Coordination of the Decentralized Concurrent Open-Shop

Yossi Bukchin*  Eran Hanany*  Yigal Gerchak*

May 2020

Abstract

Problem definition: A system has to complete several jobs, where each job consists of multiple components that are processed simultaneously by different dedicated machines. Each job has a weight which represents its relative importance to the end customer. The components are sequenced on each machine in a decentralized manner with the objective of minimizing the weighted sum of disutility of job completion times. We analyze the coordination problem that arises. Academic/Practical Relevance: The framework models real-world decentralized processes such as assembly lines in the automobile industry, testing components of an electronic system, periodic maintenance service of an airplane, supply chain assembly systems and any other pre-assembly stage in a manufacturing environment. Methodology: The decentralized system is modeled as a non-cooperative game for two scenarios: (1) local completion times, where each machine considers only the completion times of their components, disregarding the other machines; and (2) global completion times, where each machine considers the job completion times from the perspective of the system, i.e. when all components of each job are completed. Results: Tight bounds are provided on the inefficiency that might occur in the decentralized system, showing potentially severe efficiency loss in both scenarios. We propose and investigate scheduling based, coordinating job weighting mechanisms that require minimal information, showing impossibility in the local completion times scenario and possibility using the related machines mechanisms in the global completion times scenario. These results extend to a setting with incomplete information in which only the distribution of the processing times is commonly known, and each machine is additionally informed about their own processing times. Managerial Implications: Decentralized concurrent open-shop systems under the

*Department of Industrial Engineering, Tel Aviv University. E-mail: bukchin@tau.ac.il, hananye@tau.ac.il, ygerchak@tau.ac.il
global completion times scenario may be effectively coordinated based on minimal information via scheduling based mechanisms. Contracting on the global completion times scenario is therefore preferable.

Key words: scheduling, assembly, weighted completion times, non-cooperative game.

1 Introduction

Operations often involve jobs that are broken up into components which are executed in parallel by dedicated machines, where on each machine any sequence for performing the components is possible. Practical examples include project activities performed in parallel by different, expert subcontractors, or components of industrial jobs that are processed simultaneously on multiple dedicated machines before the assembly stage. As shown in Figure 1, the common feature of these systems is that a project/job is completed only when all dedicated parallel machines have finished processing all the components of this project/job. The goal is to find the sequence of components on the machines which optimizes certain performance measures. In the scheduling literature, this problem is referred to as the “concurrent open-shop problem” or the “open-shop with jobs overlap problem”, as it is a relaxation of the well-known job-shop problem. The problem was motivated via assembly lines in the automobile industry, testing components of an electronic system, or periodic maintenance service of an airplane (Wagneur and Sriskandarajah, 1993), supply chain assembly systems (Chen and Hall, 2007) and any other pre-assembly stage in a manufacturing environment. Despite being applicable to multiple environments, for simplicity, the terminology we will use in this paper is of a system consisting of jobs, components and machines.

![Figure 1: Concurrent open-shop](image)

The extant research considers centralized systems, namely there is a single decision maker,
typically the system owner, who is capable of enforcing a coordinated schedule for all components. However, often the different machines are managed, or even owned, by different companies, possibly at different locations. In any such decentralized environment each machine will act based on its own cost structure and incentives. The resulting job component sequencing may be quite different from the centralized solution, thus may be sub-optimal for the system. Since the deterioration is domain and objective specific, it is interesting to study such decentralized systems in order to evaluate this loss.

In this paper we analyze the coordination problem arising in a decentralized concurrent open-shop system as a non-cooperative game. We investigate the system efficiency loss due to decentralization, and accordingly propose managerial strategies for coordination. We adopt the natural objective of minimizing the weighted sum of disutility of completion times. Special cases of this objective include minimizing the weighted sum of completion times and minimizing the weighted sum of discounted completion times (Pinedo 2012), where the latter is equivalent to maximizing the present value of a system's revenue. This objective guides all parties involved for reasons of congruence.

To demonstrate our approach, we rely on two real world applications taken from Wagneur and Sriskandarajah (1993), which we analyze in this paper as decentralized systems. The first application is an assembly system with different dedicated suppliers providing parts, which are then assembled to jobs in the final assembly stage. The objective of the plant as well as the suppliers may be to minimize the weighted sum of completion times. The customer of the assembly plant is likely to determine the job weights, probably based on the product price or the level of importance to the customer. In contrast, each supplier will typically have different job component weights, which are likely to be determined by the assembly plant. This situation can be found in practice when the plant manager is more powerful than the suppliers, however is not capable of managing their internal operations (see, e.g., Vairaktarakis and Aydinliyim, 2015).

In the second application, different dedicated teams perform maintenance operation for airplanes, where each airplane is ready only after all teams have finished their work on it. The maintenance of an airplane is then considered a job, while each team is analogue to a machine. Assuming the same type of objective as in the first application, the job weight of each airplane should be determined by the system, typically according to the relative importance of each air-
plane, possibly based on its size. Assuming that each team is independent in determining their own work procedures, i.e. the order of visiting the airplanes, the function in charge of the maintenance operation should determine the airplane job component weights for each team. A unified view is shown in Figure 1, namely the job weights are dictated to the system by the customer, while possibly different job component weights are dictated to the machines by the system. The decision regarding the job component weights for the machines can be used as an incentive mechanism. In this paper we propose and analyze such mechanisms.

Throughout the analysis of our game theoretic model, we distinguish between two possible scenarios for job completion times as viewed from the perspective of a machine: either local completion times, whereby each machine only cares about their own completion time of each job, or global completion times, whereby each machine cares about the system completion time of each job, namely when the job has completed processing on all machines. Each of these scenarios may arise in practice, depending on the agreement between the system manager and each machine.

We provide tight job weight dependent bounds for the efficiency loss that arises in the decentralized system when compared with a centralized one. The relevant weights in the bounds we characterize are from the perspective of the system. It is reasonable to assume that the system manager may be unable to collect reliable information about the processing times on the machines. Therefore we concentrate on bounds that hold for any instance of the problem, irrespective of the processing times. In fact, we show that the bounds also hold for any reasonable continuous, increasing disutility function. The same bounds hold both for the local and global completion time scenarios. Depending on the job weights, the resulting inefficiency may be severe.

We then study mechanisms for coordination, where the incentives provided to individual machines are scheduling based, i.e. keep the type of scheduling objective unchanged. We propose a new type of scheduling based mechanisms, called the job weighting mechanism. In such a mechanism, the system manager determines the job component weights for each machine in order to coordinate the system. We ask whether it is possible that a job weighting mechanism will always coordinate the system while requiring minimal information, in the sense that the job component weights for each machine will not depend on the specific processing times or disutility function. For the local completion times scenario we show that this is impossible. In contrast, we show that it is possible under global completion times, and exactly characterize the set of coordinating job weight-
ing mechanisms that require minimal information as the *related machines mechanisms*. In these, the job weights are identical for all machines up to multiplication by a constant of proportionality. Related machine mechanisms can therefore be thought of as cost sharing arrangements.

Finally, we extend the analysis to an environment of incomplete information, namely where there is uncertainty about the processing times. Each machine is informed about their own processing times, but only knows the distribution of processing times of the other machines. Then each machine determines the sequence the job components based on this information. The outcome of the decentralized system is identified as a (Bayesian) Nash equilibrium. Despite the potential complication that may arise, we are able to extend all of the aforementioned results to this general setting of incomplete information.

In the remainder of the introduction we review the most relevant literature. Most papers on scheduling games analyze parallel machines under various assumptions on job characteristics and scheduling objectives (see Kress et al. 2018 for a review of this literature). Here we discuss the few papers that analyze scheduling systems other than parallel machines. Wagneur & Sriskandarajah (1993), followed by Yoon & Sung (2004), provided an interesting result, showing that there is a permutation schedule, namely the same job sequence for all machines, which is optimal for any performance measure which increases with the completion time of jobs. Yoon & Sung (2004) have also established some properties of the problem, by which they derived conditions that, when met, can narrow the space that needs to be searched in order to find the optimal solution. A related problem in the Project Management literature is the “payment scheduling problem”, where each event is associated with positive or negative payment, and the present value criterion is used (Russell 1970; see also Grinold 1972, Dayanand and Padman 2001 and Pinedo 2012). Multiple variants of open-shop and the concurrent open-shop problems, with respect to models, objectives and solution procedures, are nicely reviewed in Anand and Panneerselvam 2015, and Framinan et al. 2019.

Leung et al. (2005) denoted the problem as the parallel dedicated (PD) machines problem. They investigated the complexity of this problem under the objective of minimizing the sum of increasing function of the job completion times, leading to the conclusion that the problem is strongly NP-hard even for three machines. Roemer (2006) has summed up the evolving literature stream of the concurrent open-shop. When weights are associated with the jobs, Mastrolilli et al. (2010) suggested that the problem is strongly NP-Hard, even for a 2-machine system with unit
weights.

Zhao (2013) considered a real-world application where a firm signs a contract with several subcontractors and studies the impact on the firm’s success. The paper addresses the project of developing the Boeing 787, the Dreamliner, in which the first delivery was delayed by 40 months with a cost overrun of at least $10B. Based on an empirical study, the author claims that a major part of the delays was caused by subcontractors, and could have been avoided had a proper risk-sharing mechanism been applied.

