Multi-Organizational Scheduling: Individual Rationality, Optimality, and Complexity

Jiehua Chen[1] , **Martin Durand**[1,2] , **Christian Hatschka**[1]
[1]TU Wien, Institute of Logic and Computation, Vienna, Austria
[2]Sorbonne Université, CNRS, LIP6, Paris, France
{jchen, chatschka}@ac.tuwien.ac.at, martin.durand@lip6.fr

Abstract

We investigate multi-organizational scheduling problems, building upon the framework introduced by Pascual et al. in 2009. In this setting, multiple organizations each own a set of identical machines and sequential jobs with distinct processing times. The challenge lies in optimally assigning jobs across organizations' machines to minimize the overall makespan while ensuring no organization's performance deteriorates. To formalize this fairness constraint, we introduce individual rationality, a game-theoretic concept that guarantees each organization benefits from participation.

Our analysis reveals that finding an individually rational schedule with minimum makespan is Θ_2^{P}-hard, placing it in a complexity class strictly harder than both NP and coNP. We further extend the model by considering an alternative objective: minimizing the sum of job completion times, both within individual organizations and across the entire system. The corresponding decision variant proves to be NP-complete. Through comprehensive parameterized complexity analysis of both problems, we provide new insights into these computationally challenging multi-organizational scheduling scenarios.

1 Introduction

Multi-organizational scheduling (MOS) has emerged as a crucial paradigm in distributed computing environments, where organizations collaborate by sharing their computational resources to optimize job processing [Pascual *et al.*, 2009]. In this model, multiple organizations, each possessing their own machines and jobs, connect their resources through a grid network to create more efficient scheduling solutions. Collaboration can potentially improve global performance metrics, such as the overall *makespan* (i.e., completion time of the last job) or the *total sum of completion times of all jobs*. A possible real-life application of such a model would be a set of universities or research units, each owning a cluster of machines used by their respective members to run computer programs. These organizations may be willing to mutualize their resources in order to balance the computational load of their clusters.

However, this collaborative framework introduces strategic considerations, as each organization prioritizes its own objectives (e.g., minimizing its local makespan). From a game-theoretical perspective, if any organization would achieve worse performance in the collaborative schedule compared to operating independently, it is unfair for that organization; so the organization has an incentive to withdraw from the cooperation. Such a withdrawal could cascade into disrupting other organizations' schedules. Therefore, a fundamental constraint in our scheduling problem is *individual rationality*, ensuring that no organization performs worse under cooperation than it would independently.

While individually rational schedules are guaranteed to exist (as organizations can always default to an optimal local schedule), the challenge lies in finding one that additionally optimizes global performance metrics. This work focuses on two fundamental metrics: the maximum completion time ($\mathsf{C}_{\max}$) and the sum of completion times (C_{Σ}). We examine scenarios where both individual organizations and the grand coalition as a whole optimize either $\mathsf{C}_{\max}$ or C_{Σ}, leading to two distinct optimization problems: $\mathsf{C}_{\max}$-MOS and C_{Σ}-MOS (formally defined in Section 2).

Main contributions. We introduce *individual rationality* to the multi-organizational scheduling framework. Under this fairness concept, no organization has an incentive to withdraw from collaboration since no local schedule can achieve a better performance. We systematically investigate the algorithmic complexity of two optimization problems: $\mathsf{C}_{\max}$-MOS and C_{Σ}-MOS. Generally speaking, both problems are computationally hard. More precisely, the decision variant of $\mathsf{C}_{\max}$-MOS is Θ_2^{P}-complete[1] while the one of C_{Σ}-MOS is NP-complete.

We also present parameterized complexity analysis considering key parameters (and their combinations). They are:
- k: the number of organizations,
- m: the number of machines,
- n: the number of jobs,

[1]Θ_2^{P} (aka. $\mathrm{P}^{\mathrm{NP}[\log]}$ and $\mathrm{P}_{||}^{\mathrm{NP}}$) is a complexity class, consisting of all problems which can be decided in polynomial time with *logarithmically* many queries to an NP-oracle [Wagner, 1990], positioning it between NP and Σ_2^{P} in the complexity hierarchy.

– τ: the target value of the objective (i.e., either makespan or sum of competition times) function in the decision variant,
– $\mathsf{p}_{\max}$: the maximum processing time of a job,
– $n_{\max}$: the maximum number of jobs owned by an organization, and
– $\mathsf{m}_{\max}$: the maximum number of machines owned by an organization.

Note that we chose parameters that are studied in the literature on scheduling [Mnich and Wiese, 2015; Mnich and Van Bevern, 2018], as well as parameters that are unique to the multi-organizational setting: $k, n_{\max}$, and $\mathsf{m}_{\max}$. The latter two are localized versions of the global parameters n and m, respectively.

Among the parameterized findings, for the parameter combination $k + \mathsf{p}_{\max}$, we develop a *fixed-parameter tractable* (FPT) algorithm for $\mathsf{C}_{\max}$-MOS, based on integer-linear programming (ILP). Our approach is based on an FPT-algorithm by Mnich and Wiese [2015] for the classical problem of minimizing the makespan; this is equivalent to our model with a single organization. They proved the existence of an optimal solution that evenly distributes all jobs of the same processing time among the machines; the difference is upper-bounded by a function in $\mathsf{p}_{\max}$ only. This implies that there are only a few number of different types of machines. We extend this idea and show that the number of different machine types is upper-bounded by $k + \mathsf{p}_{\max}$. We can then introduce an integer variable for each machine type and use ILP to find an optimal solution.

Via a straightforward dynamic programming (DP) approach, we also demonstrate that $\mathsf{C}_{\max}$-MOS (resp. C_{Σ}-MOS) is in XP with respect to m (resp. $\mathsf{m} + \mathsf{p}_{\max}$). For the hardness, we prove that $\mathsf{C}_{\max}$-MOS remains DP-hard even when k is a small constant[2].

Table 1 summarizes our complete complexity findings. Due to space constraints, proofs of results marked with a ($\star$) symbol are deferred to the full version of the paper [Chen *et al.*, 2025].

Related work. Pascual *et al.*[2009] initiated the study of cooperation in multi-organizational scheduling, where jobs may require parallel execution across machines. They addressed the problem of minimizing the global makespan under a *local constraint* that no organization performs worse compared to a specific local schedule, computed using a heuristic. However, this constraint differs from the individual rationality we focus on in this paper, as the heuristic-based local schedule may not be optimal, meaning organizations might still have an incentive to leave the cooperation. Pascual *et al.* [2009] showed that their problem is NP-hard and provided approximation algorithms.

