before starting to present our technical results, we start, in Section 2, by introducing notations and defining formally our problem.

2 Preliminaries

2.1 Notations

By $\mathscr{O} = \{O_1, ..., O_N\}$ we denote the set of N independent organizations sharing m machines $\{1, \ldots, m\}$ and n tasks. Each organization O_i, with $i \in \{1, \ldots, N\}$ owns $m_i \geq 1$ machines, and a set $\mathscr{T}_i$ of $n_i \geq 0$ tasks. If $n_i > 0$, these tasks are denoted by $t_i^1, \ldots, t_i^{n_i}$. Tasks are sequential: each task n_i^j is executed on a single machine, during a processing time (also called length) $l_i^j > 0$. Machines are identical. We denote by $m = \sum_{i=1}^{N} m_i$ the total number of machines, and by $n = \sum_{i=1}^{N} n_i$ the total number of tasks. We denote by $\mathscr{T}$ the set of all the tasks.

Given a task j, and a considered schedule, we denote by C_j the *completion time* of task j, i.e. the date at which its execution ends. Preemption is not allowed: once a task starts to be executed, it will be executed until its completion.

MOSP takes as input the local schedules of the organizations. The local schedule of Organization O_i is a schedule of the n_i tasks of O_i on the m_i machines of O_i. This schedule may minimize the makespan of O_i, or not (this problem is indeed NP-hard [11]): Organization O_i gives its local schedule to a central entity which will return a schedule which fulfills the rationality constraint. We will denote by $\mathscr{S}_{loc}^i$ the *local schedule* of Organization O_i, and we will denote by C_{loc}^i the makespan of this schedule (this will be called the *local makespan* of O_i).

Given a schedule $\mathscr{S}$ of the n tasks on the m machines, we will denote by $C_i(\mathscr{S})$ the completion time of the last task of Organization O_i in $\mathscr{S}$. We will also call $C_i(\mathscr{S})$ the *makespan of Organization* O_i in $\mathscr{S}$. Given a schedule $\mathscr{S}$ we will denote by $C_{\max}(\mathscr{S})$ the completion time of the last task in $\mathscr{S}$. Therefore $C_{\max}(\mathscr{S}) = \max_{i \in \{1,\ldots,N\}} C_i(\mathscr{S})$, and is called the *makespan* of $\mathscr{S}$.

Given $i \in \{1, \ldots, m\}$, we will denote by $L_i(\mathscr{S})$ the *load of machine* i in Schedule $\mathscr{S}$: this is the sum of the processing times of the tasks assigned to machine i in $\mathscr{S}$. The *total load* of a schedule is the sum of the processing times of all the tasks of the schedule ($\sum_{i=1}^{N} \sum_{j=1}^{n_i} l_i^j$).

2.2 Problem statement

The objective of each organization O_i is to minimize $C_i(\mathscr{S})$, the date at which all its tasks are completed in the returned schedule $\mathscr{S}$. The multi-organization scheduling problem (MOSP) consists in scheduling the n tasks of all the organizations, on the m machines of the organizations, in order to minimize the makespan of the returned schedule with the additional constraints that no organization has a makespan larger than the makespan of its local schedule:

$$\text{minimize } C_{\max}(\mathscr{S}) \text{ such that, for each } i \in \{1, \ldots, N\}, C_i(\mathscr{S}) \leq C_{loc}^i.$$

The set of these additional constraints is called the *rationality constraint*: it ensures that each organization will have incentive to accept the schedule returned by the central entity (or trusted third party), since it will not be able to get a better makespan if it schedules its own tasks on its own machines.

Given an instance I, we will denote by $\mathscr{S}^*$ an optimal solution for MOSP, and we will denote by OPT the makespan of such a solution. In the sequel, we will also be interested to compare

OPT, or the makespan returned by an algorithm, to the best solution without the rationality constraint, that we will denote $\mathscr{S}^{*(\bar{r})}$. In such a solution, all the tasks are scheduled on all the machines in order to minimize the makespan: this is an optimal solution of the classical scheduling problem $(P||C_{\max})$. We will also denote by $OPT^{\bar{r}}$ the makespan $\mathscr{S}^{*(\bar{r})}$.

3 Interest of cooperation and algorithm

In this section, we measure to what extent cooperation can reduce the makespans of organizations, with respect to a schedule made of the local schedules only. We will show in Section 3.1 that on some instances, cooperation can decrease simultaneously all the makespans. In Section 3.2, we present an algorithm which returns a schedule whose makespan is at most $(1+\epsilon)$ times the makespan of an optimal schedule of MOSP, while the makespan of each organization in this schedule is at most $(1+\epsilon)$ times its local makespan.

3.1 Cooperation can decrease all the makespans

If the local makespan of Organization O_i is much larger than the local makespan of the other organizations, then, by load balancing tasks of O_i on the machines of all the organizations, the makespan of O_i may decrease a lot. In this section, we show that MOSP is more than load balancing tasks of heavy loaded organizations on the machines of less loaded organizations: there are instances for which all the organizations can simultaneously benefit of cooperation. Proposition 1 shows that all the organizations may together reduce their makespans up to a factor N by cooperating with each other.

Proposition 1. *In an optimal schedule for MOSP, all the organizations may decrease simultaneously their makespans up to a factor N, with respect to their local makespans (which are assumed to be optimal). This is the best possible bound : there is no instance where each organization can decrease its makespan by a factor larger than N.*

Proof. Let us first exhibit an instance where each organization improves its makespan by a factor as close as wished of N. We consider an instance where each of the N organizations owns a single machine (therefore $m = N$). For each $i \in \{1, \dots, N\}$, Organization O_i owns mx^{i-1} tasks of length 1 (thus Organization O_1 owns m tasks, while Organization O_N owns mx^{N-1} tasks). The local makespan of Organization O_i is therefore mx^{i-1}. Let us now consider the following schedule, $\mathscr{S}$, optimal for MOSP : on each machine, there are one task of Organization O_1, followed by x tasks of Organization O_2, followed by x^2 tasks of Organization O_3, and so forth. The schedule ends on each machine with x^{N-1} tasks of O_N. For each $i \in \{1, \dots, N\}$, the makespan of O_i in $\mathscr{S}$ is $1 + \sum_{j=2}^{i} x^{j-1}$. Therefore, each organization O_i decreases its makespan, from $\mathscr{S}^i_{local}$ to $\mathscr{S}$, by a factor $\frac{C^i_{loc}}{C_i(\mathscr{S})} = \frac{mx^{i-1}}{1+\sum_{j=2}^{i} x^{j-1}}$. This tends towards to $m = N$ when x tends towards to the infinity.

Let us now show that there is no instance where cooperation can make each organization decrease its makespan by a factor larger than N. By contradiction, let us assume that there exists an instance I for which there is a schedule $\mathscr{S}$ in which the makespan of each organization is decreased by a factor larger than N with respect to its local makespan (assumed to be optimal). Note that there is in I at least one organization O_i such that $m_i \geq \frac{m}{N}$ (otherwise, we would have $\sum_{i=1}^{m} m_i < m$). By hypothesis, the makespan of O_i in $\mathscr{S}$ is $C_i(\mathscr{S}) < \frac{C^i_{loc}}{N}$. We now show that this implies that it is possible for O_i to obtain a schedule of its tasks on its machines with makespan smaller than C^i_{loc}.

Indeed, let us consider the following schedule of the tasks of O_i on m_i machines : compute between time 0 and $C_i(\mathscr{S})$ the tasks scheduled in $\mathscr{S}$ on the first m_i machines, and then the tasks scheduled in $\mathscr{S}$ on machines $m_i+1, 2m_i$ between time $C_i(\mathscr{S})$ and $2C_i(\mathscr{S})$, etc. (tasks scheduled in $\mathscr{S}$ on machines $(x-1)m_i+1, xm_i$ are scheduled between time $(x-1)C_i(\mathscr{S})$ and $xC_i(\mathscr{S})$, as they are scheduled in $\mathscr{S}$: a task starting at time t on machine j will be scheduled at time $t \mod C_i(\mathscr{S})$, on machine $(j \mod m_i)$ if $j \mod m_i \neq 0$ and on machine m_i otherwise). This schedule is a feasible schedule of makespan at most $NC_i(\mathscr{S}) < C^i_{loc}$. Therefore, the local schedule of O_i was not optimal, a contradiction. □

3.2 A PTAS with resource augmentation

In this section, we show that the polynomial approximation scheme (PTAS) presented by Hall and Shmoys [12] for a scheduling problem can be used to get a PTAS with resource augmentation for our problem. More precisely: given a fixed $\epsilon > 0$, and a fixed number of organizations N, we will get a polynomial time algorithm which returns a schedule with a makespan at most $(1+\epsilon)OPT$, and in which the makespan of each organization is at most $(1+\epsilon)$ times its local makespan. The rationality constraint may thus be violated, but the increase of the makespans of the organizations is bounded, and may be acceptable if ϵ is small. Let us start by presenting the scheduling problem studied by Hall and Shmoys.