Hall and Potts (2003) studied conflicts and cooperation in supply chain scheduling, analyzing two-machine flow-shop scheduling decisions with batch deliveries. Their model addresses one supplier and several manufacturers who supply several customers. They show that cooperation between a supplier and a manufacturer may reduce the total cost in the system by at least 20%, or by up to 100%, depending upon the scheduling objective. Chen and Hall (2007) addressed cooperation mechanisms in assembly systems, consisting of multiple suppliers with dedicated capabilities and a single manufacturer, and evaluate the cost savings realized by cooperation between the decision makers. In their model they consider four problems: (1) the individual supplier’s problem; (2) the suppliers’ problem when they act jointly; (3) the manufacturer’s problem; (4) the system problem. The cost function they use for the suppliers is total completion time. For the manufacturer, however, they choose two cost functions: total completion time and maximum lateness. Bukchin and Hanany (2020) analyzed a decentralized two-machine job-shop system where each machine minimizes its own flow-time objective. They provide tight bounds on the maximal DC for flow-shop settings, and a simple, scheduling based mechanism which always generates efficiency. Hall and Liu (2008) and Aydinliyim and Vairaktarakis (2011) reviewed decentralized models related to scheduling. See also Selvarajah and Steiner (2009), Manoj et al. (2012), Vairaktarakis (2013) and Hamers et al. (2019) for related decentralized scheduling problems.

The paper is organized as follows: Section 2 describes the model, efficiency measures and two variations of the decentralized system with demonstrating examples. Section 3 provides tight bounds for the efficiency loss that arises, and studies the possibility of coordination using a scheduling based, job weighting mechanism that requires minimal information. Section 4 extends all the results to a general environment in which there is incomplete information about the processing times. Section 5 concludes.
2 Model

Our model of Concurrent Open-Shop Game (COSG) considers a system processing the jobs in $\mathcal{J} = \{1, \ldots, J\}$. Each job $j \in \mathcal{J}$ consists of components $m = 1, \ldots, M$ that are processed in parallel by machines $\mathcal{M} = \{1, \ldots, M\}$, each component $m$ on the corresponding machine $m \in \mathcal{M}$. Let $p_{jm} \geq 0$ denote the processing time of the relevant component of job $j$ on machine $m$, where $p_{jm} = 0$ indicates that job $j$ requires negligible processing on machine $m$. On each machine $m$ in the decentralized system, the $J$ components are processed in an order chosen by the machine, as specified by a permutation $S_m = (S_{jm})_{j \in \mathcal{J}}$ of $\{1, \ldots, J\}$, where $S_{jm} \in \{1, \ldots, J\}$ is the position of the component of job $j$ in the processing order of machine $m$ ($S_{jm} = 1$ means that $j$ is positioned first, $S_{jm} = 2$ means that $j$ is positioned second, $S_{jm} = J$ means that $j$ is positioned last). A schedule $S = (S_m)_{m \in \mathcal{M}}$ specifies the processing order of all jobs in all machines. A schedule $S$ where $S_m$ is identical for all machines $m$ is called a permutation schedule.

Given a schedule $S$, let $C_{jm}(S)$ denote the completion time of (the relevant component of) job $j$ in machine $m$ under schedule $S$, defined by $C_{jm}(S) = \sum_{j'|S_{jm} \leq S_{jm}} p_{j'm}$. Let $C_j(S)$ denote the system completion time of job $j$ under schedule $S$, defined by $C_j(S) = \max_m C_{jm}(S)$.

An important ingredient of any decentralized system is the identity of the decision makers, the machines and the system manager in our setting, and their objectives. We assume that each of them has a cost objective that is increasing in the job completion times. While the system manager always considers the system completion time of each job, $C_j(S)$, the machines may view the job completion times differently. Let $C^m_j(S)$ be the completion time of job $j$ from the perspective of machine $m$, namely, the completion time according to which the machine’s objective function is evaluated. This may be justified by the existence of an agreement between the system manager and each machine that determines their perspective in terms of $C^m_j(S)$. We consider the following two possible scenarios, as dictated by the aforementioned agreement.

**Definition 1.** In a decentralized setting with local completion times, the objective of each machine considers the completion time for each job component it processes, i.e., $C^m_j(S) = C_{jm}(S)$ for all $j,m$ and $S$.

With local completion times, each machine will aim at optimizing its own objective, regardless of and independently from the other machines’ schedules.
Definition 2. In a decentralized setting with global completion times, the objective of each machine considers the system completion time for each job, namely, the time at which all components of the job have been completed on all machines, i.e. $C^m_j(S) = C_j(S)$ for all $j, m$ and $S$.

Let $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ be a continuous, increasing function, representing the disutility from completion time, with $f(0) = 0$ and $\lim_{x \to \infty} \frac{f(x)}{f(x+1)} = 1$ (this limit condition is guaranteed for example if $f$ is a polynomial or is increasing and concave). Two important special cases for this function are the linear case, $f(x) = x$, and the discounted case, $f(x) = 1 - \alpha x$ for some $0 < \alpha < 1$.

Further assume that each job $j$ has a weight $w_j \in (0, 1)$, which represents the job’s importance to the customer, normalized with $\sum_j w_j = 1$ without loss of generality. Similarly, from the perspective of each machine $m$, each job $j$ has a weight $w^m_j \in (0, 1)$, which represents the allocation of the job’s importance across its components, where $\sum_m w^m_j = w_j$. As explained in the introduction, it is reasonable to assume that the job weights $w_j$ are given to the system by the end customer, whereas the job component weights $w^m_j$ are dictated by the system manager to the machines as part of a contract signed between them. We will elaborate on this contract when discussing incentive mechanisms below. To summaries the elements of the model so far, for each decentralized scenario, either with local or with global completion times, an instance of COSG is given by $[(p_{jm}) \forall j,m, (w^m_j) \forall j,m, f]$.

The objective function of the system is to process according to the schedule $S$ which minimizes the weighted sum of disutility of completion times, $F(S) \equiv \sum_j w_j f[C_j(S)]$, i.e., the schedule that solves $\min_S F(S)$. Denote such a system optimal centralized schedule by $S^O$. The objective of each machine $m$ is to set their own part of the schedule, $S_m$, so as to minimize their own weighted sum of disutility of completion times, $F^m(S) \equiv \sum_j w^m_j f[C^m_j(S_m, S_{-m})]$, taking as given the schedule parts of the other machines $S_{-m} = (S_1, S_2, ..., S_{m-1}, S_{m+1}, ..., S_M)$, i.e., the schedule that solves $\min_{S_m} F^m(S)$. This leads to an equilibrium, decentralized outcome of the non-cooperative game (Nash 1951), denoted by $S^E$, where machine $m$’s part is denoted by $S^E_m$. These definitions are relevant for both the local and global completion times scenarios.

We consider the above schedules from the system’s perspective. An equilibrium schedule $S^E$ may not be optimal for the system, hence we use the optimal schedule $S^O$ as a benchmark. Since there may be multiple equilibria, denote by $S^B$ some equilibrium which minimizes the system’s disutility
\[ F(S^E) \] among all equilibria (i.e., best equilibrium). Similarly, denote by \( S^W \) some equilibrium which maximizes the system’s disutility among all equilibria (i.e., worst equilibrium). We consider the inefficiency generated in equilibrium using the Decentralization Cost (DC), defined as \( \frac{F(S^B)}{F(S^O)} \), and using the Price of Anarchy (PoA), defined as \( \frac{F(S^W)}{F(S^O)} \). The terms PoA and DC are respectively due to Koutsoupias and Papadimitriou (1999) and Bukchin and Hanany (2007), where Anshelevich et al. (2008) use the term Price of Stability (PoS) for the same definition as DC. These measures express the system’s loss when the decentralized solution is applied (either best or worst Nash equilibrium) rather than the centralized solution (global optimum).

We now elaborate on the two scenarios defined above, namely local vs. global completion times. The local completion times scenario is the outcome of a natural agreement, as each machine is evaluated according to their own performance, irrespective of the other machines. The global completion times scenario is more difficult to implement, however it has the potential to be better aligned with the system’s objective. Still, such an agreement has been applied in the manufacturing of the Boeing 787 Dreamliner, whereby many of the part suppliers agreed not to be paid until the planes get delivered (Vairaktarakis and Aydinliyım, 2015). In this case, the completion time of a job from the point of view of each machine depends also on the decisions made by the other machines, thus each machine’s best schedule depends on the schedules the other machines have selected. In such a non-cooperative game, each machine aims to minimize its own weighted sum of disutility of completion times, taking as given the schedule parts of the other machines, thus generating a Nash equilibrium.

When there is a single machine in the system, in both the local and the global completion times scenarios the machine’s problem is clearly equivalent to the system’s problem. The following simple, 2-job, 2-machine example compares the scenarios of local vs. global completion times, and highlights the emerging differences.