Cohen *et al.* [2011] considered the same model and proposed approximation algorithms for sequential jobs. Durand and Pascual [2021] examined a more general setting where the local schedules are given as input and studied its approximability. Variants of these problems also allow organizations to pursue objectives beyond minimizing the makespan of their own jobs, such as minimizing the sum of job completion times [Cohen *et al.*, 2011] or the energy required to schedule jobs [Cohen *et al.*, 2014]. Other studies relaxed the individual rationality constraint, allowing organizations to accept schedules where their makespan increases, provided the increase is within a given factor [Ooshita *et al.*, 2009; Ooshita *et al.*, 2012; Chakravorty *et al.*, 2013; Cordeiro *et al.*, 2011]. Rzadca [2007] introduces the notion of self-reliance and Skowron and Rzadca [2014] employed cooperative game theory in multi-organizational scheduling, but using Shapley values as a measure of fairness. As mentioned earlier, our definition of individual rationality is stronger as it compares each organization's outcome to its optimal local schedule, in line with standard individual rationality definitions in coalition formation games.

Parameterized complexity has recently gained attention in scheduling [Mnich and Van Bevern, 2018]. In the classical setting (with a single organization), the problem of minimizing the makespan is shown to be FPT with respect to the maximum processing time of a task $\mathsf{p}_{\max}$ [Mnich and Wiese, 2015; Knop *et al.*, 2020]. Multi-organizational cooperation has also been studied in other contexts, such as matching [Biró *et al.*, 2019; Gourvès *et al.*, 2012] and kidney exchange [Sönmez and Ünver, 2013; Ashlagi and Roth, 2012; Ashlagi and Roth, 2014; Klimentova *et al.*, 2021].

2 Preliminaries

For details and definitions from parameterized complexity, we refer to the textbook by Cygan *et al.* [2015].

Given an integer $z \in \mathbb{Z}$, let $[z] = \{1, \ldots, z\}$. An instance of MOS is a tuple $\langle \mathcal{O}, (M_i)_{i\in[k]}, (J_i)_{i\in[k]}, (\mathsf{p}^i_j)_{i\in[k],j\in[n_i]} \rangle$, where $\mathcal{O}$ denotes a set of organizations with $|\mathcal{O}| = k$ such that for each $i \in [k]$,
– M_i denotes a non-empty set of $\mathsf{m}_i = |M_i|$ identical machines,
– J_i denotes of a set of $n_i = |J_i|$ non-preemptive (i.e., each job has to be completely processed before another job) and sequential jobs α^i_j, and
– for each $j \in [n_i]$, p^i_j denotes the processing time of the j^{th} job[3], called α^i_j in J_i,

all associated with organization O_i.

Throughout, we assume that I denotes an instance of MOS of the form $\langle \mathcal{O}, (M_i)_{i\in[k]}, (J_i)_{i\in[k]}, (\mathsf{p}^i_j)_{i\in[k],j\in[n_i]} \rangle$. Moreover, we denote by $\mathcal{M}$ the set of all machines, i.e., $\mathcal{M} = \bigcup_{i\in[k]} M_i$, by $\mathcal{J}$ the set of all jobs, i.e., $\mathcal{J} = \bigcup_{i\in[k]} J_i$, by n the total number of jobs, and finally by m the total number of machines.

Feasible schedules. A *schedule* $\sigma : \mathcal{J} \to \mathcal{M} \times \mathbb{N}$ is a function that assigns to each job a machine and a completion time. For notational convenience, for each organization O_i and each $j \in [n_i]$, we denote by $\mathsf{m}^i_j(\sigma)$ the scheduled machine and by $\mathsf{C}^i_j(\sigma)$ the scheduled completion time of the j^{th} job of organization i under σ.

A schedule is *feasible* if each job is assigned a machine with feasible completion time and no two jobs can occupy the same machine in the same processing time. Formally, we have that

[2]DP is the class of problems expressible as the difference of an NP- and a coNP problem [Papadimitriou, 1994, Chapter 17].

[3]In this paper we suppose that the instance is encoded in unary.

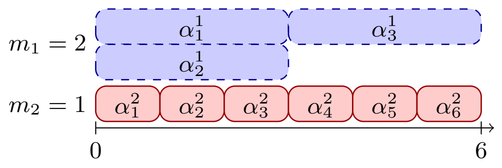

Figure 1: Possible local schedules for O_1 (top) and O_2 (bottom) from Example 1. Interpretation: Each job is represented by a rectangle, with the length depicting the processing time. Jobs on the same row are assigned to the same machine. Time goes from left to right, i.e., a job represented left to another job is processed earlier in the schedule. O_1's jobs are in blue, while O_2's jobs in red.

- for each job α_j^i, $\mathsf{C}_j^i(\sigma) \geq \mathsf{p}_j^i$ and
- for each two jobs $\alpha_j^i \neq \alpha_{j'}^{i'}$ scheduled on the same machine, $|\mathsf{C}_j^i(\sigma) - \mathsf{C}_{j'}^{i'}(\sigma)| \geq \min(\mathsf{p}_j^i, \mathsf{p}_{j'}^{i'})$, i.e., either the starting time of α_j^i is later or equal than the completion time of $\alpha_{j'}^{i'}$ or the completion time of α_j^i is earlier or equal to the starting time of $\alpha_{j'}^{i'}$.

A schedule σ is a *local* schedule for *organization $O_i \in \mathcal{O}$* if it is feasible and for every job $\alpha_j^i \in J_i$ it holds that $\mathsf{m}_j^i(\sigma) \in M_i$. We will oftentimes just use "schedules" to refer to "feasible schedules" when it is clear from the context.

Makespan and sum of completion times. Let σ be a schedule. Then, the *makespan* (resp. the *sum of completion times*, in short *Σ-time*) of a set of jobs $\mathcal{J}'$ with respect to σ, denoted as $\mathsf{C}_{\max}(\sigma, \mathcal{J}')$ (resp. $\mathsf{C}_\Sigma(\sigma, \mathcal{J}')$), is the maximum completion time (resp. sum of completion times) of all jobs in $\mathcal{J}'$. The *makespan* (resp. the *Σ-time*) of *organization $O_i \in \mathcal{O}$* with respect to σ is $\mathsf{C}_{\max}^i(\sigma) = \mathsf{C}_{\max}(\sigma, J_i)$ (resp. $\mathsf{C}_\Sigma^i(\sigma) = \mathsf{C}_\Sigma(\sigma, J_i)$). We omit the second argument $\mathcal{J}'$ when we refer to the makespan (resp. Σ-time) of all jobs, i.e., $\mathsf{C}_{\max}\,(\sigma) = \mathsf{C}_{\max}(\sigma, \mathcal{J})$ and $\mathsf{C}_\Sigma\,(\sigma) = \mathsf{C}_\Sigma(\sigma, \mathcal{J})$, respectively.