Scheduling problem with delivery times (SchedDT). The input of this problem consists in n_{DT} tasks $\{1, \ldots, n_{DT}\}$ and m_{DT} identical machines. Each task j has a processing time p_j (it must be processed without interruption for time p_j on any one of the m_{DT} machines), a release date r_j (the date at which it becomes available for processing), and a delivery time q_j. Each task's delivery begins immediately after its processing has been completed, and all tasks may be delivered simultaneously. Therefore, for a given schedule in which task j starts at time σ_j, the completion time of task j is defined as $C_j = \sigma_j + p_j + q_j$. The aim is to minimize, over all possible schedules, the makespan $C_{\max} = \max_{j \in \{1,\ldots,n\}} C_j$. In the sequel, we will denote this problem as SchedDT. As noted by Hall and Shmoys, this problem is equivalent to the scheduling problem with release dates (r_j) and due dates (d_j) – and without delivery times – in which the objective is to minimize the maximum lateness, where the lateness of task j is $L_j = \sigma_j + p_j - d_j$. This last problem is denoted as $(P|r_j|L_{max})$, using Graham's notation for scheduling problems. However, while this problem is inapproximable in polynomial time if $P \neq NP$, there exists a PTAS for SchedDT. Let us now give a high level description of this PTAS, that we will use for our problem is the sequel. The details can be found in the original paper [12].

High level description of the PTAS for SchedDT. This PTAS is a generalization of the PTAS of Hochbaum and Shmoys [13] for problem $(P||C_{\max})$. The principle of Hall and Shmoys's algorithm is the following one. It assumes that there are a lower bound LB and an upper bound UB of the optimal makespan OPT_{DT} of SchedDT, such that $LB \leq OPT_{DT} \leq UB \leq 2OPT_{DT}$. It then does a dichotomic search with a target value T on this interval: for each target value, the algorithm either builds a schedule of makespan at most $T(1+\epsilon)$, or it assures that there is no schedule of makespan at most T. At the end of the dichotomic search, the schedule found with the smallest value of T which lead to a feasible schedule is returned.
Before this, a preprocessing step consists in rounding the input: the releases dates are rounded down to obtain a fixed number of distinct ones. The same thing is done for delivery times.

Tasks are partitioned into two sets: large tasks (tasks whose processing times are larger than or equal to a given number δ function of ϵ), and small tasks (smaller than δ). Large tasks are rounded down so that there is a fixed number of different processing time for the large tasks. Given that, for large tasks, there are a fixed number of different values of q, r and p, there is now a fixed number τ_1 of different types of large tasks. Small tasks are "glued" into small components of size δ and of common values r and q (once these values have been rounded): there is now a fixed number τ_2 of different types of small components (which gather small tasks). Let X be the set of possible types of tasks ($|X| = \tau_1 + \tau_2$). A machine configuration indicates, for each type of task $t \in X$ how many tasks of type t are on the machine. Given the size of the large tasks, we can upper bound the maximum number of large tasks per machine in a schedule with a makespan smaller than $2OPT_{DT}$ and show that the number of relevant machine configurations is fixed (let us denote by γ this number). For a given schedule, x_l indicates the number of machines with configuration l: vector $x = (x_1, \dots, x_\gamma)$ defines an outline for the schedule. Therefore, the number of relevant outlines is at most m^γ, a polynomial in m. The order of tasks on machine is based on a generalization of the Jackson's rule [15], a polynomial time optimal algorithm for $(1||L_{max})$ (where the aim is to minimize the maximum lateness on a single machine), when there are release date. This problem, $(1|r_j|L_{max})$, is solved in polynomial time [12]. The algorithm tries every relevant outline. If at least a schedule with makespan at most T is found, the algorithm outputs the best schedule – a schedule, with rounded tasks of makespan at most T. When the tasks take back their true values, this becomes a schedule of makespan at most $(1+\epsilon)T$. By doing a dichotomic search over T, this algorithm returns a $(1+\epsilon)$-approximate solution for the SchedDT [12]. Let us now see how we can use it to get a PTAS with resource augmentation for MOSP.

Algorithm for our problem. Let I be an instance of MOSP, and T an integer (T will be a target makespan). We will create an instance $I'(T)$ of SchedDT from I and T in the following way. We fix $n_{DT} = n$ and $m_{DT} = m$. For task t_i^j (the j-th task of Organization i), which is of length l_i^j in I, we create in $I(T)'$ a task t_k, with $k = (\sum_{x=1}^{i-1} n_x) + i$ (i.e. to each task of I is associated a task in $I'(T)$). We set: $p_k = l_i^j$, $r_k = 0$, and $q_k = \max\{0, T - C_{loc}^i\}$. The idea is the following one: tasks are available at date 0, and a task of Organization O_i should be scheduled before the local makespan of O_i, C_{loc}^i. Whereas the lengths will be rounded, we will not round down the values q in the PTAS if the number of organizations is fixed (in this case, there will be a fixed number of sizes q – at most N, the number of organizations –, and this will allow us to better bound the deterioration of the local makespans of the organizations). Once this reduction has been done, we use the above described PTAS of Hall and Shmoys with instance $I'(T)$ (the only differences between the original PTAS and our utilization of it is that the values q are not rounded – if N is fixed –, and that the instance $I'(T)$ slightly differs at each step of the dichotomic search since the values q are a function of T).

We do a dichotomic search over the target makespan T in the interval $[LB, UB]$, where $LB = \max\left\{\max_{i,j} l_i^j, \frac{\sum_{i=1}^N \sum_{j=1}^{n_i} l_i^j}{m}\right\}$ and UB is the makespan of the schedule returned by a greedy 2-approximate algorithm for MOSP [8] (UB is the makespan of a schedule without idle times, so we have $UB \leq \frac{\sum_{i=1}^N \sum_{j=1}^{n_i} l_i^j}{m} + \max_{i,j} l_i^j \leq 2LB$). Note that $\max_{i,j} l_i^j$ and $\frac{\sum_{i=1}^N \sum_{j=1}^{n_i} l_i^j}{m}$ are lower bounds of OPT (since $\max_{i,j} l_i^j$ is the length of the longest task, and $\frac{\sum_{i=1}^N \sum_{j=1}^{n_i} l_i^j}{m}$ is the average load of a machine), and thus $LB = \max\left\{\max_{i,j} l_i^j, \frac{\sum_{i=1}^N \sum_{j=1}^{n_i} l_i^j}{m}\right\}$ is a lower bound of OPT. Let us denote ApproxViaDT(ϵ) this algorithm.

Proposition 2. *Let $\epsilon > 0$. If the number of organizations is fixed, Algorithm* ApproxViaDT*(ϵ) returns a schedule of makespan at most $(1+\epsilon)OPT$ in which each organization $i \in \{1, \dots, N\}$ has a makespan at most $(1+\epsilon)C^i_{loc}$.*

Proof. Let us first show that Algorithm ApproxViaDT(ϵ), returns a schedule of makespan at most $(1+\epsilon)OPT$.

Let us consider an instance I of MOSP, and let us denote by OPT the value of its optimal makespan. Let us consider a target makespan T examined at a given step of the dichotomic search. For this value T, the algorithm either returns a schedule of value $T(1+\epsilon)$, or assures that there is no schedule of value at most T. If $T = OPT$, then, $OPT_{DT} \leq OPT$, where OPT_{DT} is the makespan of an optimal solution of instance $I'(T)$ of the scheduling problem with delivery times. Indeed, let us consider an optimal schedule for MOSP, and let us view it from the viewpoint of SchedDT. For each task k of Organization O_i, $q_k = \max\{0, OPT - C^i_{loc}\}$. In a feasible schedule for MOSP, the execution of this task k will end at most at time C^i_{loc}, and thus its completion (as defined in the scheduling problem with delivery times) will be at most at time $C^i_{loc} + q \leq OPT$. Therefore, there is a feasible schedule of makespan OPT for instance $I'(OPT)$.

Note that, during the dichotomic search of ApproxViaDT(ϵ), if there is a solution for instance $I'(T)$ for problem SchedDT then there is no solution for instances $I'(T')$ with $T' < T$ (by construction) and there are solutions for instances $I'(T')$ with $T' > T$ since a solution for instance T will be a solution for instance T' (the values q increase at most by $T' - T$, while the makespan also). Therefore, ApproxViaDT(ϵ), which returns a schedule $(1+\epsilon)$-approximate for SchedDT, will return a solution of makespan at most $(1+\epsilon)OPT$.