**Example 1.** Consider two jobs with weights \( w_1 = \frac{3}{5} \) and \( w_2 = \frac{2}{5} \) to be processed on two machines, where the processing times for job 1 are \( p_{11} = 2 \), \( p_{12} = 1 \), and for job 2 are \( p_{21} = 1 \), \( p_{22} = 4 \). The job component weights for the machines are \( w_1^1 = 0.51 \), \( w_1^2 = 0.09 \) for job 1 and \( w_2^1 = 0.30 \), \( w_2^2 = 0.10 \) for job 2. The system’s and the machines’ objectives are to minimize the weighted sum of completion times, i.e. the disutility function is \( f(x) = x \). Each machine \( m \) has two possible permutations \( S_m \),
either (1, 2) or (2, 1). Each schedule generates different job completion times on each machine, as shown in Figure 2. Each row represents a possible permutation for machine 1, and each column a possible permutation for machine 2. The following tables present, for each schedule \( S = (S_1, S_2) \), each machine’s objective value, \( F^1(S), F^2(S) \), and in parenthesis the system’s objective value, \( F(S) \); this is given for local completion times in left table, and for global completion times in the right table. Note that in the global completion times scenario, the system’s objective value is equal to the sum of the objectives values of the machines.

For the local completion times scenario, both machine 1 and machine 2 have a strictly dominant strategy: for machine 1 it is setting the permutation (2, 1), and for machine 2 it is the permutation (1, 2). Thus there is a unique Nash equilibrium with this schedule (underlined). The system, however, would strictly prefer the machines to set the permutation schedule (1, 2) (marked in bold). The resulting DC and PoA in this case are equal to \( \frac{38}{32} = 1.1875 \). The decentralized environment is costly for the system, as it generates a 18.75% higher cost as compared with a centralized system. In contrast, for the global completion times scenario, both machines have a dominant strategy of setting the permutation (1, 2) (this dominance is strict for machine 2 and weak for machine 1, as machine 1 would be indifferent between the two permutations if machine 2 set the permutation (2, 1)). Thus there is a unique permutation schedule equilibrium (1, 2). Since
the system’s preference is the same for both scenarios, the resulting DC and PoA are now equal to 1, i.e. the decentralization is not costly.

Note, however, that it is possible for the two scenarios to generate the exact opposite comparison. For instance, this would occur in this example if we just swapped the job component weights of job 2 to \( w_2^1 = 0.10, w_2^2 = 0.30 \), with no further changes.

\[
\begin{array}{ccc}
\text{local} & (1,2) & (2,1) \\
(1,2) & 1.32,1.59 (3.20) & 1.32,1.65 (4.60) \\
(2,1) & 1.63,1.59 (3.80) & 1.63,1.65 (4.60) \\
\text{global} & (1,2) & (2,1) \\
(1,2) & 1.52,1.68 (3.20) & 2.95,1.65 (4.60) \\
(2,1) & 2.03,1.77 (3.80) & 2.95,1.65 (4.60) \\
\end{array}
\]

With this modification, the local completion times scenario now has the optimal permutation schedule \((1,2)\) as the unique equilibrium, thus the resulting DC and PoA are equal to 1. In contrast, under global completion times there are two equilibria, with machine 2 setting the permutation \((2,1)\) and machine 1 being indifferent in their choice. The two equilibria have the same objective value for the system, so the resulting DC and PoA are equal to \(46 \div 3.2 = 1.4375\) for global completion times. The decentralization is now very costly only for global completion times.

As demonstrated in Example 1, the estimation of the DC and PoA is highly relevant for the system, since this would make clear whether it is worthwhile trying to affect the outcome of the decentralized system towards the centralized optimum. To this end, the system manager may implement an incentive mechanism. It is reasonable to restrict attention to scheduling based mechanisms, i.e. those that keep the type of scheduling objective unchanged. In our model, each machine would still minimize their own weighted sum of disutility of completion times, either under the local or the global completion times scenarios. Since job processing times and job weights are given in the system and cannot be altered, it is natural to define a mechanism as determining the job component weights from the perspective of the machines. The job component weights represent relative importance, thus a mechanism is implemented by setting specific weights. As explained above, these weights are dictated by the system manager to the machines as part of the contract signed between them.

**Definition 3.** A job weighting mechanism is a scheduling based mechanism in which the job component weights are dictated to the machines by the system manager.
Therefore, the job weighting mechanism determines, for any job weights \((w_j)\) of each job \(j\) from the perspective of each machine \(m\), where \(\sum_m w_j^m = w_j\) for all \(j\). One natural candidate for such a mechanism is the following.

**Definition 4.** Given an instance \([(p_{jm})_{j,m}, (w_j^m)_{j,m}, f]\), the proportional mechanism is a job weighting mechanism that allocates the job weights to the machines proportionally to the processing times of their components for this job, i.e.,

\[
  w_j^m = \frac{p_{jm}}{\sum_{m'} p_{jm'}} w_j.
\]

One may view the proportional mechanism as a default arrangement according to which the machines are paid proportionally to their processing times, thus the job component weights represent a fixed percentage of these costs. Interestingly, with this mechanism and the objective of minimizing the weight sum of completion times, there always exists a permutation equilibrium under the local completion times scenario, whereby the job components are sequenced on each machine in a non-increasing order of the ratio \(\frac{w_j}{\sum_m p_{jm}}\). The next example demonstrates possible implications of the proportional mechanism.

**Example 2.** Consider again two jobs, now with equal weights \(w_1 = w_2 = \frac{1}{2}\), to be processed on two machines, where the processing times for job 1 are \(p_{11} = 1, p_{12} = 3\), and for job 2 are \(p_{21} = 3, p_{22} = 1\). The job component weights for the machines are determined by the proportional mechanism, namely \(w_1^1 = 0.125, w_1^2 = 0.375\) for job 1 and \(w_2^1 = 0.375, w_2^2 = 0.125\) for job 2.

The system’s and the machines’ objectives are still to minimize the weighted sum of completion times, i.e. the disutility function is \(f(x) = x\). Now the objective value tables for local vs. global completion times are as follows:

<table>
<thead>
<tr>
<th></th>
<th>(1,2)</th>
<th>(2,1)</th>
<th></th>
<th>(1,2)</th>
<th>(2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>local</td>
<td>1.625,1.625 (3.5)</td>
<td>1.625,1.625 (4.0)</td>
<td>global</td>
<td>1.875,1.625 (3.5)</td>
<td>2.000,2.000 (4.0)</td>
</tr>
<tr>
<td>(1,2)</td>
<td>1.625,1.625 (4.0)</td>
<td>1.625,1.625 (3.5)</td>
<td></td>
<td>(2,1)</td>
<td>2.000,2.000 (4.0)</td>
</tr>
<tr>
<td>(2,1)</td>
<td></td>
<td></td>
<td></td>
<td>1.625,1.875 (3.5)</td>
<td></td>
</tr>
</tbody>
</table>

For the local completion times scenario (left table), both machine 1 and machine 2 are indifferent between the two permutations. Therefore any schedule forms an equilibrium, generating DC equal to 1 and PoA equal to \(\frac{4.0}{3.5} \approx 1.1429\), i.e. up to 14.29% higher cost in the decentralized setting as compared with a centralized one (a small modification to the processing times would even generate DC strictly greater than 1). In contrast, for the global completion times scenario (right table),
each machine strictly best responds with an identical permutation as the other machine, thus both permutation schedules form an equilibrium. Since both permutation schedules are optimal for the system, the resulting DC and PoA in this case are equal to 1.

As in Example 1, it is possible also under the proportional mechanism for the two scenarios to generate a very different comparison. For instance, this would occur in this example if we just changed the processing time of job 1 on machine 2 to \( p_{12} = 2 \), with corresponding changes in the job component weights for each machine according to the proportional mechanism.

<table>
<thead>
<tr>
<th>local</th>
<th>(1,2)</th>
<th>(2,1)</th>
<th>global</th>
<th>(1,2)</th>
<th>(2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>1.667,1.042 (3.0)</td>
<td>1.667,1.125 (3.5)</td>
<td>(1,2)</td>
<td>1.667,1.042 (3.0)</td>
<td>1.667,1.125 (3.5)</td>
</tr>
<tr>
<td>(2,1)</td>
<td>1.792,1.042 (3.5)</td>
<td>1.792,1.125 (3.5)</td>
<td>(2,1)</td>
<td>1.792,1.708 (3.5)</td>
<td>1.792,1.708 (3.5)</td>
</tr>
</tbody>
</table>

With this modification, the local completion times scenario now has the optimal permutation schedule (1,2) as the unique equilibrium, thus the resulting DC and PoA are equal to 1. In contrast, under global completion times there are two equilibria, with machine 1 setting the permutation (2,1) and machine 2 being indifferent in their choice. The two equilibria have the same objective value for the system, so the resulting DC and PoA are equal to \( \frac{35}{30} \approx 1.1667 \). The decentralization is now very costly only for global completion times.

Given the examples above, our aim in the following sections is to investigate bounds on the DC and PoA when considering all possible instances of our COSG, and to investigate the possibility of coordination using an incentive mechanism. To this end, for the benchmark centralized system, we may rely on Wagneur & Sriskandarajah (1993)’s analysis, which established that there is always an optimal permutation schedule. For completeness of the presentation, we give here their result and its proof.

**Proposition 1.** There always exists an optimal permutation schedule \( S^O \), i.e. \( S^O_m \) is the same for all \( m \).