The *optimal local-makespan* (resp. *optimal local-Σ-time*) of *organization O_i*, denoted as $\mathsf{OPT}\text{-}\mathsf{C}_{\max}^i$ (resp. $\mathsf{OPT}\text{-}\mathsf{C}_\Sigma^i$) is the minimum over the makespans (resp. minimum over the Σ-times) of all *local* schedules of O_i. In other words, the optimal local-makespan (resp. optimal local-Σ-time) of an organization O_i is the minimum makespan (resp. minimum Σ-time) achievable by any schedule where all jobs of O_i are only scheduled to the machines of O_i.

A schedule σ is an *optimal local schedule* for organization O_i if it is a local schedule for O_i and has makespan equal to $\mathsf{OPT}\text{-}\mathsf{C}_{\max}^i$ (resp. Σ-time equal to $\mathsf{OPT}\text{-}\mathsf{C}_\Sigma^i$).

Example 1. *We consider an example with two organizations: O_1 and O_2. Organization O_1 owns $\mathsf{m}_1 = 2$ machines and $n_1 = 3$ jobs with processing times $\mathsf{p}_1^1 = \mathsf{p}_2^1 = \mathsf{p}_3^1 = 3$. Organization O_2 owns $\mathsf{m}_2 = 1$ machine and $n_2 = 6$ jobs with processing times $\mathsf{p}_1^2 = \mathsf{p}_2^2 = \cdots = \mathsf{p}_6^2 = 1$. Possible local schedules for the instance are drawn in Figure 1.*

Individual rationality. A schedule σ is called *individually rational* if no organization is worse-off by looking at the optimal local-makespan or local-sum of completion times. More precisely, for the objective of minimizing the makespan ($\mathsf{C}_{\max}$), we require that $\mathsf{C}_{\max}^i(\sigma) \leq \mathsf{OPT}\text{-}\mathsf{C}_{\max}^i$ holds for each $O_i \in \mathcal{O}$, while for the objective of minimizing the sum of competition times (C_Σ), we require that $\mathsf{C}_\Sigma^i(\sigma) \leq \mathsf{OPT}\text{-}\mathsf{C}_\Sigma^i$ holds for each $O_i \in \mathcal{O}$.

Clearly, for both objectives, individually rational schedules exist as one can compute an optimal schedule for each organization and combine them into a global one.

Example 2. *The local schedules displayed in Figure 1 are optimal local schedules for both organizations and both objectives. The optimal local-makespans of both organizations are $\mathsf{OPT}\text{-}\mathsf{C}_{\max}^1 = \mathsf{OPT}\text{-}\mathsf{C}_{\max}^2 = 6$. By definition, this is an individually rational schedule. We consider a schedule σ in which two jobs of O_2 are first on each machine followed by one of the three jobs of O_1, then σ is individually rational and minimizes the overall makespan: The processing times of all jobs sum up to 15 and we have 3 machines. So the minimum makespan is at least 5. Moreover, the makespan of O_1 is 5 while the makespan of O_2 is 2.*

One can check that σ is also optimal when we aim at minimizing Σ-time instead, with a total of $24 = 3 + 6 + 15$. However it is not individually rational if Σ-time is the objective. Indeed, the optimal local-Σ-time of the organizations are $\mathsf{OPT}\text{-}\mathsf{C}_\Sigma^1 = 3 + 3 + 6 = 12$ and $\mathsf{OPT}\text{-}\mathsf{C}_\Sigma^2 = 1 + 2 + 3 + 4 + 5 + 6 = 21$ and the sum of completion times of O_1 in σ is 15.

Central problems. We look at two optimization problems, which aim for an optimal solution among all individually rational schedules. In the following, let $\Omega \in \{\mathsf{C}_{\max}, \mathsf{C}_\Sigma\}$.

Ω-MOS
Input: An instance I of MOS.
Task: Find a schedule σ among all *individually rational* schedules for I such that $\Omega(\sigma)$ is minimum.

The decision variants, called Ω-MOS-DEC have additionally a non-negative integer τ as input and ask whether there exists an *individually rational* schedule σ with $\Omega(\sigma) \leq \tau$.

Remarks. Note that in the classical setting (i.e., when the number of organizations is one), it is NP-hard to find a schedule with minimum makespan whereas for the minimum Σ-time case it is polynomial-time solvable [Brucker, 2004]. Hence, $\mathsf{C}_{\max}$-MOS-DEC is contained in Σ_2^{P} while C_Σ-MOS-DEC in NP. We will show that $\mathsf{C}_{\max}$-MOS-DEC is between NP and Σ_2^{P}; it is Θ_2^{P}-complete (Theorem 1), while C_Σ-MOS-DEC is NP-complete (Theorem 3).

3 Minimizing the Makespan

3.1 General Complexity

We start this section by showing that $\mathsf{C}_{\max}$-MOS-DEC is Θ_2^{P}-complete.

Theorem 1 ($\star$). $\mathsf{C}_{\max}$-MOS-DEC *is* Θ_2^{P}-complete.

Proof Sketch. We start with the containment proof. To this end, we introduce an intermediate scheduling NP problem for MOS and show how to answer $\mathsf{C}_{\max}$-MOS-DEC by making only logarithmically many calls to the NP-oracle of the newly introduced problem.

	$\mathsf{C}_{\mathsf{max}}$-MOS	C_{Σ}-MOS
Dec. Variant	Θ_2^{P}-c [T1]	NP-c [T3]
k	DP-h/? [P1]	W[1]-h/? [P5]
m	W-h$^\spadesuit$/XP [P4]	? ?
n	FPT [P3]	FPT [P6]
τ	FPT [C1]	FPT [C3]
$\mathsf{p}_{\mathsf{max}}$	? ?	? ?
$n_{\mathsf{max}} + \mathsf{m}_{\mathsf{max}}$	NP-c† [P2]	NP-c [T3]
$\mathsf{p}_{\mathsf{max}} + k$	FPT [T2]	? ?
$\mathsf{p}_{\mathsf{max}} + \mathsf{m}$	FPT [T2]	?/XP [P7]
$\mathsf{p}_{\mathsf{max}} + n_{\mathsf{max}}$	FPT [C2]	? ?