Let us now show that in the returned solution, the makespan of each organization is at most $(1+\epsilon)C^i_{loc}$. Recall that the makespan of the schedule returned by the PTAS for SchedDT (for the final target makespan T) is at most $(1+\epsilon)T \leq (1+\epsilon)OPT$. Recall also that the values q have not been rounded, and that the value q of a task of O_i is 0 if $C^i_{loc} > T$ and $T - C^i_{loc}$ otherwise. The schedule of the tasks of types in X (tasks with rounded sizes, or small tasks glued into small components) has a makespan at most T. The factor $(1+\epsilon)$ is obtained when we replace these tasks by the tasks with their real lengths. Since the values q have not been rounded down, a task of O_i will end, when considering the schedule with the rounded sizes, at time at most C^i_{loc} if $T < C^i_{loc}$, and at most $T - q = C^i_{loc}$ otherwise: in both cases, its execution ends at time at most C^i_{loc}. By replacing the tasks of X by the true tasks, each completion time may be increased by factor $(1+\epsilon)$. Therefore, we obtain a schedule in which the execution of each task of O_i ends at most at time $(1+\epsilon)C^i_{loc}$. This concludes the proof. □

Note that, if the number of organizations is not fixed, we can use the same algorithm, by rounding the values q (as in the original PTAS for SchedDT). This will return a schedule of makespan at most $(1+\epsilon)OPT$ and in which each organization has a makespan at most $C^i_{loc} + \epsilon OPT$.

4 Efficiency vs. increase of the local makespans

In this section, we study how the aim of minimizing the makespan is in opposition with the rationality constraint. We start, in Section 4.1, to show that if we want to return a schedule optimal for the makespan, then we may have to increase the local makespans up to a factor $m-1$. Since it is unlikely that the organizations agree to increase their local makespan of such a large factor, in Section 4.2, we assume that each organization agrees to increase its makespan by a factor $(1+\alpha)$, with $\alpha \geq 0$. We then look at the increase of the makespan in function of α (when $\alpha = 0$, the problem is MOSP, the higher α is, the more relaxed the rationality constraint is).

Note that, contrarily to what we have done in Section 3.2, in this section, we compare the makespan of an optimal solution of $(1+\alpha)$-MOSP to the optimal makespan *without the rationality constraint*, $OPT^{\bar{r}}$. The algorithm of Section 3.2 returned a schedule close to OPT, the optimal solution of $MOSP$, but not necessarily close to $OPT^{\bar{r}}$ (this can be very different, since, as we will see in the sequel, OPT, can be twice larger than $OPT^{\bar{r}}$).

4.1 The aim is to minimize the makespan: impact on the local makespans.

Let us first consider instances made of two organizations in which each organization owns one machine ($m=2$). In this case, the makespan of the best schedule of MOSP minimizes the global makespan: $OPT = OPT^{\bar{r}}$. Indeed, let us assume without loss of generality that $C^1_{loc} \leq C^2_{loc}$, and let us consider $\mathscr{S}^{*(\bar{r})}$, an optimal schedule for $(P||C_{\max})$ for such an instance. In $\mathscr{S}^{*(\bar{r})}$, let us schedule on each machine the tasks of O_1 before the tasks of O_2. By construction, this schedule minimizes the makespan; O_2 does not increase its makespan (otherwise the local schedules would have a makespan smaller than OPT, which is not possible); and O_1 does not increase its makespan neither since its jobs are at the beginning of the schedule on each machine.

This is the best case: the rationality constraint does not prevent from obtaining the best schedule concerning the makespan. This is however not always the case when $m > 2$. The following proposition shows that, in order to get a schedule minimizing the makespan, an organization may have to increase its makespan up to a factor $m-1$.

Proposition 3. *In a schedule which minimizes the makespan of the tasks of $\mathscr{T}$ on m machines, an organization may necessarily increase its makespan up to a factor $m-1$ (compared to its local makespan), but never up to a factor larger than m. This holds even if there are two organizations.*

Proof. Let us consider an optimal schedule of the tasks $\mathscr{T}$ for problem $(\mathrm{P}||C_{\max})$. In this schedule, we reorder the tasks such that on each machine the tasks are scheduled by increasing number of their organizations (i.e. tasks of O_1 are scheduled before the one of O_2, and so forth). Let us denote by $\mathscr{O}$ the schedule obtained. This schedule stays an optimal schedule since the load on each machine, and thus the makespan, do not change. Let us show that for each $i \in \{1, \ldots, n\}$, $C^i_{\mathscr{O}} \leq mC^i_{loc}$. Let us consider a given machine j and a task x of O_i on machine j. If it is not the first task on machine j, task x is preceded by tasks of $\{O_1, \ldots, O_i\}$ on M_j. The load of the tasks which precede the x (plus the length of x) is thus at most $\sum_{k=1}^{i} m_k C^k_{loc}$ (since the load of each organization O_k is at most $m_k C^k_{loc}$). Since the organizations are indexed by non decreasing local makespans, $\sum_{k=1}^{i} m_k C^k_{loc} \leq \sum_{k=1}^{i} m_k C^i_{loc} \leq mC^i_{loc}$. The completion time of each task of O_i in $\mathscr{O}$ is at most mC^i_{loc}. Therefore $C^i_{\mathscr{O}} \leq mC^i_{loc}$.

Let us now output an instance in which an organization has to increase its makespan up to a factor $m-1$. Consider the instance with two organizations, where O_1 has $m-1$ machines and $m-1$ tasks of length 1 (its local makespan is thus 1), and where O_2 has $m-1$ tasks of length $m-1$ and 1 machine. An optimal schedule of these tasks on m machines has a makespan of

$m-1$. Indeed, in such a schedule, the tasks of O_1 are necessarily scheduled on the same machine and are completed at time $m-1$: the makespan of O_1 is increased by a factor $m-1$. $\square$

Note that the bound of $m-1$ can be increased up to m if the organizations are allowed to own tasks but no machine. The instance showing this is almost the same than the one in the proof above (O_1 owns m machines and m tasks of length 1 and O_2 has $m-1$ tasks of length m).

We have seen that what we could call "the price of efficiency", the factor at which a local makespan may have to increase to get an optimal schedule for the makespan, is between $m-1$ and m, which is high. We can assume that organizations may accept to increase their makespans in order to get an efficient schedule, but only if this does not increase to much their makespans. In the following section, we assume that each organization agrees to increase a little bit its makespan: it will accept a schedule in which its makespan is increased by a factor at most $(1+\alpha)$ compared to its local makespan.

4.2 The aim is to minimize the increase of the local makespans: impact on the makespan.

Let $\alpha \geq 0$. We now assume that each organization O_i agrees to have a makespan at most equal to $(1+\alpha)C^i_{loc}$. If $\alpha = 0$, this is the MOSP. Otherwise, it means that each organization agrees to increase a little bit its makespan (the higher α is, the higher an organization agrees to increase its makespan). We call $(1+\alpha)$-MOSP, the problem where we wish to minimize the makespan with these relaxed constraints:

$$\text{minimize } C_{\max}(\mathscr{S}) \text{ such that, for each } i \in \{1,\dots,N\}, C_i(\mathscr{S}) \leq (1+\alpha)C^i_{loc}.$$

Our aim is to give a lower bound on the approximation ratio of an algorithm for $(1+\alpha)$-MOSP with respect to the optimal makespan $OPT^{\bar{r}}$: this will show what we loose, in term of makespan, due to the relaxed rationality constraint.

Proposition 4. *Let $\alpha \geq 0$, $\varepsilon > 0$. If each organization accepts to increase its makespan by a factor $(1+\alpha)$, there is no* $(\max_{k \in \{\lfloor \sqrt{\frac{\alpha m^2+m}{1+\alpha}} \rfloor, \lceil \sqrt{\frac{\alpha m^2+m}{1+\alpha}} \rceil\}} \left(1 + \frac{(m-k)(k(1+\alpha)-m\alpha-1)}{k(m-1)}\right) - \varepsilon)$-*approximate algorithm with respect to the global makespan.*

Proof. Given m machines, and $k \in \{1,\dots,m-1\}$, let us consider the following set of tasks: k tasks of length $xk(m-1)$ (these tasks are said *large*) and $n_{small} = (m-1)xk(m-k)$ tasks of length 1 (these tasks are said *small*). The optimal makespan of these tasks is $OPT = xk(m-1)$: it is obtained when each large task is alone on a machine, and the small tasks are scheduled on the $(m-k)$ remaining machines.

Let us now assume that Organization O_1 owns $m-1$ machines and all the small tasks, and that Organization O_2 owns one machine and all the large tasks. The local makespan of O_1 is then $C^1_{loc} = xk(m-k) \leq OPT$, and the local makespan of O_2 is $C^2_{loc} = xk^2(m-1) \geq OPT$.