**Proof.** Consider any schedule \( S \) which is not a permutation schedule. Let \( j \) and \( j' \) be two distinct jobs which are not ordered in the same way in all machines according to \( S \), where without loss of generality we may assume \( C_{j'}(S) \leq C_j(S) \), and let \( m' \) be a machine in which \( j \) precedes \( j' \) according to \( S \). Let \( S' \) be the schedule obtained from \( S \) by changing on \( m' \) the location of job \( j \) to be the
immediate successor of \( j' \), with no other changes. Then \( C_{jm'}(S') = C_{j'm'}(S) \leq \max_{\hat{m}} C_{j'\hat{m}}(S) = C_{j'}(S) \leq C_j(S) \), and for any machine \( m \neq m' \), \( C_{jm}(S') = C_{jm}(S) \leq \max_{\hat{m}} C_{j\hat{m}}(S) = C_j(S) \). It follows that \( C_j(S') = \max_{\hat{m}} C_{j\hat{m}}(S') \leq C_j(S) \). Additionally, \( C_j(S') \leq C_j(S) \) for all \( \hat{j} \neq j \) because no such job \( \hat{j} \) was delayed. Since the objective function of the system, \( F(S) \), is increasing in each job’s completion time, schedule \( S' \) is weakly better than \( S \). Therefore, applying this argument repeatedly starting from any optimal schedule generates a sequence of optimal schedules ending at an optimal permutation schedule \( S^O \).

3 Inefficiency bounds and coordination

We begin our analysis by providing bounds for the DC and PoA that may occur in each of the scenarios considered, namely local and global completion times. Recall that from the perspective of the system manager, the job weights \( w_j \) are given by the end customer, while the job component weights \( w_{jm} \) and the processing times \( p_{jm} \) are not given because, as explained in Section 2, the former is determined by the system using the job weighting mechanism and the latter are not necessarily known to the system. Therefore our approach is to develop the inefficiency bounds for given job weights \( w_j \) while considering all possible instances in any other respects. Our main result follows.

**Theorem 1.** In a decentralized system with either local or global completion times, the DC and PoA of all instances are bounded from above by \( \frac{1}{\min_j w_j} \). Moreover, for all job weights \((w_j)_{j=1}^n\) and disutility functions \( f \) this bound is tight, i.e. there exists an instance achieving this bound.

Prior to the proof, we now sketch the argument for Theorem 1. The bound is derived from the fact that in any schedule, the absence of idle times implies that the system completion time of any job is at most the system makespan, with equality for at least one job. Consequently, the system’s objective value at any equilibrium is bounded from above by the system makespan multiplied by the sum of job weights, while the system’s objective value at any optimum is bounded from below by the system makespan times the minimal job weight, which generates the bound. Note that without the normalization \( \sum_j w_j = 1 \), the bound would be \( \frac{\sum_j w_j}{\min_j w_j} \). The bound is shown as tight for DC, thus also for PoA, by constructing a sequence of instances with DC approaching this bound. In each instance along the sequence, under a linear disutility function, the minimal weight
job, say job $J$, has a large processing time on machine 1, all the remaining jobs have small, positive processing times on this machine, and the processing times on all other machines are zero for all jobs. Consequently, only the permutation chosen by machine 1 matters, and it is optimal for the system that this machine places job $J$ last. Additionally, the instance is constructed so that the job weight of job $J$ from the perspective of machine 1 is sufficiently large as compared to the weights of all other jobs, so that the uniquely optimal permutation from the perspective of this machine is to place job $J$ first. This leads to an inefficient equilibrium with DC value approaching the bound. The proof requires a more general argument, as it establishes the result for any disutility function $f$.

**Proof.** Fix any instance $[(p_{jm}, w_{jm}, v_{jm}, f)]$ such that $\sum_m w_{jm} = w_j$ for all $j$ for the given job weights $w_j$. For any schedule $S$, the system makespan is

$$M \equiv \max_j C_j(S) = \max_m \max_j C_{jm}(S) = \max_m \sum_j p_{jm},$$

thus the system makespan does not depend on $S$. Consider an optimal schedule $S^O$, and let $m^L$ be the machine that determines the system makespan in $S^O$, i.e. for which $\sum_j p_{jm^L} = M$. For the last job on machine $m^L$ according to $S^O$, i.e. the job $j^L$ for which $S^O_{j^L m^L} = J$, the completion time of $j^L$ in the system and on this machine are both equal to the system makespan, i.e. $C_{j^L}(O) = C_{j^L m^L}(O) = \sum_j p_{jm^L} = M$. Therefore the DC and PoA are bounded from above by

$$\frac{F(S^B)}{F(S^O)} = \frac{\sum_j w_j f[C_j(S^B)]}{\sum_j w_j f[C_j(S^O)]} \leq \frac{\sum_j w_j f[C_j(S^W)]}{\sum_j w_j f[C_j(S^O)]} \leq \frac{\sum_j w_j f(M)}{\sum_j w_j f[C_j(S^O)]} \leq \frac{f(M)}{\sum_j w_j f(M)} \leq \frac{1}{\min_j w_j},$$

where the first inequality follows from the definitions of DC and PoA, the second inequality follows because the system completion time of any job is at most the system makespan and $f$ is increasing, the third inequality follows because the system completion time of any job is at least zero and $f(0) = 0$, and the fourth inequality follows because the weight of any job is at least the minimal weight of all jobs.
To see that for all job weights \((w_j)_{v_j}\) and disutility functions \(f\) the upper bound is tight for DC, thus also for POA which is always weakly above DC, consider the following instance: indexing the jobs in decreasing order of \(w_j\), so that in particular \(w_J = \min_j w_j\), for a sufficiently small \(\varepsilon > 0\) and all \(j' < J\) and \(m' > 1\), set \(p_{j'1} = 1\) if \(p_{j'1} = \frac{w_{j'}}{\varepsilon} - M\) and \(p_{j'm'} = p_{Jm'} = 0\), and set \(w'_{j'} = \left[1 - \frac{f(p_{j'1})}{f(1+p_{j'1})}\right]^2\), \(w_J' = w_J - (M-1)\varepsilon\), \(w''_{j'} = \frac{w'_{j'} - w_{j'}}{M-1}\) and \(w''_{J'} = \varepsilon\). In this case, \(C^m_{jm'}(S) = 0\) for any \(j, S\) and \(m' > 1\). Therefore \(C_j(S) = C^1_j(S) = C_{j1}(S)\) depends only on \(S_1\). Additionally, any machine \(m' > 1\) is indifferent between all permutations, either because \(C^m_{jm'}(S) = C_{jm'}(S)\) for local completion times, or because \(C^m_{jm'}(S) = C_j(S)\) for global completion times. For any \(1 \leq k \leq J\), let \(S_k\) be the permutation where for all \(m\), \(S_{jm} = k\), and all \(j' < J\) are in increasing order of \(j'\). Since \(p_{j'1}\) is identical for all jobs \(j' < J\) and \(w_j\) is decreasing in \(j\), the permutation schedule \(S_k\) is optimal for the system among all schedules \(S\) for which \(S_{J1} = k\). Permutation \(S_k\) is also weakly dominant for machine 1 among all permutations \(S_j\) for which \(S_{J1} = k\). Then, for sufficiently small \(\varepsilon > 0\), \(S^1\) is the unique equilibrium permutation for machine 1. To see this, note that for any \(1 < k \leq J\),

\[
F^1(S_k) = \sum_j w_j f[C_j^1(S_k)] = \sum_j w_j f[C_{j1}(S_k)] = \sum_{j' = 1}^{k-1} w_{j'} f(j') + w_{j'} f(k-1 + p_{J1}) + \sum_{j' = k}^{J-1} w_{j'} f(j' + p_{J1}) > w_j f(p_{J1}) + \sum_{j' = 1}^{J-1} w_{j'} f(j' + p_{J1}) = \sum_j w_j f[C_j^1(S^1)] = F^1(S^1)
\]

if and only if

\[
\sum_{j' = 1}^{k-1} \frac{f(j') - f(j' + p_{J1})}{f(1 + p_{J1})} + w_{j'} \frac{f(k-1 + p_{J1}) - f(p_{J1})}{w_{j'} f(1 + p_{J1})} > 0,
\]

where this inequality holds for sufficiently small \(\varepsilon > 0\) because its left-hand side approaches \(+\infty\) when \(\varepsilon \to 0\). Thus the permutation schedule \(S^1\) is a best equilibrium \(S^B\), where \(F(S^B) = \sum_j w_j f(C_j(S^B)) = w_J f(p_{J1}) + \sum_{j' = 1}^{J-1} w_{j'} f(j' + p_{J1})\). In contrast, \(S^J\) is an optimal schedule \(S^O\). To see this, note that for any \(1 \leq k < J\),

\[
F(S^k) = \sum_j w_j f[C_j(S^k)] = \sum_{j' = 1}^{k-1} w_{j'} f(j') + w_{j'} f(k - 1 + p_{J1}) + \sum_{j' = k}^{J-1} w_{j'} f(j' + p_{J1})
\]

\[
> \sum_{j' = 1}^{J-1} w_{j'} f(j') + w_J f(J - 1 + p_{J1}) = \sum_j w_j f[C_j(S^J)] = F(S^J)
\]
if and only if
\[
\sum_{j'=1}^{k-1} w_{j'} f(j') \frac{f(j')}{f(J-1+p_{J1})} + w_J f(k-1+p_{J1}) + \sum_{j'=1}^{J-1} w_{j'} f(j') \frac{f(j'+p_{J1})}{f(J-1+p_{J1})} > 1,
\]

where this inequality holds for sufficiently small \( \varepsilon > 0 \) because its left-hand side approaches \( 1 + \frac{1}{w_J} \sum_{j'=1}^{J-1} w_{j'} > 1 \) when \( \varepsilon \to 0 \). This leads to the DC value of \( \frac{w_J f(p_{J1}) + \sum_{j'=1}^{J-1} w_{j'} f(j'+p_{J1})}{\sum_{j'=1}^{J-1} f(j') + w_J f(J-1+p_{J1})} \), which approaches \( \frac{1}{\min_j w_j} \) as \( \varepsilon \to 0 \).