Table 1: See the introduction for the definition of the parameters. "DP-h" means the problem remains DP-hard even if the value of the corresponding parameter is a constant. "W-h$^\spadesuit$" means the problem is W[1]-hard and it is due to Jansen *et al.* [2013]. "NP-c" for the parameter combination $n_{\mathsf{max}} + \mathsf{m}_{\mathsf{max}}$ means that the decision variant is contained in NP when either of the parameters is a constant and it remains NP-hard even if both parameters have values bounded by a constant, the $\mathsf{C}_{\mathsf{max}}$-MOS proof follows directly from [Cohen *et al.*, 2011]. Note that all other two parameter combinations either have one parameter subsumed by the other, or have the result follow directly from another.

$\mathsf{C}_{\mathsf{max}}$-MOS-LOCALSCHEDULES ($\mathsf{C}_{\mathsf{max}}$-MOS-LS)
Input: An instance I of MOS, two integers τ_{g} and T'.
Question: Are there two schedules σ and σ_{lo} of all jobs such that:
(1) For all $(i, j) \in [k] \times [n_i]$: $\mathsf{m}^i_j(\sigma_{\mathsf{lo}}) \in M_i$,
(2) $\sum_{i\in[k]} \left(\max_{j\in[n_i]} \mathsf{C}^i_j(\sigma_{\mathsf{lo}})\right) \leq T'$,
(3) $\left(\max_{\alpha^i_j \in \mathcal{J}} \mathsf{C}^i_j(\sigma)\right) \leq \tau_{\mathsf{g}}$, and
(4) for all $(i, j) \in [k] \times [n_i]$: $\mathsf{C}^i_j(\sigma) \leq \max_{j'\in[n_i]} \{\mathsf{C}^i_{j'}(\sigma_{\mathsf{lo}})\}$?

Clearly, $\mathsf{C}_{\mathsf{max}}$-MOS-LS is contained in NP as we can check in polynomial time whether two given schedules σ and σ_{lo} fulfill the conditions. Intuitively, this problem asks whether there is a local schedule with sum of makespans of the organizations equal to T' and a global schedule with makespan at most τ_{g} such that no organization has a larger makespan in the global schedule than in the local schedule. We now describe an algorithm answering the $\mathsf{C}_{\mathsf{max}}$-MOS problem using only a logarithmic number of calls to an oracle solving $\mathsf{C}_{\mathsf{max}}$-MOS-LS.

Let I be an instance of $\mathsf{C}_{\mathsf{max}}$-MOS and τ the target makespan. First, we perform a binary search on $\mathsf{C}_{\mathsf{max}}$-MOS-LS to find the minimum sum T' of makespans among all local schedules of I. We can do this because if σ_{lo} is a local schedule with minimum sum T' of makespans, then (I, T', T') is a yes-instance of $\mathsf{C}_{\mathsf{max}}$-MOS-LS such that $(\sigma, \sigma_{\mathsf{lo}})$ with $\sigma = \sigma_{\mathsf{lo}}$ is a witness. Formally, we start with $T' = \mathsf{p}_{\mathsf{max}} \cdot n$. For every NP-oracle call, we set $\tau_{\mathsf{g}} = T'$ and then do a binary search to find the minimum value T' towards which the instance is still a yes-instance of $\mathsf{C}_{\mathsf{max}}$-MOS-LS. Note that a local schedule with minimum sum T' of makespans among all local schedules is also an optimal local schedule for each organization. This is because if one organization would have a smaller local-makespan, then by exchanging the corresponding schedule one would get a smaller sum of makespans of all organizations.

Once the minimum sum is found, we make one last call of the NP-oracle, where we set T' to be the found minimum and $\tau_{\mathsf{g}} = \tau$; recall that τ is the target makespan. We answer yes if and only if the last call gives a yes-answer. The correctness follows by checking the definition. This completes the containment proof. Regarding hardness, we only give a brief sketch and defer the detailed proof to the full paper. We reduce from a Θ_2^{P}-complete problem consisting of comparing two ordered sets of 3-PARTITION instances, where we assume that in each ordered set, all yes-instances appear before all no-instances. An instance of the Θ_2^{P}-complete problem is a yes-instance if and only if there are more 3-PARTITION yes-instances in the first set than in the second. We group instances by pairs, one from the first set, $\mathcal{I}$, and one from the second, $\mathcal{I}'$, and create a set of organizations for each pair. The local schedules of these organizations are such that if $\mathcal{I}'$ is a yes-instance, then $\mathcal{I}$ must also be a yes-instance to meet both individual rationality and the makespan requirement of τ. □

By reducing from a DP-hard problem, we can show that the problem is beyond NP and coNP, even in the case where there are only two organizations. We conjecture that the problem remains Θ_2^{P}-complete in this case.

Proposition 1 ($\star$). $\mathsf{C}_{\mathsf{max}}$-MOS *is* DP-hard *even if* $k = 2$.

The next result shows that the problem remains NP-hard even for the case when finding an optimal local schedule for each organization is easy. The hardness persists even if each organization has only two jobs. The hardness proof follows from [Cohen *et al.*, 2011].

Proposition 2 ($\star$). *For constant* n_{max} *or constant* $\mathsf{m}_{\mathsf{max}}$, $\mathsf{C}_{\mathsf{max}}$-MOS-DEC *is* NP-*complete. It remains* NP-*hard even if* $n_{\mathsf{max}} = 2$ *and* $\mathsf{m}_{\mathsf{max}} = 1$.

3.2 Algorithmic Results

We start with a fairly straightforward FPT result for the number n of jobs.

Proposition 3 ($\star$). $\mathsf{C}_{\mathsf{max}}$-MOS *is* FPT *with respect to* n.

Now, we turn to our main result: Theorem 2. As mentioned in the introduction, we extend the idea of the FPT algorithm by Mnich and Wiese [2015]. The idea is to group machines that for each processing time have the same number of jobs of that time together since jobs of the same processing time are interchangeable. They observed that there is always a *balanced* optimal schedule. Here, *balanced* means that all jobs of the same processing time can be evenly assigned among the machines so the difference is upper-bounded by a function in $\mathsf{p}_{\mathsf{max}}$. Due to this, the number of different groups is bounded by a function in $\mathsf{p}_{\mathsf{max}}$. Finally, one can design an ILP formulation that has an integer variable for each group specifying how many machines of that group exist in a balanced optimal schedule.

For the MOS setting, jobs belong to different organizations and may not be interchangeable, even if they have the same processing time. We circumvent this by also considering the parameter "the number k of organizations". By grouping the jobs according to the optimal local-makespan of their organization and showing that for each group and each processing time, each machine has the same number of jobs of that processing time up to a difference of a function of $\mathsf{p}_{\max}$, we are able to design an ILP similarly to Mnich and Wiese.