Let $\mathscr{S}$ be a schedule in which each organization increases its makespan by a factor at most $(1+\alpha)$. In $\mathscr{S}$, each task of O_1 (small task) is completed at the latest at time $\lfloor(1+\alpha)C^1_{loc}\rfloor = \lfloor(1+\alpha)xk(m-k)\rfloor$. Therefore, on $m-k$ machines, there are at most $\lfloor(1+\alpha)xk(m-k)\rfloor$ tasks of length 1, and the other small tasks are on the k remaining machines. The minimal number of small tasks to schedule on the k remaining machines is $n_{small} - (m-k)\lfloor(1+\alpha)xk(m-k)\rfloor =$

$(m-1)xk(m-k)-(m-k)\lfloor(1+\alpha)xk(m-k)\rfloor$. On one of these k machines, there is at least $1/k$ of these tasks, that is $(m-1)x(m-k)-\frac{(m-k)\lfloor(1+\alpha)xk(m-k)\rfloor}{k} \geq (m-1)x(m-k)-(1+\alpha)x(m-k)^2$. If there are at least two large tasks on the same machine, the makespan is at least equal to $2(xk(m-1)) = 2OPT$. Otherwise, there are at most one large task by machine. The makespan of such a schedule is then at least the length of a large task plus the length of the small tasks. This is larger than or equal to $xk(m-1)+(m-1)x(m-k)-(1+\alpha)x(m-k)^2$. The approximation ratio is thus at least $\frac{xk(m-1)+(m-1)x(m-k)-(1+\alpha)x(m-k)^2}{xk(m-1)} = 1+\frac{(m-k)(k(1+\alpha)-m\alpha-1)}{k(m-1)}$.

By deriving $f(k) = 1+\frac{(m-k)(k(1+\alpha)-m\alpha-1)}{k(m-1)}$ (with $k \in [1,+\infty)$), we find that the value of k which maximizes $f(k)$ is $k = \sqrt{\frac{\alpha m^2+m}{1+\alpha}}$.

Since $f(k)$ is an increasing function between $[1, \sqrt{\frac{\alpha m^2+m}{1+\alpha}}]$ and a decreasing function in $[\sqrt{\frac{\alpha m^2+m}{1+\alpha}}, +\infty)$, the maximum value of $f(k)$ when k is an integer is:

$$\max_{k\in\{\left\lfloor\sqrt{\frac{\alpha m^2+m}{1+\alpha}}\right\rfloor, \left\lceil\sqrt{\frac{\alpha m^2+m}{1+\alpha}}\right\rceil\}} \left(1+\frac{(m-k)(k(1+\alpha)-m\alpha-1)}{k(m-1)}\right).$$

□

When $\alpha = 0$, the value of k which maximizes the ratio ($f(k)$) is $\lceil\sqrt{m}\rceil$ or $\lfloor\sqrt{m}\rfloor$. When $\sqrt{m}$ is an integer, there is no algorithm for MOSP which returns $\left(1+\frac{m-2\sqrt{m}+1}{m-1}-\varepsilon\right)$-approximate schedules with respect to the global makespan. This tends towards 2 when m tends towards the infinity, which lead to the following corollary.

Corollary 1. *Let $\epsilon > 0$. There is no algorithm which returns schedules which fulfill the rationality constraint, and which is $(2-\epsilon)$-approximate with respect to the global makespan $OPT^{\bar{r}}$.*

This bound improves the previous one, $\frac{3}{2}$, which had been given by Pascual et al. [20] for two organizations and by Cohen et al. [6] for more than two organizations. Furthermore, in [8] the authors show that no approximation algorithm for MOSP has a ratio asymptotically better than 2 w.r.t. the global makespan (when m tends towards the infinity) when we add the constraint that on the returned schedule, each machine schedules the tasks of its organization (if any) before the tasks of other organizations. This constraint is thus not necessary to obtain the asymptotic ratio of 2.

When m tends towards the infinity and $\alpha > 0$ the value of k maximizing $f(k)$ is then $m\sqrt{\frac{\alpha}{\alpha+1}}$. In that case we can express the approximation ratio depending on only α as $2+2\alpha-(\alpha+1)\sqrt{\frac{\alpha}{\alpha+1}}-\frac{\alpha}{\sqrt{\frac{\alpha}{\alpha+1}}}$. The value $\sqrt{\frac{\alpha}{\alpha+1}}$ quickly increases with α and tends towards 1 when α tends towards the infinity. This means that this ratio is close to 2 when α is close to 0, and it quickly decreases and tends towards 1.

Figure 4.2 shows the lower bound given in Proposition 4. This ratio is given as a function of α (Left), or of the number of machines, m (Right). The higher m is, the higher the ratio is. When α increases, this ratio decreases quickly. The first points of the curves in Figure 4.2 Left shows the lower bound of the ratio between the best makespan in a schedule satisfying the rationality constraint, and the best makespan without this constraint (as seen above, this ratio tends towards 2 when m increases). This ratio when $\alpha = 0$ can also be seen in the blue curve of Figure 4.2 Right.

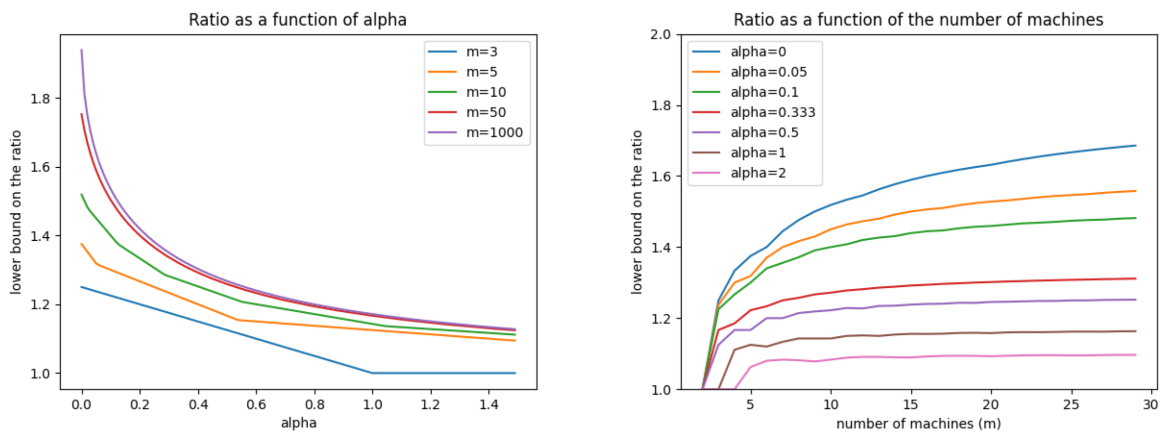


Figure 1: Each organization accepts to decrease its makespan by a factor $(1+\alpha)$. Lower bound on the ratio between the best possible makespan when no organization increases its makespan by a factor larger than $(1+\alpha)$, and the optimal makespan.

We end this section by mentioning that we can easily adapt the algorithm described in Section 3.2 to the case of $(1+\alpha)$-MOSP: whereas, for a target makespan T, we had set the delivery time of a task of Organization O_i to $q = \max\{0, T - C^i_{loc}\}$ (so that this task is completed at time C^i_{loc} in the returned schedule of rounded tasks), we fix this value to $q = \max\{0, T - (1+\alpha)C^i_{loc}\}$ in the case of $(1+\alpha)$-MOSP. We thus get, for any fixed $\epsilon > 0$, a polynomial time algorithm returning a schedule of makespan at most $(1+\epsilon)$ times the makespan of an optimal solution of $(1+\alpha)$-MOSP, and in which the makespan of each organization is at most $(1+\epsilon)(1+\alpha)$ its local makespan.

In the previous sections, we have assumed either that the rationality constraint should be fulfilled (but we then had as only objective function to minimize the makespan, and the gains for the organizations – the decrease of their makespans – in the returned schedule could be very different), or we have even assumed than we can relax (in a bounded way) the rationality constraint to get a schedule with an even smaller makespan. In the following section, we focus on fairness issues: we will keep the rationality constraint, and our focus will not be to decrease the makespan, but to get schedule in which *all* the organizations decrease their makespans by a factor as large as possible.

5 Max Min Gain

5.1 Problem statement

Let us first define the gain $g_i(\mathscr{S})$ of Organization O_i in a schedule $\mathscr{S}$: $g_i(\mathscr{S})$ represents how much Organization O_i has decreased its makespan in the schedule $\mathscr{S}$ in comparison to its local schedule:

$$g_i(\mathscr{S}) = \frac{C^i_{loc}}{C_i(\mathscr{S})}.$$

The Maximal Minimal Gain problem, denoted as MaxMinGain, takes the same input as MOSP. It builds a schedule of the n tasks of all the organizations on the m machines of the organizations, in order to maximize the minimum gain among the organizations. The returned schedule is thus $\mathscr{S} = \arg\max_{\mathscr{S}} \min_{i \in \{1,\dots,N\}} g_i(\mathscr{S})$

Note that the schedule $\mathscr{S}$ which is made of N local schedules has a minimum gain of 1, which means that an optimal schedule for MaxMinGain always has a minimum gain larger than or equal to 1 and satisfies the rationality constraint.