Given the potentially severe inefficiency shown in Theorem 1, we are interested in coordinating mechanisms set by the system that may resolve the problem. As explained in Section 2, we propose the job weighting mechanisms, which for any given system job weights \( w_j \), determine the job component weights \( w_{jm} \) of each job \( j \) from the perspective of each machine \( m \), with \( \sum_m w_{jm} = w_j \) for all \( j \). The goal is to set the job weighting mechanism so that it always generates full efficiency. Furthermore, we ask whether it is possible for these incentive mechanisms to have minimal informational requirements, namely that they will not depend on any of the job processing times or disutility function. These requirements are summarized as follows.

**Definition 5.** A job weighting mechanism is coordinating if both the DC and PoA are equal to 1 for all processing times \( (p_{jm})_{\forall j,m} \), job weights \( (w_j)_{\forall j} \) and disutility functions \( f \). A job weighting mechanism requires minimal information if for all job weights \( (w_j)_{\forall j} \), the allocated job component weights \( (w_{jm})_{\forall j,m} \) do not depend on the processing times \( (p_{jm})_{\forall j,m} \) or the disutility function \( f \).

As shown in Example 2, the proportional mechanism fails to be coordinating. Moreover, by definition it also fails the requirement of minimal information because it directly depends on the processing times. We will therefore pay attention instead to the following natural job weighting mechanism that does require minimal information.

**Definition 6.** Given job weights \( w_j \) for all \( j \), a job weighting mechanism \( w_{jm} \) is said to be a related machines mechanism if, for some \( w_m \) for each \( m \), the job component weights satisfy \( w_{jm} = w_j w_m \) for all \( j,m \).

Note that any distribution of weights \( w_m \) for each \( m \) defines some related machines mechanism, and since \( \sum_j w_j = 1 \), for any related machines mechanism with weights \( w_m \) for each \( m \) we have \( \sum_m w_m = 1 \).
Our first result on job weighting mechanisms concerns the impossibility of achieving the desired coordination under the local completion times scenario.

**Theorem 2.** In a decentralized system with local completion times, there does not exist a coordinating job weighting mechanism that requires minimal information.

The argument for Theorem 2 is to consider a case where two jobs have equal weights, and show that for each possible job weighting mechanism that requires minimal information, there exists an instance with the given job weights for which the mechanism fails to achieve efficiency. The corresponding instance is constructed with processing times that are positive only for these two jobs and only on two machines. Moreover, these processing times are such that the two machines choose different permutations in any equilibrium, whereas the system optimum is a permutation schedule, thus generating the inefficiency. Interestingly, the positive processing times are the same in almost all the constructed instances, namely different processing times are required in the argument only when considering a related machines mechanism.

**Proof.** It is sufficient to show that all job weighting mechanisms that require minimal information fail to generate DC or PoA equal to 1 for some instance with given job weights, for example when two jobs, say jobs $J_{1-1}$ and $J$, have equal job weights, i.e. $w_{1-1} = w_j$. In fact, we show this holds for any disutility function $f$. Consider the following two mutually exclusive and exhaustive cases: (i) any related machines mechanism, i.e. $w^m_j = w^m_j$ for all $j, m$, and (ii) any job weighting mechanism that is not of the related machines type. Begin with case (i). Consider any disutility function $f$, and for all $j' < J - 2$ and $m' > 2$, set $p_{j'1} = p_{j'2} = 0$, $p_{(J-1)1} = p_{J2} = 1$, $p_{(J-1)2} = p_{J1} = 2$ and $p_{j'm'} = p_{(J-1)m'} = p_{Jm'} = 0$. In this case, for any $j, S$ and $m' > 2$, $C^m_j(S) = C^m_{jm'}(S) = 0$, so any machine $m' > 2$ is indifferent between all permutations. Since $p_{j'1} = 0$ for all $j' < J$ and $w^1_{J-1}f(p_{(J-1)1}) + w^1_jf(p_{(J-1)1} + p_{J1}) = w^1_jw^1[f(1) + f(3)] < w^1_jw^1[f(2) + f(3)] = w^1_{J-1}f(p_{(J-1)1} + p_{J1}) + w^1_jf(p_{J1})$, any permutation $S_1 = (..., J - 1, J)$ is weakly dominant for machine 1. Similarly, any permutation $S_2 = (..., J, J - 1)$ is weakly dominant for machine 2. Therefore any equilibrium schedule, including the best $S^B$, has $S_1, S_2$ with $F(S^B) = w_{(J-1)}f(\max\{1, 1 + 2\}) + w_Jf(\max\{1 + 2, 1\}) = 2w_Jf(3)$. Additionally, any permutation schedules $(..., J - 1, J)$ and $(..., J, J - 1)$ are optimal with $F(S^O) = w_{(J-1)}f(\max\{1, 2\}) + w_Jf(\max\{1 + 2, 2 + 1\}) = w_J[f(2) + f(3)]$. Thus $PoA = DC = \frac{2f(3)}{f(2)+f(3)} > 1$. 

18
Now consider case (ii). Note that since this case considers any job weighting mechanism other than the related machines and since \( w_{J-1} = w_J \), there must exist two machines, indexed as 1 and 2, such that \( w_{j-1}^1 > w_j^1 \) and \( w_{j-1}^2 < w_j^2 \). Consider any disutility function \( f \), and for all \( j' < J-2 \) and \( m' > 2 \), set \( p_{j'1} = p_{j'2} = 0 \), \( p_{(J-1)1} = p_{(J-1)2} = p_{J1} = 1 \) and \( p_{j'm'} = p_{Jm'} = 0 \). In this case, as in case (i), for any \( j, S \) and \( m' > 2 \), \( C_{m'}^j(S) = C_{Jm'}^j(S) = 0 \), so any machine \( m' > 2 \) is indifferent between all permutations. Since \( p_{j'1} = p_{j'2} = 0 \) for all \( j' < J \), \( w_{J-1}^1 f(1) + w_J^1 f(2) < w_{J-1}^1 f(2) + w_J^1 f(1) \) and the reverse inequality holds for machine 2, any permutation \( S_1 = (..., J-1, J) \) is weakly dominant for machine 1 and any permutation \( S_2 = (..., J, J-1) \) is weakly dominant for machine 2. Therefore any equilibrium schedule, including the best \( S^B \), has \( S_1, S_2 \) with \( F(S^B) = w_{J-1} f(\max\{1, 1+1\}) + w_J f(\max\{1+1, 1\}) = 2w_J f(2) \). Additionally, any permutation schedules \((..., J-1, J)\) and \((..., J, J-1)\) are again optimal with \( F(S^O) = w_{J-1} f(1) + w_J f(1+1) = w_J[f(1) + f(2)] \). Thus \( \text{PoA} = DC = \frac{2f(2)}{f(1)+f(2)} > 1 \). \( \square \)

Theorem 2 shows that for the case of local completion times, a job weighting mechanism that achieves efficiency for all instances necessarily requires the information about the specific processing times in each instance. The negative result described in Theorem 2 is dramatically overturned when considering the global completion times scenario, as shown by our next main result.

**Theorem 3.** In a decentralized system with global completion times, there is a range of coordinating job weighting mechanisms that include the related machines mechanisms. Furthermore, a minimal information requiring job weighting mechanism is coordinating if and only if it is a related machines mechanism.

Intuitively, the first part of Theorem 3 follows since any related machines mechanism is similar in nature to a cost sharing arrangement with fixed allocations of the system cost across the machines. In this way, the incentives of all parties involved become aligned. Since the set of feasible schedules is finite, this alignment holds within a range of possible weights around any coordinating job weighting mechanism. The argument for the second part is to consider a minimal information requiring job weighting mechanism that is not of the related machines type, for which there necessarily exist two jobs with relative job weights that are different from their relative job component weights on some machine, and this holds irrespective of the processing times. Constructing an instance such that the processing times are positive only for these two jobs on this machine, only the permutation
chosen by this machine matters. Moreover, under a linear disutility function, these processing times are constructed to have a ratio in between the ratio of job weights for the system and the ratio of job component weights for the machine. Therefore the machine strictly prefers to deviate from any optimal schedule, thus the coordination requirement is violated. Also here, the proof requires a more general argument, as it establishes the result for any disutility function \( f \).