Before we show Theorem 2, we need two auxiliary lemmas and an observation and some additional definitions. In the $\mathsf{C}_{\max}$-MOS, each organization cares only about when its last job is finished. This time cannot exceed their optimal local-makespan. We order the jobs based on the optimal local-makespan of their organization. We say two jobs α^i_j and $\alpha^{i'}_{j'}$ belong to the same *phase* if their organizations have the same optimal local-makespan, i.e., $\mathsf{OPT}\text{-}\mathsf{C}^i_{\max} = \mathsf{OPT}\text{-}\mathsf{C}^{i'}_{\max}$. The jobs that belong to organizations with the smallest optimal local-makespan belong to phase 1. Formally, phase 1 consists of the jobs $\{\alpha^i_j \mid \nexists i' \mathsf{OPT}\text{-}\mathsf{C}^{i'}_{\max} < \mathsf{OPT}\text{-}\mathsf{C}^i_{\max}\} = \bigcup_{\arg\min \mathsf{OPT}\text{-}\mathsf{C}^i_{\max}} J_i$. Similarly, the jobs that have the next smallest optimal local-makespan will be referred to as jobs in phase 2 and so on. As all the jobs belonging to a single organization belong to the same phase it follows that the number of phases is upper-bounded by k. Let $\mathsf{phase}(\alpha^i_j)$ be the phase that job α^i_j belongs to. We define the *end of phase b for machine z and schedule σ* to be $\mathsf{end}_\sigma(b,z) = \max\{0\} \cup \{\mathsf{C}^i_j(\sigma) \mid \mathsf{phase}(\alpha^i_j) \le b \wedge \mathsf{m}^i_j(\sigma) = z\}$. Note that 0 is added to the set, as it would be possible for the set to be empty otherwise.

We start with a simple observation that jobs in an individually rational schedule can be *well ordered*.

Observation 1. *For each individually rational schedule σ, there exists another individually rational schedule σ' with makespan at most $\mathsf{C}_{\max}(\sigma)$ such that for each two jobs α and β that are assigned to the same machine, if α is in a phase earlier than β, then α is scheduled earlier than β as well.*

Proof. Such a schedule σ' can be found by iteratively switching consecutive jobs if they violate the well-ordered property. Each such exchange maintains individual rationality, as a job from a later phase belongs to an organization with larger optimal local-makespan and no job except the two which were exchanged in the ordering has a different completion time after this exchange. Repeating this process exhaustively yields the desired schedule σ'. □

By Observation 1, we assume from now on that every individually rational schedule satisfies the well-ordered property. We utilize this to upper-bound the difference between completion times of each phase between two machines.

Lemma 1 ($\star$). *For each individually rational schedule σ, there exists an individually rational schedule σ' with $\mathsf{C}_{\max}(\sigma') \le \mathsf{C}_{\max}(\sigma)$ such that for each pair of machines z_1 and z_2 and for each phase b it holds that $|\mathsf{end}_{\sigma'}(b,z_1) - \mathsf{end}_{\sigma'}(b,z_2)| \le \mathsf{p}^3_{\max} + \mathsf{p}_{\max}$.*

The next lemma upper-bounds the number of machines of the same type and phase. Specifically, we upper- and lower-bound the number of jobs of each processing time and phase that can be assigned to a machine.

Lemma 2 ($\star$). *Given an instance I of $\mathsf{C}_{\max}$-MOS let $J_{t,b}$ be the set of jobs of processing time t in phase b. Then I admits an optimal individually rational solution in which for every phase b and every distinct processing time t it holds that the number of jobs in $J_{t,b}$ scheduled on each machine is in the range $[\lfloor \frac{|J_{t,b}|}{m} \rfloor - O(\mathsf{p}_{\max}^{\mathsf{p}_{\max}}), \lfloor \frac{|J_{t,b}|}{m} \rfloor + O(\mathsf{p}_{\max}^{\mathsf{p}_{\max}})]$.*

The observation and lemmas allow us to search for an optimal solution with a very specific structure. This allows us to formulate an ILP with FPT running time and leads to the following theorem:

Theorem 2. *$\mathsf{C}_{\max}$-MOS is* FPT *with respect to $\mathsf{p}_{\max} + k$ and therefore $\mathsf{p}_{\max} + \mathsf{m}$.*

Proof. First note that $k \le \mathsf{m}$, as each organization has at least one machine. Therefore, it suffices to show the result for $\mathsf{p}_{\max} + k$. This approach is based on the FPT algorithm with respect to $\mathsf{p}_{\max}$ that solves $(P||\mathsf{C}_{\max})$ described by Mnich and Wiese [2015]. Intuitively, the proof works by running an ILP that fixes the schedule for each phase and linking the phases together afterwards.

We start by computing the number of phases by computing the optimal local makespan for each organization. By Mnich and Wiese's result this is doable in FPT time for each of the organizations. So computing it for all organizations is also doable in FPT time. We now ignore the organizations and group jobs by phases as previously described. Let $[B]$ be the set of phases and t_b the latest time by which jobs of phase b must be finished. Let P_b be the jobs in phase $b \in [B]$ and P^ℓ_b the jobs with processing time ℓ in phase b. For each phase $b \in [B]$ we compute $y_b := \frac{|P_b|}{m}$, this is the average makespan among the machines for jobs of only phase b. Note that these precomputation steps are also doable in polynomial time once the optimal local solutions for the organizations have been computed. Due to Lemma 1, we know that for every machine z, we can require that a machine must satisfy that $\mathsf{end}_\sigma(b,z) \in [\sum_{\ell=1}^{b} y_\ell - (\mathsf{p}^3_{\max} + \mathsf{p}_{\max}), \min \sum_{\ell=1}^{b} y_\ell + (\mathsf{p}^3_{\max} + \mathsf{p}_{\max}), t_b]$, for an optimal schedule S. Note that the left-hand side of the interval does not need a minimum, as assigning the optimal local schedule for each machine is individually rational.

Similarly we can see that the first job of the phase b must be scheduled in $[\sum_{\ell=1}^{b-1} y_\ell - (\mathsf{p}^3_{\max} + \mathsf{p}_{\max}), \min \sum_{\ell=1}^{b-1} y_\ell + (\mathsf{p}^3_{\max} + \mathsf{p}_{\max}), t_{b-1}]$, as we can assume that the jobs are ordered according to their phase due to Observation 1.