Given a considered instance I, we will denote by $\mathscr{S}^*$ an optimal solution for MaxMinGain, and we will denote by OPT the minimum gain among the organizations in such a solution. In the sequel, we will also be interested to compare the makespan $C_{\max}(\mathscr{S}^*)$, or the makespan returned by an algorithm, to the best solution without the rationality constraint, that we will denote by $\mathscr{S}^{*(\bar{r})}$.

Let us note that it is also possible to define the gain of an organization O_i in a schedule $\mathscr{S}$ not as $\frac{C^i_{loc}}{C_i(\mathscr{S})}$ (multiplicative case) but as $C^i_{loc} - C_i(\mathscr{S})$ (additive case). Note that this second definition is very close to the definition of utility used in a recent paper by Agnetis et al. [2]. In the work of Agnetis et al., two agents, each one owning a subset of tasks, share a single machine. The two agents A and B have different objective functions f^A and f^B. The utility of agent A in a schedule S is defined as $f^A_\infty - f^A(S)$, where f^A_∞ denotes the value of f^A when the subset of tasks of A is scheduled after the subset of tasks owned by B, which is the worst case for A. Even though the contexts are different, the idea is the same: we evaluate individual satisfaction by comparing a worst case for the agent to the current solution. In our case, for each organization O_i, we compare the makespan obtained by O_i in a schedule to the worst makespan O_i could have, and this worst makespan is its local makespan, C^i_{loc}, since we fulfill the rationality constraint. All the results in Section 5 are proved for the multiplicative case, but can be proved in a similar way for the additive case.

5.2 Case of unit tasks

In this section, we show that problem MaxMinGain can be solved in polynomial time when all the tasks have the same length. Moreover, in this case, it is possible to find a schedule $\mathscr{S}$ which is optimal for MaxMinGain and optimal for problem $(P||C_{\max})$: the global makespan is minimized while the minimal gain of an organization is maximized. Let us now present the following algorithm which returns such a schedule. This algorithm, called LS-IM, is a list scheduling algorithm: it greedily schedules all the tasks, considering the tasks by increasing local makespans of their owners:

```
Sort the organizations by non decreasing local makespans. If two organizations have the
  same local makespan, sort them by non decreasing number of tasks. Let $O_{x_1}, \dots, O_{x_N}$
  be the result of this sort.
for i=1 to N do
    for each task $t^j_{x_i} \in \mathscr{T}_{x_i}$ do
        schedule $t^j_{x_i}$ on the first available machine;
    end
end
```

Algorithm 1: List scheduling by increasing local makespans (Algorithm LS-IM)

Proposition 5. *When all the tasks have the same length, Algorithm* LS-IM *returns schedules which are optimal for* MaxMinGain *and optimal for* $(P||C_{\max})$.

Proof. Let us assume that the organizations are labelled such that $C_{loc}^{1} \leq C_{loc}^{2} \leq \cdots \leq C_{loc}^{N}$ and that, for all $l \in \{2, \ldots, N\}$, if $C_{loc}^{l} = C_{loc}^{l-1}$, then $n_l \geq n_{l-1}$. We also assume all the tasks have the same length.

Let us suppose, for the sake of contradiction, that the schedule $\mathscr{S}$ returned by algorithm LS-IM is not optimal for MaxMinGain. Let O_k be an organization which get a minimal gain in $\mathscr{S}$. In order to increase the gain of O_k, we should build a schedule $\mathscr{S}'$ in which $C_k(\mathscr{S}') < C_k(\mathscr{S})$. Tasks all have the same length and there is no idle time in $\mathscr{S}$: there is not enough slots so that all the tasks of $O_1, \ldots, O_k$ are completed before time $C_k(\mathscr{S})$. Therefore, in $\mathscr{S}'$, a task of O_l, with $l < k$ will be completed at time at least $C_k(\mathscr{S})$. The gain of O_l in $\mathscr{S}'$ will thus be at least $\frac{C_{loc}^{l}}{C_k(\mathscr{S})} \leq \frac{C_{loc}^{k}}{C_i(\mathscr{S})}$: the minimal gain in $\mathscr{S}'$ is larger than or equal to the minimal gain in $\mathscr{S}$. Therefore, $\mathscr{S}$ is optimal for MaxMinGain.

Schedule $\mathscr{S}$ has no idle time and all the tasks are the same length, therefore if a task t_i^j starts at time t in $\mathscr{S}$, all machines are busy at least until t, which means that at least one tasks has to start at t or later: the schedule $\mathscr{S}$ is also optimal for $(P||C_{\max})$. □

We showed that, in the particular case of unit tasks, we can find a polynomial time algorithm which builds schedules which both minimize the global makespan and maximize the minimal gain of an organization. In this special case we do not have to compromise between global optimization and individual satisfaction. Unfortunately, this result does not hold in the general case, as we will see in the following section.

5.3 General case

In this section, we study MaxMinGain in the general case. We fist show that MaxMinGain is strongly NP-hard and hard to approximate.

Proposition 6. *If $P \neq NP$, problem* MaxMinGain *is NP-hard and inapproximable in polynomial time, even if there are only two organizations and two machines.*

Proof. Let $r > 1$. By contradiction, let us assume that $P \neq NP$ and that there exists a polynomial time r-approximate algorithm for MaxMinGain. We will show that this algorithm allows us to solve the NP-complete Partition problem. The Partition problem is the following one: given a set $S = \{a_1, \ldots, a_k\}$ of k positive integers such that $\sum_{i=1}^{k} a_i = 2B$, is it possible to partition S into two subsets S_1 and S_2 such that $\sum_{a_i \in S_1} a_i = \sum_{a_i \in S_2} a_i = B$?

We will exhibit an instance for which the maximum minimal gain is strictly greater than 1 if and only if there is a yes answer to the Partition problem. Note first that, if this is true, then our r-approximate algorithm allows us to solve the Partition problem. Indeed, if there is a yes answer to the Partition problem then the maximal minimal gain is $OPT > 1$: a r-approximate algorithm should return a solution in which the gain of each organization is a least $rOPT > 1$. If the answer to the Partition problem is 'no' then $OPT = 1$, and any algorithm, including the r-approximate algorithm, will return a solution with minimal gain 1. Therefore, the r-approximate algorithm permits to determine whether the answer to the partition problem is positive or not. Since this r-approximate algorithm is a polynomial time algorithm, this implies that $P = NP$, a contradiction.

Let us now consider the following instance of MaxMinGain, and show that, for this instance, there is a yes answer to the Partition problem if and only if the maximum minimal gain, OPT, is strictly greater than 1. There are two organizations, each one having a single machine. Organization O_1 owns k tasks $t_1^1, \ldots, t_k^1$ such that for each $i \in \{1, \ldots, k\}$, the length of task t_i^1

is equal to a_i. Organization O_2 owns 2 tasks, each of length $B+1$. The local makespan of O_1 is thus $2B$, while the local makespan of O_2 is $2B+2$.

Let us first consider that answer of the Partition problem is 'yes'. Therefore, there exists a partition (S_1, S_2) of the tasks of O_1 such that $\sum_{t_i^1 \in S_1} t_i^1 = \sum_{t_i^2 \in S_2} t_i^2 = B$. By scheduling the tasks of S_1 followed by a task of O_2 on a machine, and the tasks of S_2 followed by the second task of O_2 on the second machine, the makespan of O_1 is B, while the makespan of O_2 is $B + (B+1) = 2B+1$. Since the local makespan of O_1 is $2B$ and the local makespan of O_2 is $2B+2$, both organizations have a gain strictly greater than $1(O_1$ decreases its makespan by a factor 2, and O_2 by a factor $\frac{B+2}{B+1}$).

Let us now consider that the answer of the Partition problem is 'no'. Let us first notice that organization O_2 can have a gain greater than 1 only if there is one of its tasks on each machine (otherwise its makespan will be at least its local makespan, $2B+2$). In this case, there is a set S_1 of tasks of O_1 with the first task of O_2, and a set S_2 of tasks of O_1 with the second task of O_2. Since the answer of the Partition problem is 'no', any partition (S_1, S_2) of the tasks of O_1 is such that the weight of a subset is at least $B+1$ (each t_i^1 corresponds to $a_i \in \mathbb{N}$). Therefore, the load of a machine is at least $2(B+1) = 2B+2$. The last task on such a machine should be a task of O_2 (since the local makespan of O_1 is smaller than $2B+2$), and thus the makespan of O_2 is at least $2B+2$, its local makespan: this organization has a gain smaller than 1.

We have shown that the minimal gain is strictly greater than 1 if and only if the answer of the Partition problem is 'yes': this concludes the proof. □

Let us now show that MaxMinGain is strongly NP-hard. This implies that there is no pseudo-polynomial algorithm to solve it.