**Proof.** Fix any processing times \((p_{jm})_{\forall j,m}\), job weights \((w_j)_{\forall j}\) and disutility functions \( f \), and let \(S^O\) be a corresponding optimal permutation schedule. Consider a related machines mechanism with weights \(w^m\) for each \(m\), i.e., \(w^m_j = w_j w^m\) for all \(j,m\). Then for each \(m\) and \(S_m\),

\[
F^m(S^O) = \sum_j w^m_j f[C^m_j(S^O_m, S^O_{-m})] = w^m \sum_j w_j f[C^m_j(S^O_m, S^O_{-m})] 
\leq w^m \sum_j w_j f[C^m_j(S_m, S^O_{-m})] = \sum_j w^m_j f[C^m_j(S_m, S^O_{-m})] = F^m(S_m, S^O_{-m}),
\]

where the inequality follows from optimality of \(S^O\). Note that the cost objective value of machine \(m\) is equal to the system cost objective value times \(w^m\), namely the machine cost is proportional to the system’s in any schedule \(S\), thus any non-optimal schedule for the system is also non-optimal for the machine. Therefore a schedule is optimal if and only if it is an equilibrium, and both the DC and PoA are equal to 1. Furthermore, since fixing any schedule \(S\), \(\sum_j w^m_j f[C^m_j(S_m, S_{-m})]\) is continuous in each \(w^m_j\), the set of inequalities \(\sum_j w^m_j f[C^m_j(S^O_m, S^O_{-m})] \leq \sum_j w^m_j f[C^m_j(S_m, S^O_{-m})]\) for each \(m\) and \(S_m\) remains valid for a range of job component weights \(w^m_j\) around \(w_j w^m\) for all \(j,m\). Thus the DC and PoA remain equal to 1 for all job weighting mechanisms assigning job component weights in this range.

It remains to show that for all minimal information requiring job weighting mechanisms and all job weights \((w_j)_{\forall j}\), if the DC and PoA are equal to 1 for all processing times \((p_{jm})_{\forall j,m}\) and disutility functions \(f\), then necessarily \(w^m_j = w_j w^m\) for all \(j,m\). Fix job weights \((w_j)_{\forall j}\), and suppose that \(w^m_j\) is not given by a related machines mechanism, so without loss of generality \(\frac{w_1^1}{w_1} \neq \frac{w_1^2}{w_2}\). Consider an instance defined with any disutility function \(f\), and with \(p_{11} = 1\), \(p_{21}\) satisfying \(\frac{f(1+p_{21})-f(1)}{f(1+p_{21})-f(p_{21})} = \frac{1}{2} (\frac{w_2}{w_1} + \frac{w_1}{w_2})\), and \(p_{jm} = 0\) for all other \(j,m\). To see that \(p_{21}\) is well defined, note that the r.h.s of the equation defining \(p_{21}\) is positive, and its l.h.s is continuous in \(p_{21}\), equals 0 when \(p_{21} = 0\), and approaches \(\infty\) when \(p_{21} \to \infty\) by our assumption that \(\lim_{x \to \infty} \frac{f(x)}{f(x+1)} = 1\). Since the processing times
are zero on any machine other than 1, for each $m$, $C^m_j(S)$ is independent of the sequence $S_m$ chosen by any machine other than 1. Furthermore, any optimal schedule has jobs 1, 2 positioned last. For such sequences, each machine’s objective value $\sum_j w_j^m f[C^m_j(S^1, S\ldots)]$ is $w_1^m f(1) + w_2^m f(1 + p_{21})$ for $S_1 = \ldots, 1, 2$ and $w_1^m f(1 + p_{21}) + w_2^m f(p_{21})$ for $S_1 = (\ldots, 2, 1)$. Since there exists an optimal permutation schedule by Proposition 1, it is sufficient to show that either some permutation schedule $(\ldots, 1, 2)$ is strictly better for the system than all permutation schedules $(\ldots, 2, 1)$ and machine 1 strictly prefers to deviate to some schedule $S_1 = (\ldots, 2, 1)$, or some permutation schedule $(\ldots, 2, 1)$ is strictly better for the system than all $(\ldots, 1, 2)$ and machine 1 strictly prefers to deviate to some schedule $S_1 = (\ldots, 1, 2)$. Equivalently, it is sufficient to establish the negativity of

$$\left( \sum_j w_j f[C_j((\ldots, 1, 2)] - \sum_j w_j f[C_j((\ldots, 2, 1)] \right) \left( \sum_j w_j^1 f[C^1_j((\ldots, 1, 2)] - \sum_j w_j^1 f[C^1_j((\ldots, 2, 1)] \right).$$

Substituting using the specification of $p_{21}$ and simplifying, this is equal to

$$[w_2(f(1 + p_{21}) - f(p_{21})) - w_1(f(1 + p_{21}) - f(1))]\left[w_2^1(f(1 + p_{21}) - f(p_{21})) - w_1^1(f(1 + p_{21}) - f(1))\right]$$

$$= [w_2 - w_1 \frac{f(1 + p_{21}) - f(1)}{f(1 + p_{21}) - f(p_{21})}]\left[w_2^1 - w_1^1 \frac{f(1 + p_{21}) - f(1)}{f(1 + p_{21}) - f(p_{21})}\right]\left[f(1 + p_{21}) - f(p_{21})\right]^2$$

$$= (w_2 - \frac{w_1}{2} \frac{w_2}{w_1} + \frac{w_2^1}{w_1^1})(w_2^1 - \frac{w_1}{2} \frac{w_2}{w_1} + \frac{w_2^1}{w_1^1})\left[f(1 + p_{21}) - f(p_{21})\right]^2$$

$$= -\frac{(w_1 w_2^1 - w_2 w_1^1)^2}{4w_1 w_1^1}\left[f(1 + p_{21}) - f(p_{21})\right]^2,$$

which is indeed negative since $w_1 w_2^1 \neq w_2 w_1^1$, $w_1 > 0$, $w_1^m > 0$ for all $j, m$, and $f$ is increasing. □

Theorems 2 and 3 show that the scenarios of local and global completion times are very different in that the latter does allow the possibility of coordination without requiring the information about the specific processing times in each instance. Intuitively, the global completion times scenario is closer to a centralized system, as it has the centralized completion time characteristics. Therefore, if the system manager may affect the scenario via an appropriate contract, it would be preferable to select the global completion times scenario.
4 Decentralized system under incomplete information

The analysis so far emphasized the role of the available information when considering incentive mechanisms. Theorem 3 established the possibility of coordination with minimal information in a decentralized system with global completion times. This conclusion was reached under the assumption that all parties have complete information concerning all the problem parameters. As explained in the introduction and in Section 2, it is reasonable to assume that the job weights are known, as they are given to the system by the end customers. Similarly, the job component weights for each machine are known as these are determined by the system as part of the job weighting mechanism. However, it may be argued that complete information also about all the remaining problem parameters is a too strong assumption in some cases. It may be reasonable to assume instead that each machine is informed about their own processing times, and only knows the distribution of processing times of other machines. This is the assumption we take in this section. In particular, the system manager only knows the distribution of processing times. We show that all the results obtained in previous sections continue to hold under this more realistic assumption.

We generalize the model of Section 2 in order to incorporate incomplete information about the processing times. Suppose there is a finite number, $n \geq 1$, of possible processing time scenarios/states, and state $1 \leq i \leq n$ has probability $q_i$, with $\sum_{i=1}^{n} q_i = 1$. At state $i$, the processing time of job $j$ on machine $m$ is $p_{jm}^i$. The probabilities $q_i$ are commonly known to all the machines and to the system manager. Additionally, at state $i$, each machine $m$ is informed only of the vector of their own processing times $p_m^i = (p_{jm}^i)_{\forall j}$. An instance of our COSG with incomplete information is therefore given by $[(p_{jm}^i)_{\forall i,j,m}, (q_i)_{\forall i}, (w_{jm}^m)_{\forall j,m,f}]$. Let $P_m = \{p_m | \exists i, p_m = p_{jm}^i\}$ be the set of all distinct such vectors for machine $m$. Machine $m$ may therefore choose their own job permutation as a function of their known processing time vector $p_m \in P_m$. We denote this permutation strategy by $\pi_m = (\pi_{mp_m})_{\forall p_m} = (\pi_{jmp_m})_{\forall j,p_m}$. A schedule profile $\overline{S} = (S_m)_{\forall m}$ specifies the strategy $\overline{S}_m$ for all machines.

Given a schedule profile $\overline{S}$, the realized schedule at state $i$ is $\overline{S}_m^i = (\overline{S}_{mp_m})_{\forall p_m}$. Consequently, the realized completion time of job $j$ in machine $m$ is $C_{jm}(\overline{S}_m^i) = \sum_{j' | \overline{S}_{jm'p_{jm'}} \leq \overline{S}_m^i} p_{jm'}$, and the realized system completion time of job $j$ is $C_j(\overline{S}_m^i) = \max_{m} C_{jm}(\overline{S}_m^i)$. The realized completion time of job $j$ from the perspective of machine $m$ is then $C_{jm}(\overline{S}_m^i) = C_{jm}(\overline{S}_m^i)$ under local completion times.
and \( C_j^m(S_i) = C_j(S_i) \) under global completion times.