We can now describe the constraints and variables of the integer linear program (ILP) that solves this problem instance. Note that we do not distinguish between jobs that belong to the same phase and type, in the following. We start by describing the variables:

– For each phase $b \in [B]$, each possible starting point $\mathsf{start} \in [\sum_{\ell=1}^{b-1} y_\ell - (\mathsf{p}^3_{\max} + \mathsf{p}_{\max}), \min\{\sum_{\ell=1}^{b-1} y_\ell + (\mathsf{p}^3_{\max} + \mathsf{p}_{\max}), t_{b-1}\}]$ (if $b = 1$ we fix $start = 0$), each possible ending point $\mathsf{end} \in [\sum_{\ell=1}^{b} y_\ell - (\mathsf{p}^3_{\max} + \mathsf{p}_{\max}), \min\{\sum_{\ell=1}^{b} y_\ell + (\mathsf{p}^3_{\max} + \mathsf{p}_{\max}), t_b\}]$, and each vector M of length $\mathsf{p}_{\max}$ that satisfies that M_t is at least $\lfloor \frac{|P^p_b|}{m} \rfloor -$

$f(\mathsf{p}_{\max})$ and at most $\lfloor \frac{|P_b^p|}{m} \rfloor + f(\mathsf{p}_{\max})$ and $\sum_{t=1}^{\mathsf{p}_{\max}} M_t \cdot t = \mathsf{end} - \mathsf{start}$, we create a variable $v_{b,\mathsf{start},\mathsf{end},M}$. Intuitively, the vector M keeps track of how many jobs of processing time t are scheduled on a machine through the entry M_t. Note that all parameters must be non-negative integers (including zero) and that M may be the zero vector. These variables must all take integer values in the range $[0, \mathsf{m}]$. Informally, the value this variable takes is the number of machines that finished the previous phase(s) at time start, has exactly the number of jobs of each processing time as in M scheduled on them in phase b, and finishes phase b exactly at time end.

– We can also introduce an auxiliary variable e in order to solve the optimization problem. This is not necessary.

These variables turn out to be the only variables that are needed. In this proof, parameters will be called valid if they can form a variable as described above. We now describe the constraints that are needed.

(1) We need a constraint that limits the number of machines that can be used in each phase. As the total number of machines is m, this simply means that the variables need to sum up to m for each fixed $b \in [B]$.

$$\forall\, b \in [B]\colon \sum_{\forall \text{ valid start, end, } M} v_{b,\mathsf{start},\mathsf{end},M} = \mathsf{m}$$

(2) We need a constraint that makes sure that the starting times and end times of machines match between phases, such that each machine is ensured to only run one job at a time and not have any time when it is not processing any job. For the following constraint, let $C = \mathsf{p}_{\max}^3 + \mathsf{p}_{\max}$.

$$\forall\, b \in [B] \setminus \{1\}, \forall\, \mathsf{time} \in \Big[\sum_{\ell=1}^{b-1} y_\ell - C, \min\{\sum_{\ell=1}^{b} y_\ell + C, t_b\}\Big]\colon \sum_{\forall \text{ valid start, } M^1} v_{b-1,\mathsf{start},\mathsf{time},M^1} = \sum_{\forall \text{ valid end, } M^2} v_{b,\mathsf{time},\mathsf{end},M^2}$$

(3) We need a constraint that ensures that in each phase all the jobs that are part of this phase are scheduled. For each processing time t we add:

$$\forall\, b \in [B]\colon \sum_{\forall \text{ valid start,end,} M} M_t \cdot v_{b,\mathsf{start},\mathsf{end},M} = |P_i^t|$$

(4) In order to find an optimal solution we need to link the auxiliary variable e to the other variables.

$$\forall\, v_{b,\mathsf{start},\mathsf{end},M} : \min\{v_{b,\mathsf{start},\mathsf{end},M} \cdot \mathsf{end}, \mathsf{end}\} \leq e$$

In order to solve the optimization problem for the makespan we can then minimize e in the ILP.

We now show correctness of the ILP, by arguing that each valid schedule σ that has the form as described in Observation 1, Lemma 1, and Lemma 2 is a valid solution for the ILP (ignoring the minimization over e) and showing that every solution to the ILP can be transformed to a valid schedule σ.

Let σ be a valid schedule, for each phase b, start, end, and M we set $v_{b,\mathsf{start},\mathsf{end},M}$ to be equal to the number of machines that schedule jobs according to the vector M in that phase, such that $\mathsf{end}_\sigma(b-1, z) = \mathsf{start}$. This satisfies constraint 1, as each phase obviously only uses m machines. Constraint 2 is satisfied, as we set $\mathsf{end}_\sigma(b-1, z) = \mathsf{start}$, and constraint 3 is satisfied as we have a valid schedule.

For the other direction, we assign jobs phase by phase. For the first phase, we $v_{1,0,\mathsf{end},M}$ many machines with exactly the job seen in M. Then in step b we choose $v_{b,\mathsf{start},\mathsf{end},M}$ many machines that satisfy that $\mathsf{end}_\sigma(b-1, z) = start$ and assign the jobs in M to them. This is necessarily possible, due to constraint 2. Note that the number of machines in this step is exactly m due to constraint 1 and all jobs in this phase are scheduled due to constraint 3.

Finally, as e only tracks the largest end among variables $v_{b,\mathsf{start},\mathsf{end},M} \neq 0$, it returns the makespan.

As the number of variables as well as the number of constraints is FPT with respect to $\mathsf{p}_{\max} + k$, it follows that the ILP solves the problem in FPT time with respect to $\mathsf{p}_{\max} + k$. This concludes the proof. □

The next two corollaries follow directly from the proof of Theorem 2, as the number of phases and $\mathsf{p}_{\max}$ can be upper-bounded by the given parameters.

Corollary 1 ($\star$). $\mathsf{C}_{\max}$-MOS *is* FPT *with respect to* τ.

Corollary 2 ($\star$). $\mathsf{C}_{\max}$-MOS *is* FPT *with respect to* $n_{\max} + \mathsf{p}_{\max}$.

Finally, we use a dynamic programming approach that keeps track of the makespans of each machines for the assigned jobs in order to show the following result:

Proposition 4. $\mathsf{C}_{\max}$-MOS *is* XP *with respect to* m.

Proof. We first show how to compute an optimal local schedule for an arbitrary organization in XP-time with respect to m; we call this problem MINC$_{\max}$ via DP. Then, we show how to modify it to solve our problem $\mathsf{C}_{\max}$-MOS. For MINC$_{\max}$, we describe the DP for a given organization O_i with jobs J_i and M_i machines. We note that the ordering of jobs on a machine does not matter, but rather their processing times. We go through the jobs of O_i in this order $\alpha_1^i, \ldots, \alpha_{|J_i|}^i$.