Proposition 7. *Problem* MaxMinGain *is strongly NP-hard, even if there are only two organizations.*

Proof. Let us reduce the NP-complete problem 3-Partition to the decision version of MaxMinGain. The 3-Partition problem is the following one: given a set $S = \{a_1, \ldots, a_{3k}\}$ of $3k$ positive integers such that $\sum_{i=1}^{3k} a_i = kB$, with $B \in \mathbb{N}$, is it possible to partition S into k subsets $\{S_1, \ldots, S_k\}$ such that for each $i \in \{1, \ldots, k\}, \sum_{a_j \in S_i} a_j = B$? Our problem is the following one: given the local schedules of n organizations, and given a bound $X \in \mathbb{Z}$, with $X < 1$, is is possible to create a schedule of all the tasks on all the machines such that the makespan of each organization is at most X times its makespan in its local schedule? The instance of MaxMinGain corresponding to the instance of Partition is the following one: there are two organizations, O_1 and O_2. Organization O_1 owns one machine and $3k$ tasks $t_1, \ldots, t_{3k}$ such that for each $i \in \{1, \ldots, 3k\}$, the length of task t_i is equal to a_i. Organization O_2 owns $k-1$ machines and k tasks, each of length kB. We set $X = \frac{k+1}{2k}$.

Let us show that there is a yes answer to the Partition problem if and only if the answer of the corresponding instance of MaxMinGain is also 'yes'.

Let us first consider that the answer of the Partition problem is 'yes': there exists a partition $(S_1, \ldots, S_k)$ of the tasks of O_1 such that for each $i \in \{1, \ldots, k\}, \sum_{a_j \in S_i} a_j = B$. For each $i \in \{1, \ldots, k\}$, by scheduling on machine i the tasks corresponding to the numbers of S_i followed by a task of O_2, the makespan of O_1 is B, while the makespan of O_2 is $B + kB = (k+1)B$. Since the local makespan of O_1 is kB and the local makespan of O_2 is $2kB$, the makespan of each organization in the constructed schedule is at most equal to $X = \frac{k+1}{2k}$ times its local makespan: the answer of the decision problem of MaxMinGain is 'yes'.

Let us now consider that the answer of the decision problem of MaxMinGain is 'yes'. The makespan of each organization is at most $X = \frac{k+1}{2k}$ its local makespan, that is $\frac{k+1}{2k}kB = \frac{(k+1)B}{2}$ for O_1 and $\frac{k+1}{2k}2kB = (k+1)B$ for O_2. Thus the global makespan is at most $(k+1)B$. Therefore, there is necessary one task of O_2 on each of the k machines. Since the global makespan is at most $(k+1)B$ and since the total load is $(k+1)kB$, then there is necessarily a load of $(k+1)B$ on each of the k machines. The load due to the tasks of O_2 on each machine is kB, so, on each machine, the load due to the tasks of O_1 is B. It is therefore possible to partition the numbers $t_1, \dots, t_k$ into k sets of weight B: the answer of the 3-Partition problem is 'yes'. $\square$

We showed that when the tasks have the same lengths, there is always a schedule which is both optimal for MaxMinGain and for the minimization of the makespan (problem $(P||C_{\max})$). It is easy to note that, in the general case, we can obtain an optimal solution $\mathscr{S}^*$ of MaxMinGain that is also 2-approximate for $(P||C_{\max})$. Since every schedule with no idle time is 2-approximate for $(P||C_{\max})$, we can obtain such a schedule from $\mathscr{S}^*$ by removing idle times between tasks and by moving every task which would be scheduled after that some other machines are available, to schedule it on the first available machine. By doing this, we do not delay any task, so every organization has at least the same gain as in $\mathscr{S}^*$, and this new schedule is thus still optimal for MaxMinGain. This schedule does not contain any idle time before that the last task starts to be executed, and is thus 2-approximate for $(P||C_{\max})$.

Let us now show that there is no algorithm which is optimal for MaxMinGain and which has an approximation smaller than 2 for MOSP. Naturally, this implies that no algorithm can be optimal for MaxMinGain and have an approximation ratio smaller than 2 for $(P||C_{\max})$.

Proposition 8. *Let $m \geq 4$ and $\epsilon > 0$. There is no algorithm which is optimal for* MaxMinGain *and $(2 - \frac{7}{m+3} - \epsilon)$-approximate for MOSP.*

Proof. Let us consider the following instance, with two organizations. Organization O_1 owns $m-1$ machines and m^2 tasks of length 1. Organization O_2 owns one machine, 2 tasks of length $m-1$ and one task of length 3. O_1's local makespan is $m+2$ and O_2's local makespan is $2m+1$.

In an optimal schedule $\mathscr{S}_{MMG}$ for MaxMinGain, the m^2 tasks of O_1 are scheduled first, followed by three tasks of O_2 on three different machines. Indeed, in $\mathscr{S}_{MMG}$, the makespan of O_1 is m and the makespan of O_2 is $2m-1$, which is also the global makespan. O_1's gain is $\frac{m+2}{m}$ and O_2's gain is $\frac{2m+1}{2m-1}$. Since O_2 has the minimum gain, in order to increase the minimum gain we should to decrease O_2's makespan. This is only possible if a task of O_1 is delayed, being completed at time at least $m+1$ instead of m. The gain of O_1 would then be at most $\frac{m+1}{m}$, which is larger than the minimal gain in $\mathscr{S}_{MMG}$. Schedule $\mathscr{S}_{MMG}$ is thus optimal for MaxMinGain. We can also note that $\mathscr{S}_{MMG}$ is one with the smallest makespan among the optimal schedules for MaxMinGain.

Let us now consider $\mathscr{S}_{MOSP}$, an optimal schedule for MOSP. In $\mathscr{S}_{MOSP}$, the two tasks of O_2 of length $m-1$ are scheduled at time 0, first and $(m-1)(m-2)$ tasks of O_1, so that the load of every machine is $m-1$. Then, m tasks of O_1 are scheduled between $m-1$ and m. Schedule $\mathscr{S}_{MOSP}$ ends by the last task of O_2 (of length 3) at time m, and the remaining $2(m-1)$ tasks of O_1 on the $m-1$ other machines. In $\mathscr{S}_{MOSP}$, the makespan of O_1 is thus $m+2$ and the makespan of O_2 is $m+3$. Note that $\mathscr{S}_{MOSP}$ fulfills the rationality constraint and is optimal for $(P||C_{\max})$.

Let r be the ratio between the makespan in $\mathscr{S}_{MMG}$, the best schedule (w.r.t the minimization of the makespan) among schedules optimal for MaxMinGain, and the makespan in $\mathscr{S}_{MOSP}$, optimal for MOSP:

$$r = \frac{2m-1}{m+3} = 2 - \frac{7}{m+3}.$$

Therefore, for this instance, there is no optimal schedule for MaxMinGain which has an approximation ratio better than $(2 - \frac{7}{m+3})$ for MOSP. □

When m tends towards the infinity the ratio tends towards 2: it is thus impossible to find an optimal algorithm for MaxMinGain less than 2-approximate for the makespan minimization. We have showed that MaxMinGain is strongly NP-hard, hard to approximate and that, in the general case, ensuring a fair schedule can lead to low global efficiency. We will now propose a polynomial time heuristic which, in practice, returns good solutions for both the minimum gain and the global makespan.

5.4 Heuristic

In this section, we propose a polynomial time heuristic for MaxMinGain. The idea is to schedule the tasks of the organizations by increasing local makespans, in order to favor the organization with the lowest makespan, and then the organization with the second lowest makespan and so forth. The algorithm always maintains the minimum gain at 1 or above, fulfilling the rationality constraint. As explained in the sequel, it delays as much as possible the tasks owned by organizations with high local makespan in order to create space for organization with low makespans. An execution of this algorithm can be seen in Figure 2, in which each color corresponds to an organization (there are thus 5 organizations).

We will assume that the organizations are labelled in non decreasing order of their makespans: $C^1_{loc} \leq \cdots \leq C^N_{loc}$. In this algorithm we will consider two subroutines. The first one is the list scheduling algorithm LPT (for Longest Processing Time), which schedules the tasks of an organization by non increasing lengths (as soon as a machine is available, the remaining task of largest length is scheduled on this machine). The second subroutine consists in delaying the tasks of an organization in a way that no task ends after the local makespan of its organization and no task begins before the makespan of any organization with a lower local makespan, unless it is scheduled on the machines of the organization owning it (in practice, the tasks of O_i are scheduled using LPT from time C^i_{loc}, on all the machines and in the reverse order of time, under the constraint that no task should start on a machine of O_k (with $k < i$) before C^k_{loc}).

We start from the local schedules, showed in the first schedule in Figure 2. In the second schedule of this figure, the tasks of all the organizations have been delayed as much as possible, according to the delay operation that we have just presented. By delaying tasks in this way, we make sure that the organizations always have a makespan equal to their local makespan. Then, for each organization O_i, from O_1 to O_N, we execute two steps.