The objective function of the system is to process according to the schedule profile \( S \) which minimizes the expected weighted sum of disutility of completion times, \( F(S) = \sum_i q_i \sum_j w_j f[C_j(S_i)] \), i.e., the schedule profile that solves \( \min_S F(S) \). Denote such a system optimal centralized schedule profile by \( S_O \). For a decentralized system with either local or global completion times, the objective of each machine \( m \) is to set their own part of the schedule profile, \( S_m \), so as to minimize their own expected weighted sum of disutility of completion times, \( F_m(S) = \sum_i q_i \sum_j w_j^m f[C^m_j(S_i)] \), taking as given the schedule profile parts of the other machines \( S_{-m} = (S_1, S_2, ..., S_{m-1}, S_{m+1}, ..., S_M) \), i.e., the schedule strategy that solves \( \min_{S_m} F_m(S) \). Therefore, due to the incomplete information, this leads to a (Bayesian) Nash equilibrium denoted by \( S^E \), where machine \( m \)'s part is denoted by \( S^E_m \).

Finally, denoting by \( S^B \) some equilibrium which minimizes the system’s disutility \( F(S^E) \) among all equilibria (i.e., best equilibrium), and by \( S^W \) some equilibrium which maximizes the system’s disutility among all equilibria (i.e., worst equilibrium), the DC is \( \frac{F(S^B)}{F(S^E)} \) and the PoA is \( \frac{F(S^W)}{F(S^E)} \).

The following example demonstrates our model in an environment with incomplete information.

**Example 3.** Consider two jobs with weights \( w_1 = \frac{1}{3} \) and \( w_2 = \frac{2}{3} \) to be processed on two machines. There are two states \( i = 1, 2 \), with probabilities \( q_1 = \frac{1}{3} \) and \( q_2 = \frac{2}{3} \). In state 1, the processing times for job 1 are \( p_{11} = 3, p_{12} = 4 \), and for job 2 are \( p_{21} = 1, p_{22} = 6 \), whereas in state 2 they are \( p_{11} = 5, p_{12} = 7, p_{21} = 2, p_{22} = 13 \). Thus at each state, each machine knows their own processing times but faces uncertainty about the processing times of the other machine. The job component weights for the machines are \( w_1^1 = \frac{1}{12}, w_1^2 = \frac{3}{12} \) for job 1 and \( w_2^1 = \frac{5}{12}, w_2^2 = \frac{3}{12} \) for job 2. The system’s and the machines’ objectives are to minimize the weighted sum of completion times, i.e. the disutility function is \( f(x) = x \). Each machine \( m \) has two possible permutations \( S_m \), either \( (1, 2) \) or \( (2, 1) \), and may choose their permutation as a function of their private information, namely differently at each state, as the processing times are different across the two states. Thus each machine \( m \) has 4 possible schedule strategies \( S_m \). The following tables present for state 1, for each possible realized schedule \( S^1 = (S_{1(3,4)}, S_{2(1,6)}) \), each machine’s realized objective value, \( F^1(S^1), F^2(S^1) \), and in parenthesis the system’s realized objective value, \( F(S^1) \); this is given for local completion times in left table, and for global completion times in the right table.

23
Similarly, the realized objective values at state 2 for each possible realized schedule $S^2 = (S_{1(5,7)}, S_{2(2,13)})$ are the following.

Finally, recall that the machines do not know the realized state, and may only choose depending on their private information about their own processing times. Therefore for each schedule profile $S = (S_1, S_2)$, where $S_m = \{S_{mp_1}, S_{mp_2}\}$ is the schedule strategy consisting of the permutation chosen by machine $m$ for states $i = 1, 2$, the following tables present each machine’s objective value, $F^i(S)$, $F^2(S)$, and in parenthesis the system’s objective value, $\overline{F}(S)$.

<table>
<thead>
<tr>
<th>local</th>
<th>(1,2)</th>
<th>(2,1)</th>
<th>global</th>
<th>(1,2)</th>
<th>(2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>1.92,3.50 (8.00)</td>
<td>1.92,4.00 (7.33)</td>
<td>(1,2)</td>
<td>4.50,3.50 (8.00)</td>
<td>3.33,4.00 (7.33)</td>
</tr>
<tr>
<td>(2,1)</td>
<td>0.75,3.50 (8.00)</td>
<td>0.75,4.00 (7.33)</td>
<td>(2,1)</td>
<td>4.50,3.50 (8.00)</td>
<td>3.33,4.00 (7.33)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>local</th>
<th>(1,2)</th>
<th>(2,1)</th>
<th>global</th>
<th>(1,2)</th>
<th>(2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>3.33,6.75 (15.7)</td>
<td>3.33,8.25 (15.3)</td>
<td>(1,2)</td>
<td>8.92,6.75 (15.7)</td>
<td>7.08,8.25 (15.3)</td>
</tr>
<tr>
<td>(2,1)</td>
<td>1.42,6.75 (15.7)</td>
<td>1.42,8.25 (15.3)</td>
<td>(2,1)</td>
<td>8.92,6.75 (15.7)</td>
<td>7.08,8.25 (15.3)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>local</th>
<th>(1,2)</th>
<th>(2,1)</th>
<th>global</th>
<th>(1,2)</th>
<th>(2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>2.86,5.67 (13.1)</td>
<td>2.86,6.67 (12.9)</td>
<td>(1,2)</td>
<td>7.44,5.67 (13.1)</td>
<td>6.22,6.67 (12.9)</td>
</tr>
<tr>
<td>(2,1)</td>
<td>1.58,5.67 (13.1)</td>
<td>1.58,6.67 (12.9)</td>
<td>(2,1)</td>
<td>7.44,5.67 (13.1)</td>
<td>6.22,6.67 (12.9)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>local</th>
<th>(1,2)</th>
<th>(2,1)</th>
<th>global</th>
<th>(1,2)</th>
<th>(2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>2.47,5.67 (13.1)</td>
<td>2.47,6.67 (12.9)</td>
<td>(1,2)</td>
<td>7.44,5.67 (13.1)</td>
<td>6.22,6.67 (12.9)</td>
</tr>
<tr>
<td>(2,1)</td>
<td>1.19,5.67 (13.1)</td>
<td>1.19,6.67 (12.9)</td>
<td>(2,1)</td>
<td>7.44,5.67 (13.1)</td>
<td>6.22,6.67 (12.9)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>global</th>
<th>(1,2)</th>
<th>(2,1)</th>
<th>(2,1)</th>
<th>(2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>7.44,5.67 (13.1)</td>
<td>6.22,6.67 (12.9)</td>
<td>7.06,5.83 (12.9)</td>
<td>5.83,6.83 (12.7)</td>
</tr>
<tr>
<td>(2,1)</td>
<td>7.44,5.67 (13.1)</td>
<td>6.22,6.67 (12.9)</td>
<td>7.06,5.83 (12.9)</td>
<td>5.83,6.83 (12.7)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>global</th>
<th>(1,2)</th>
<th>(2,1)</th>
<th>(2,1)</th>
<th>(2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>7.44,5.67 (13.1)</td>
<td>6.22,6.67 (12.9)</td>
<td>7.06,5.83 (12.9)</td>
<td>5.83,6.83 (12.7)</td>
</tr>
<tr>
<td>(2,1)</td>
<td>7.44,5.67 (13.1)</td>
<td>6.22,6.67 (12.9)</td>
<td>7.06,5.83 (12.9)</td>
<td>5.83,6.83 (12.7)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>local</th>
<th>(1,2)</th>
<th>(2,1)</th>
<th>global</th>
<th>(1,2)</th>
<th>(2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>3.33,6.75 (15.7)</td>
<td>3.33,8.25 (15.3)</td>
<td>(1,2)</td>
<td>8.92,6.75 (15.7)</td>
<td>7.08,8.25 (15.3)</td>
</tr>
<tr>
<td>(2,1)</td>
<td>1.42,6.75 (15.7)</td>
<td>1.42,8.25 (15.3)</td>
<td>(2,1)</td>
<td>8.92,6.75 (15.7)</td>
<td>7.08,8.25 (15.3)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>local</th>
<th>(1,2)</th>
<th>(2,1)</th>
<th>global</th>
<th>(1,2)</th>
<th>(2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>2.86,5.67 (13.1)</td>
<td>2.86,6.67 (12.9)</td>
<td>(1,2)</td>
<td>7.44,5.67 (13.1)</td>
<td>6.22,6.67 (12.9)</td>
</tr>
<tr>
<td>(2,1)</td>
<td>1.58,5.67 (13.1)</td>
<td>1.58,6.67 (12.9)</td>
<td>(2,1)</td>
<td>7.44,5.67 (13.1)</td>
<td>6.22,6.67 (12.9)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>local</th>
<th>(1,2)</th>
<th>(2,1)</th>
<th>global</th>
<th>(1,2)</th>
<th>(2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>2.47,5.67 (13.1)</td>
<td>2.47,6.67 (12.9)</td>
<td>(1,2)</td>
<td>7.44,5.67 (13.1)</td>
<td>6.22,6.67 (12.9)</td>
</tr>
<tr>
<td>(2,1)</td>
<td>1.19,5.67 (13.1)</td>
<td>1.19,6.67 (12.9)</td>
<td>(2,1)</td>
<td>7.44,5.67 (13.1)</td>
<td>6.22,6.67 (12.9)</td>
</tr>
</tbody>
</table>
For the local completion times scenario, both machine 1 and machine 2 have a strictly dominant strategy: for machine 1 it is setting the permutation (2, 1), and for machine 2 it is the permutation (1, 2), for both machines irrespective of their processing times. Thus there is a unique Nash equilibrium with this schedule profile (underlined). The system, however, would strictly prefer machine 2 to set the permutation (2, 1) irrespective of their processing times, with indifference to the decision of machine 1, thus leading to four optimal schedule profiles (marked in bold). For the global completion times scenario, machine 2 has the same strictly dominant strategy as in the local completion times scenario, and machine 1 is indifferent between their choices, consequently there are four cost-equivalent equilibria. The resulting DC and PoA in both scenarios are equal to $\frac{13}{127} \approx 1.0315$.