We maintain a dynamic table $\mathcal{D}_{\mathsf{lo}}(z_1, \ldots, z_{\mathsf{m}_i}, j) \in \{0, 1\}$, where $z_1, \ldots, z_{\mathsf{m}_i} \in [\sum_{\ell=1}^{|J_i|} \mathsf{p}_\ell^i]$ and $j \in [|J_i|] \cup \{0\}$. Intuitively, the table entry is 1 if it is possible to assign the first j jobs to the machines such that machine d, $d \in [\mathsf{m}_i]$ has makespan z_d. We initialize the table with $\mathcal{D}_{\mathsf{lo}}(0, \ldots, 0, 0) = 1$. We now describe the recurrence:

$$\mathcal{D}_{\mathsf{lo}}(z_1, \ldots, z_{\mathsf{m}_i}, j) = \begin{cases} 1, & \textbf{if } \exists d \in [\mathsf{m}_i]\colon \mathcal{D}_{\mathsf{lo}}(z_1, \ldots, \\ & \quad z_d - \mathsf{p}_j^i, \ldots, z_{\mathsf{m}_i}, j-1) = 1 \\ 0, & \textbf{else} \end{cases}$$

The correctness of the recurrence is straightforward since it branches over the options of where to assign the jth job. Since the table has $(n \cdot \mathsf{p}_{\max})^{m+1}$ entries, by finding the table entry $\mathcal{D}_{\mathsf{lo}}(z_1, \ldots, z_{m'}, |J_i|) = 1$ that minimizes $\max\{z_1, \ldots, z_{m'}\}$ an optimal schedule can be found. Therefore, we can solve MINC$_{\max}$ and compute OPT-$\mathsf{C}_{\max}^i$ for each organization

O_i in XP-time with respect to m. Note that we require $\mathsf{OPT\text{-}C}^i_{\max}$ for the individual rationality constraint.

Now, we turn to our problem. Similarly to the proof of Theorem 2 we divide the jobs according to phases. As a reminder, $\mathsf{phase}(\alpha^i_j)$ refers to the phase of α^i_j. We order the jobs in a way $\alpha_1, \dots, \alpha_n$, such that the jobs satisfy $\mathsf{phase}(\alpha_1) \leq \dots \leq \mathsf{phase}(\alpha_n)$. Note that jobs from the same phase can be ordered in an arbitrary manner. We go through the jobs in this order.

We use $t(b)$ to refer to the time by which jobs in phase b need to be done, i.e., the optimal local-makespan of the organizations whose jobs belong to this phase.

We maintain a dynamic table $\mathcal{D}(z_1, \dots, z_{\mathsf{m}}, j) \in \{0, 1\}$, where $z_1, \dots, z_{\mathsf{m}} \in [\sum_{\ell=1}^{\sum_{i\in[k]} n_i} \mathsf{p}^i_\ell]$ and $j \in [\sum_{i\in[k]} n_i] \cup \{0\}$. Intuitively, the table entry is 1 if it is possible to assign the jobs $\alpha_1, \dots, \alpha_j$ to the machines such that machine d, $d \in [\mathsf{m}]$ has a makespan of z_d and individual rationality is upheld.

We initialize the table with $\mathcal{D}(0, \dots, 0, 0) = 1$. We now describe the recursive step:

$$\mathcal{D}(z_1, \dots, z_{\mathsf{m}}, j) =$$
$$\begin{cases} 1, & \textbf{if } \exists d \in [\mathsf{m}] \colon \mathcal{D}(z_1, \dots, \\ & \quad z_d - \mathsf{p}_j, \dots, z_{\mathsf{m}}, j-1) = 1 \\ & \textbf{and } \max\{z_1, \dots, z_{\mathsf{m}}\} \leq t(\mathsf{phase}(\alpha_j)) \\ 0 & \textbf{, else} \end{cases}$$

The correctness of the recurrence is straightforward since it branches over the options of where to assign the j^{th} job. We can do this, as we can assume that the optimal schedule is well-ordered due to Observation 1 and the ordering of jobs in the same phase on the same machine does not matter for a well-ordered schedule. Individual rationality is guaranteed, as the job must be done before the deadline due to $\max\{z_1, \dots, z_{\mathsf{m}}\} \leq t(\mathsf{phase}(\alpha_j))$. Since the table has $(n \cdot \mathsf{p}_{\max})^{m+1}$ entries, by finding the table entry $\mathcal{D}(z_1, \dots, z_m, n) = 1$ that minimizes $\max\{z_1, \dots, z_m\}$ an optimal schedule can be found. Therefore $\mathsf{C}_{\max}$-MOS is in XP with respect to m. □

4 Minimizing the Sum of Completion Times

While $\mathsf{C}_{\max}$-MOS inherits hardness from the matching scheduling problem, minimizing the makespan, it is not clear that C_Σ-MOS is NP-hard. Indeed, in the traditional scheduling setting, a schedule with minimum sum of completion times can be found in polynomial time [Brucker, 2004]. We show that C_Σ-MOS-DEC is NP-complete, even for a constant maximum number of jobs (resp. machines) per organization.

Theorem 3 ($\star$). C_Σ-MOS-DEC *is* NP-complete. *It remains* NP-*hard even if* $n_{\max} = 3$ *and* $\mathsf{m}_{\max} = 2$.

Proof sketch. Containment follows from the fact that optimal local schedules can be computed in polynomial time. For hardness, we reduce from the NP-complete problem 3-PARTITION which aims at partitionning a set of integers into triplets of the same sum B. We will create "triplet" organizations which benefit from the cooperation by starting one of their jobs earlier. A triplet organization can then accept to delay its jobs but only by a total processing time B, otherwise the schedule would not be individually rational. Other organizations own "integer" jobs with processing times matching the integers from the 3-PARTITION instance. To meet the sum of completion times objective, it will be necessary to schedule an integer job first on all machines, therefore delaying jobs from triplet organizations. To both fulfill individual rationality and meet the sum of completion times objective, it will be necessary to delay jobs from each triplet organizations by exactly B, which is only possible if the integers can be partitioned into triplets of sum B. □

Using a similar idea as for Theorem 3, we show that C_Σ-MOS is W[1]-hard with respect to k, i.e., it is unlikely to be in FPT according to current complexity assumptions.

Proposition 5 ($\star$). C_Σ-MOS *is W[1]-hard with respect to* k.

Similarly to Proposition 3, the sum of completion times case allows for an FPT result with respect to n using a simple brute-forcing approach.

Proposition 6 ($\star$). C_Σ-MOS *is* FPT *with respect to* n.

As the τ upper-bounds the number of jobs, the following corollary follows directly.

Corollary 3 ($\star$). C_Σ-MOS *is* FPT *with respect to* τ.