- Firstly, we schedule the tasks of O_i with LPT, at the end of the tasks of organizations O_k, with $k < i$ which have already been scheduled. This step is shown on schedule 3 of Figure 2, for Organization O_1, which is the one with the lowest local makespan (and the dark blue tasks). By doing so, Organization O_i may decrease its makespan.
- If O_i decreases its makespan, as a second step, we run another delay operation for the tasks of O_{i+1}, considering the new makespan of Organization O_i. We can see this in step 4

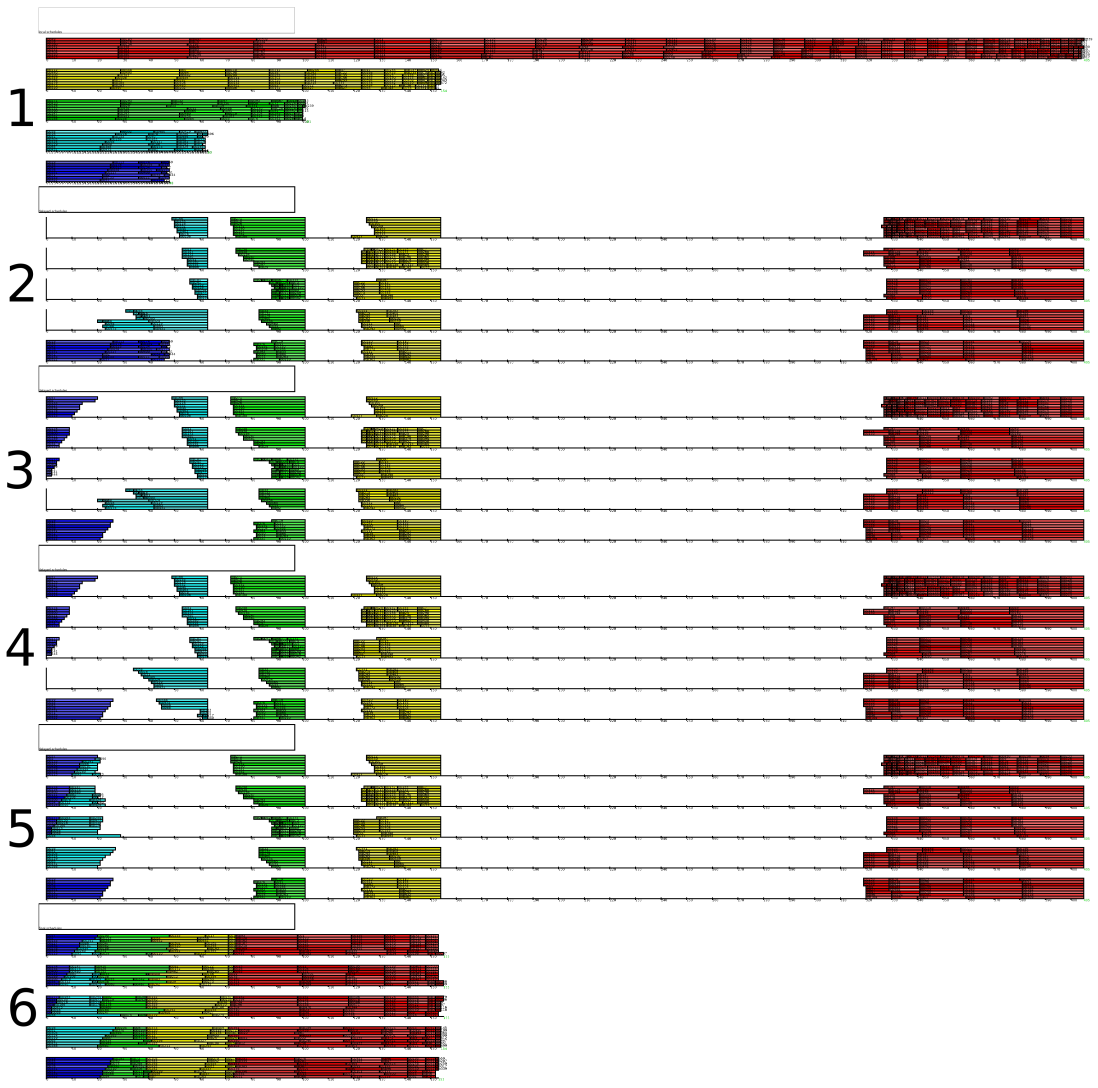


Figure 2: Example of 6 steps of the heuristic execution. Step 1: local schedules. Step 6: the scheduled returned by MCEDD.

of Figure 2: Organization O_2 (with the second smallest local makespan and light blue tasks) has been able to free space on its machine by delaying tasks that couldn't be moved earlier because the local makespan of O_1 was too high.

We repeat these steps for every organization, considering the organizations in non decreasing order of their makespans. We get our final schedule in schedule 6 of Figure 2.

Note that we decided to use LPT in order to schedule the tasks of an organization because of its low computational cost and its good approximation ratio, but it is possible to consider other scheduling algorithms. Note also that this algorithm is 2-approximate for $(P||C_{\max})$ (and thus for MOSP) since it returns a schedule with no idle time before the start of the last task. Since MaxMinGain is hard to approximate, we have no approximation ratio for the minimum gain. Let us now evaluate this algorithm experimentally.

5.5 Experimental evaluation

In this section, we study the efficiency of our algorithm on randomly generated instances. Let $\mathscr{S}$ be the schedule returned by our algorithm when executed on the instance I. We call :

$$s(I) = \frac{C_{\max}(\mathscr{S})}{\max(\overline{L(\mathscr{S})}, p_{max}(I))}$$

where $\overline{L(\mathscr{S})} = \frac{\sum_{i=1}^{m} L_i(\mathscr{S})}{m}$ is the average load of a machine in $\mathscr{S}$ and $p_{max}(I)$ denotes the largest length of a task in I. As we have seen before, $\max(\overline{L(\mathscr{S})}, p_{max}(I))$ is a lower bound of an optimal makespan for instance I. We also define :

$$s'(I) = \frac{\min\limits_{i \in \{1,\dots,N\}} \frac{C_i^{loc}}{C_i(\mathscr{S})}}{\min\limits_{i \in \{1,\dots,N\}} \frac{C_i^{loc}}{\max(\overline{L_i(I)}, p_{max}(O_i))}}$$

where $\overline{L_i(I)}$ is the average load of a machine if the only tasks in I were the one of O_i; $p_{max}(O_i)$ denotes the largest length of a task owned by O_i. We can note that $\max(\overline{L_i(I)}, p_{max}(O_i))$ is a lower bound of the best makespan O_i could get. Then, $\frac{C_i^{loc}}{\max(\overline{L_i(I)}, p_{max}(O_i))}$ is a higher bound of the gain O_i can get.

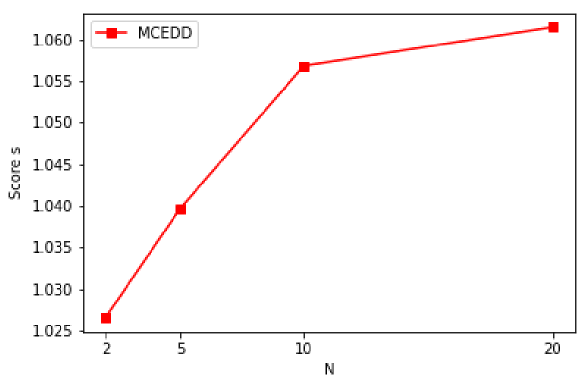
The local schedules are obtained with the LPT list scheduling. Instances are randomly generated thanks to a realistic generator [16]. We set the maximum task length to 50. Tasks are spread among the organizations following a zipf distribution; we set the number of elements of the distribution to N and s to 1.4267 which corresponds to the data observed in [14]. We create instances varying three parameters: the number of tasks n, the number of machines m and the number of organizations N. Machines are spread uniformly. We consider 9600 instances.
Since our algorithm tries to return a solution as fair as possible, we will focus on the impact of the number of organizations on the quality of the solution returned by our algorithm on the tested instances.

We see in Figure 3(a) that the score s increases with the number of organizations. This is consistent with the idea that the more organizations there are, the more difficult it is to satisfy each one of them. We also observe that the s score is below 1.07, which means the schedule returned by our algorithm has on average a makespan lower than 1.07 times a lower bound of optimal global makespan with no rationality constraint.