Finally, we use a dynamic programming approach similar to the one used in the proof of Proposition 4. As it is not possible to order the jobs according to phase, as it was done for the $\mathsf{C}_{\max}$-MOS case, we require an additional parameter for the dynamic programming approach to function.

Proposition 7 ($\star$). C_Σ-MOS *is* XP *with respect to* $\mathsf{p}_{\max} + \mathsf{m}$.

5 Conclusion

We introduce the concept of individual rationality into multi-organizational scheduling and explore the parameterized complexity of two optimization problems, $\mathsf{C}_{\max}$-MOS and C_Σ-MOS. For the former problem, an important open question remains: Is the problem fixed-parameter tractable (FPT) with respect to the maximum completion time $\mathsf{p}_{\max}$? Notably, the classical single-organization variant of this problem is already known to be FPT with respect to $\mathsf{p}_{\max}$.

Our research opens up several promising avenues for future work. First, an immediate extension is to consider the case of parallel jobs. Second, our framework can be applied to other scheduling problems, such as those with precedence constraints or hard deadlines. Third, it would be also interesting to study scenarios where each organization provides a precomputed local schedule as input. For C_Σ-MOS, the complexity remains unchanged, as optimal local schedules can be computed in polynomial time. For $\mathsf{C}_{\max}$-MOS, this setting reduces the problem to within NP, and preliminary investigations suggest that the parameterized results carry over.

Finally, an intriguing direction is to study scenarios where the local and global objectives differ. For instance, the global objective might be to minimize C_Σ, while individual rationality mandates that the makespan of each organization matches its locally optimal makespan. A related model was previously examined by Cohen *et al.* [2011], but without considering individual rationality as a constraint.

Acknowledgements

The authors are supported by the Vienna Science and Technology Fund (WWTF) [10.47379/ VRG18012]. We would like to thank the reviewers for their helpful comments.

References

[Ashlagi and Roth, 2012] Itai Ashlagi and Alvin E. Roth. New challenges in multihospital kidney exchange. *American Economic Review*, 102(3):354–359, 2012.

[Ashlagi and Roth, 2014] Itai Ashlagi and Alvin E. Roth. Free riding and participation in large scale, multi-hospital kidney exchange: Free riding. *Theoretical Economics*, 9(3):817–863, 2014.

[Biró *et al.*, 2019] Péter Biró, Walter Kern, Dömötör Pálvölgyi, and Daniël Paulusma. Generalized matching games for international kidney exchange. In *proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 413–421, 2019.

[Brucker, 2004] Peter Brucker. *Scheduling algorithms (4. ed.)*. Springer, 2004.

[Chakravorty *et al.*, 2013] Anirudh Chakravorty, Neelima Gupta, Neha Lawaria, Pankaj Kumar, and Yogish Sabharwal. Algorithms for the relaxed multiple-organization multiple-machine scheduling problem. In *proceedings of the 20th Annual International Conference on High Performance Computing*, pages 30–38, 2013.

[Chen *et al.*, 2025] Jiehua Chen, Martin Durand, and Christian Hatschka. Multi-organizational scheduling: Individual rationality, optimality, and complexity. Technical report, arXiv:2505.12377, 2025.

[Cohen *et al.*, 2011] 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.

[Cohen *et al.*, 2014] Johanne Cohen, Daniel Cordeiro, and Pedro Luis F Raphael. Energy-aware multi-organization scheduling problem. In *proceedings of the Euro-Par 2014 Parallel Processing: 20th International Conference*, pages 186–197, 2014.

[Cordeiro *et al.*, 2011] Daniel Cordeiro, Pierre-François Dutot, Grégory Mounié, and Denis Trystram. Tight analysis of relaxed multi-organization scheduling algorithms. In *proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium*, pages 1177–1186, 2011.

[Cygan *et al.*, 2015] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[Durand and Pascual, 2021] Martin Durand and Fanny Pascual. Efficiency and equity in the multi organization scheduling problem. *Theoretical Computer Science*, 864:103–117, 2021.

[Gourvès *et al.*, 2012] Laurent Gourvès, Jérôme Monnot, and Fanny Pascual. Cooperation in multiorganization matching. *Algorithmic Operations Research*, 7(2), 2012.

[Jansen *et al.*, 2013] Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39–49, 2013.

[Klimentova *et al.*, 2021] Xenia Klimentova, Ana Viana, João Pedro Pedroso, and Nicolau Santos. Fairness models for multi-agent kidney exchange programmes. *Omega*, 102:1–14, 2021.

[Knop *et al.*, 2020] Dušan Knop, Martin Kouteckỳ, and Matthias Mnich. Combinatorial n-fold integer programming and applications. *Mathematical Programming*, 184:1–34, 2020.

[Mnich and Van Bevern, 2018] Matthias Mnich and René Van Bevern. Parameterized complexity of machine scheduling: 15 open problems. *Computers & Operations Research*, 100:254–261, 2018.

[Mnich and Wiese, 2015] Matthias Mnich and Andreas Wiese. Scheduling and fixed-parameter tractability. *Mathematical Programming*, 154:533–562, 2015.

[Ooshita *et al.*, 2009] Fukuhito Ooshita, Tomoko Izumi, and Taisuke Izumi. A generalized multi-organization scheduling on unrelated parallel machines. In *proceedings of the 2009 International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 26–33, 2009.

[Ooshita *et al.*, 2012] Fukuhito Ooshita, Tomoko Izumi, and Taisuke Izumi. The price of multi-organization constraint in unrelated parallel machine scheduling. *Parallel Processing Letters*, 22(02), 2012.

[Papadimitriou, 1994] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[Pascual *et al.*, 2009] Fanny Pascual, Krzysztof Rzadca, and Denis Trystram. Cooperation in multi-organization scheduling. *Concurrency and Computation: Practice and Experience*, 21(7):905–921, 2009.

[Rzadca, 2007] Krzysztof Rzadca. Scheduling in multi-organization grids: measuring the inefficiency of decentralization. In *proceedings of the Parallel Processing and Applied Mathematics: 7th International Conference*, pages 1048–1058, 2007.

[Skowron and Rzadca, 2014] Piotr Skowron and Krzysztof Rzadca. Fair share is not enough: measuring fairness in scheduling with cooperative game theory. In *proceedings of the Parallel Processing and Applied Mathematics: 10th International Conference*, pages 38–48, 2014.

[Sönmez and Ünver, 2013] Tayfun Sönmez and Utku M. Ünver. Market design for kidney exchange. In *The Handbook of Market Design*, pages 92–137. Oxford University Press, 2013.

[Wagner, 1990] Klaus W. Wagner. Bounded query classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.