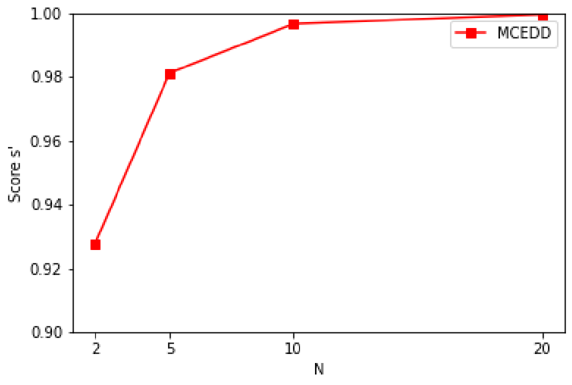
Figure 3(b) shows the variation of score s'. We note that s' is on average above 0.92. This means that the minimum gain in the schedule returned by our algorithm is higher than 0.92 times a higher bound of the optimal minimum gain. This figure also shows that when the number of organizations increases, s' gets closer to 1. This means that the approximation gets better when the number of organizations increases. The reason is that when we have a lot of organizations, it is likely that at least one organization cannot decrease its makespan or can decrease it very slightly: the higher bound is smaller and it is easier to get to this bound.

6 Conclusion and future work

In this paper, we have focused on two problems: MOSP (or $(1+\alpha)$-MOSP), for which we have studied the necessary tradeoff between efficiency (in term of low makespan) and the (relaxed)



(a) Variation of score s



(b) Variation of score s'

Figure 3: Experimental evaluation of MCEDD.

rationality constraint. We have also shown the interest of cooperation, that can benefit to all the organizations, and proposed an algorithm which returns schedules $(1+\epsilon)$-approximate for MOSP while the makespans of the organizations are increased by at most a factor $(1 + \epsilon)$. We then introduced problem MaxMinGain, for which we have also shown the necessary tradeoff between the minimization of the makespan and the minimization of the minimal gain (excepted if the tasks have all the same length, instances for which there is a polynomial time algorithm optimal for both objectives). We have shown that MaxMinGain is inapproximable in polynomial time if $P \neq NP$, but we have given a heuristic which, in practice, returns good schedules for both the minimization of the makespan and the maximization of the minimal gain.

Note that most results can be adapted if the tasks have released dates. Indeed, the "negative" results are still valid (this concerns complexity proofs, and results showing the necessary tradeoff between the global makespan and either the rationality constraint or the maximization of the minimal gain). The optimal algorithm for MaxMinGain with unit tasks can also be easily adapted. The PTAS of Hall and Shmoys [12] works with release dates, and thus we can use its adaptation with release dates too: there is also in this case a $(1 + \epsilon)$-approximate schedule for MOSP while the makespans of the organizations are increased by at most a factor $(1 + \epsilon)$. Likewise, this algorithm can be adapted when machines are not necessary identical but can have a fixed number of different speeds.

Note also that in this paper we have considered that each organization owns at least one machine, but results also hold if there are organizations with tasks but without any machine (in this case, they do not have "local makespan", and we do not apply the rationality constraint for these organizations).

A possible future work direction would be to consider online tasks (when we do not know their release dates in advance). In this case, allowing preemption of the tasks could certainly be useful. Another interesting direction is when the organizations do not have the same objectives. Some organizations could aim at minimizing their makespan, while some others could wish to minimize the average completion time of their tasks, or the average tardiness of their tasks, etc. There are numerous works in multi agent scheduling [1], where agents can have different objectives, but these work assume that machines are shared and do not belong to the agents, which only own tasks.

Last but not least, numerous things remain to be done considering fairness. Notions widely used in the fair allocation field [3] could certainly be useful. For example, envy-freeness is a standard notion in fair division problems, where a set of items should be shared among agents – the aim is that no agent is "jealous" from another agent. This notion could for example be introduced in the context where organizations share tasks and machines.

Acknowledgements. We thank Krzysztof Rzadca for giving us the source code that he wrote to draw nice schedules (this allowed us to get the drawings of Figure 2). We also wish to thank the anonymous reviewers for their helpful comments.

References

[1] Alessandro Agnetis, Jean-Charles Billau, Stanislaw Gawiejnowicz, Dario Pacciarelli, and Ameur Soukhal. *Multiagent Scheduling, Models and Algorithms.* Springer, 2014.

[2] Alessandro Agnetis, Bo Chen, Gaia Nicosia, and Andrea Pacifici. Price of fairness in two-agent single-machine scheduling problems. *European Journal of Operational Research*, 276(1):79–87, 2019.

[3] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel Procaccia, editors. *Fair Allocation*, page 259–260. Cambridge University Press, 2016.

[4] Anirudh Chakravorty, Neelima Gupta, Neha Lawaria, Pankaj Kumar, and Yogish Sabharwal. Algorithms for the relaxed multiple-organization multiple-machine scheduling problem. In *20th Annual International Conference on High Performance Computing, HiPC 2013, Bengaluru (Bangalore), Karnataka, India, December 18-21, 2013*, pages 30–38. IEEE Computer Society, 2013.

[5] Johanne Cohen, Daniel Cordeiro, and Pedro Luis F. Raphael. Energy-aware multi-organization scheduling problem. In Fernando M. A. Silva, Inês de Castro Dutra, and Vítor Santos Costa, editors, *Euro-Par 2014 Parallel Processing - 20th International Conference, Porto, Portugal, August 25-29, 2014. Proceedings*, volume 8632 of *Lecture Notes in Computer Science*, pages 186–197. Springer, 2014.

[6] Johanne Cohen, Daniel Cordeiro, Denis Trystram, and Frédéric Wagner. Analysis of multi-organization scheduling algorithms. In Pasqua D'Ambra, Mario Rosario Guarracino, and Domenico Talia, editors, *Euro-Par 2010 - Parallel Processing, 16th International Euro-Par Conference, Ischia, Italy, August 31 - September 3, 2010, Proceedings, Part II*, volume 6272 of *Lecture Notes in Computer Science*, pages 367–379. Springer, 2010.

[7] Johanne Cohen, Daniel Cordeiro, Denis Trystram, and Frédéric Wagner. Multi-organization scheduling approximation algorithms. *Concurr. Comput. Pract. Exp.*, 23(17):2220–2234, 2011.

[8] Johanne Cohen, Daniel Cordeiro, Denis Trystram, and Frédéric Wagner. Multi-organization scheduling approximation algorithms. *Concurrency and Computation: Practice and Experience*, 23(17):2220–2234, 2011.

[9] Daniel Cordeiro, Pierre-François Dutot, Grégory Mounié, and Denis Trystram. Tight analysis of relaxed multi-organization scheduling algorithms. In *25th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2011, Anchorage, Alaska, USA, 16-20 May, 2011 - Conference Proceedings*, pages 1177–1186. IEEE, 2011.

[10] Pierre-François Dutot, Fanny Pascual, Krzysztof Rzadca, and Denis Trystram. Approximation algorithms for the multiorganization scheduling problem. *IEEE Trans. Parallel Distrib. Syst.*, 22(11):1888–1895, 2011.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, 1979.

[12] Leslie A. Hall and David B. Shmoys. Approximation schemes for constrained scheduling problems. In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 134–139. IEEE Computer Society, 1989.

[13] Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *J. ACM*, 34(1):144–162, January 1987.

[14] Alexandru Iosup, Catalin Dumitrescu, Dick Epema, Hui Li, and Lex Wolters. How are Real Grids Used? The Analysis of Four Grid Traces and Its Implications. In *2006 7th IEEE/ACM International Conference on Grid Computing*, pages 262–269, September 2006. ISSN: 2152-1093.

[15] J.R. Jackson. *Scheduling a production line to minimize maximum tardiness*. Research report. Office of Technical Services, 1955.

[16] Uri Lublin and Dror G. Feitelson. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63(11):1105–1122, November 2003.

[17] Fukuhito Ooshita, Tomoko Izumi, and Taisuke Izumi. A generalized multi-organization scheduling on unrelated parallel machines. In *2009 International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT 2009, Higashi Hiroshima, Japan, 8-11 December 2009*, pages 26–33. IEEE Computer Society, 2009.

[18] Fukuhito Ooshita, Tomoko Izumi, and Taisuke Izumi. The price of multi-organization constraint in unrelated parallel machine scheduling. *Parallel Process. Lett.*, 22(2), 2012.

[19] Fanny Pascual, Krzysztof Rzadca, and Piotr Skowron. Collective schedules: Scheduling meets computational social choice. *CoRR*, abs/1803.07484, 2018.

[20] Fanny Pascual, Krzysztof Rzadca, and Denis Trystram. Cooperation in multi-organization scheduling. In Anne-Marie Kermarrec, Luc Bougé, and Thierry Priol, editors, *Euro-Par 2007, Parallel Processing, 13th International Euro-Par Conference*, volume 4641 of *Lecture Notes in Computer Science*, pages 224–233. Springer, 2007.

[21] Fanny Pascual, Krzysztof Rzadca, and Denis Trystram. Cooperation in multi-organization scheduling. *Concurr. Comput. Pract. Exp.*, 21(7):905–921, 2009.

[22] Piotr Skowron and Krzysztof Rzadca. Non-monetary fair scheduling: a cooperative game theory approach. In *SPAA*, pages 288–297, 2013.