Example 3 has an optimal permutation schedule profile, namely $\overline{S}$ for which the permutation of jobs $\overline{S}_{m_{p_m}}$ is identical for each machine $m$ and each processing times $p_m$ realized for this machine (in Example 3 it is the permutation (2, 1)). One characteristic that may create potential complications in the analysis is the fact that the permutation schedule result of Proposition 1 does not extend to incomplete information, as shown next. The proof of Proposition 2 provides an example.

**Proposition 2.** Under incomplete information, an optimal permutation schedule profile may not always exist.

**Proof.** Consider two jobs with equal weights $w_1 = w_2 = \frac{1}{2}$, two machines, and two equally likely states, $q_1 = q_2 = \frac{1}{2}$. There is uncertainty only about the processing time of job 1 on machine 1, which is $p_{11}^1 = 1$ in state 1 and $p_{11}^2 = 3$ in state 2. The other processing times are $p_{21}^i = 2$ and $p_{12}^i = p_{22}^i = 1$ for all $i$. The disutility function is $f(x) = x$. For this example, the uniquely optimal schedule profile $\overline{S}^O$ has machine 1 always choosing according to an SPT order given their processing times, namely $\overline{S}^O_{1(1, 2)} = (1, 2)$ in state 1 and $\overline{S}^O_{1(3, 2)} = (2, 1)$ in state 2, and machine 2 choosing $\overline{S}^O_2 = (1, 2)$ as the strictly better compromise permutation given the uncertainty about the processing times of machine 1. The expected weighted sum of completion times under $\overline{S}^O$ is $\frac{1}{2}[\frac{1}{2} \cdot 1 + \frac{1}{2} (1 + 2)] + \frac{1}{2} [\frac{1}{2} \cdot (2 + 3) + \frac{1}{2} \cdot 2] = 2.75$, which is strictly lower than any other feasible outcome. However, $\overline{S}^O$ is not a permutation schedule profile. \qed

Despite Proposition 2, we are now ready to state our main conclusion, namely that all the
results in the paper extend to environments with incomplete information. Definition 5 generalizes as follows.

**Definition 7.** A job weighting mechanism is coordinating if both the DC and PoA are equal to 1 for all processing times \((p_{jm})_{v_{j,m}}\), probabilities \((q_i)_{v_i}\), job weights \((w_j)_{v_j}\) and disutility functions \(f\). A job weighting mechanism requires minimal information if for all job weights \((w_j)_{v_j}\), the allocated job component weights \((w^m_j)_{v_{j,m}}\) do not depend on the processing times \((p^i_{jm})_{v_{j,m}}\), probabilities \((q_i)_{v_i}\) or the disutility function \(f\).

**Theorem 4.** Theorems 1, 2 and 3 hold also for decentralized systems with incomplete information.

Theorem 4 may be understood as follows. The bound for DC and PoA relies on the fact that the system completion time of any job is at most the system makespan, and this property continues to hold under incomplete information. Since complete information can be seen as a special case of incomplete information, the bound remains tight. This special case argument is also the reason for the impossibility of general coordination under local completion times, and for the coordination with minimal information under global completion times to be possible only under a related machines mechanism. Finally, the cost sharing nature of any related machines mechanism allows the coordination result under incomplete information and global completion times.

**Proof.** Some parts of the extension to incomplete information follow directly from the same arguments as in the proofs of the relevant theorems, with the same instances used there but within a single state assigned probability 1, i.e. with complete information as a special case of incomplete information. These parts include the fact that the bound in Theorem 1 is tight, the whole argument of Theorem 2, and the fact established in 3 that a coordinating job weighting mechanism that requires minimal information must be a related machines mechanism.

It is therefore sufficient to show all other parts of these theorems, as they require a more elaborate proof. We start by considering the bound in Theorem 1. Fix any instance \([ (p^i_{jm})_{v_{i,j,m}}, (q_i)_{v_i}, (w^m_j)_{v_{j,m}}, f ] \) such that \( \sum_m w^m_j = w_j \) for all \( j \) for the given job weights \( w_j \). For any schedule profile \( S_i \), the realized system makespan at state \( i \) is

\[
M_i^i = \max_j C_j(S_i^i) = \max_m \max_j C_{jm}(S_i^i) = \max_m \sum_j p^i_{jm},
\]

\[26\]
thus is independent of $\mathcal{S}$. Consider an optimal schedule profile $\mathcal{S}^O$, and let $m^L_i$ be the machine that determines the realized system makespan according to $\mathcal{S}^O$ at state $i$, i.e. for which $\sum_j p^i_{jm^L_i} = M_i$.

For the last job on machine $m^L_i$ according to $\mathcal{S}^O$ at state $i$, i.e. the job $j^L_i$ for which $\mathcal{S}^O_j = J$, the completion time of $j^L_i$ in the system and on this machine are both equal to the realized system makespan, i.e. $C_{j^L_i}(\mathcal{S}^O, i) = C_{j^L_i m^L_i}(\mathcal{S}^O, i) = \sum_j p^i_{jm^L_i} = M_i$. Therefore the DC and PoA are bounded from above by

$$\frac{F(\mathcal{S}^B)}{F(\mathcal{S}^O)} = \frac{\sum_i q_i \sum_j w_j f[C_j(\mathcal{S}^B, i)]}{\sum_i q_i \sum_j w_j f[C_j(\mathcal{S}^O, i)]} \leq \frac{\sum_i q_i \sum_j w_j f[C_j(\mathcal{S}^W, i)]}{\sum_i q_i \sum_j w_j f[C_j(\mathcal{S}^O, i)]} \leq \frac{\sum_i q_i f(M_i)}{\min_j w_j} \leq \frac{\sum_i q_i f(M_i)}{\min_j w_j} = \frac{1}{\min_j w_j},$$

where the first inequality follows from the definitions of DC and PoA, the second inequality follows because the realized system completion time of any job is at most the realized system makespan and $f$ is increasing, the third inequality follows because the system completion time of any job is at least zero and $f(0) = 0$, and the fourth inequality follows because the weight of any job is at least the minimal weight of all jobs.

It remains to show, as stated in Theorem 3, that there is a range of coordinating job weighting mechanisms that include the related machines mechanisms. Fix any processing times $(p^i_{jm})_{i,j,m}$, state probabilities $(q_i)_{i,j,m}$, job weights $(w_j)_{i,j}$ and disutility functions $f$, and let $\mathcal{S}^O$ be a corresponding optimal permutation schedule. Consider a related machines mechanism with weights $w^m$ for each $m$, i.e., $w^m_j = w_j w^m$ for all $j,m$. Then for each $m$ and $\mathcal{S}_m$,

$$F^m(\mathcal{S}_m^O, \mathcal{S}_{-m}^O) = \sum_i q_i \sum_j w^m_j f[C^m_j(\mathcal{S}_m^O, \mathcal{S}_{-m}^O)] = \sum_i q_i \sum_j w^m_j f[C^m_j(\mathcal{S}_m^O, \mathcal{S}_{-m}^O)] \leq \sum_i q_i \sum_j w^m_j f[C^m_j(\mathcal{S}_m^O, \mathcal{S}_{-m}^O)] = F^m(\mathcal{S}_m, \mathcal{S}_{-m}^O),$$

where the inequality follows from optimality of $\mathcal{S}^O$. Note that it still holds that the cost objective value of machine $m$ is equal to the system cost objective value times $w^m$, namely the machine cost is proportional to the system’s in any schedule $S$, thus any non-optimal schedule for the system is also non-optimal for the machine. Therefore a schedule is optimal if and only
if it is an equilibrium, and both the DC and PoA are equal to 1. Furthermore, since fixing any schedule \( S \), \( \sum_i q_i \sum_j w_j^m f[C^m_j(S^i_m, \overline{S}^i_m)] \) is still continuous in each \( w_j^m \), the set of inequalities \( \sum_i q_i \sum_j w_j^m f[C^m_j(S^O,i_m, \overline{S}^O,i_m)] \leq \sum_i q_i \sum_j w_j^m f[C^m_j(S^i_m, \overline{S}^i_m)] \) for each \( m \) and \( \overline{S}^i_m \) remains valid for a range of job component weights \( w_j^m \) around \( w_j^m \) for all \( j, m \). Thus the DC and PoA remain equal to 1 for all job weighting mechanisms assigning job component weights in this range. \( \square \)

5 Concluding remarks

Our setting has specialized machines processing their portion of several jobs, where each job is finished only when all of the machines completed working on it. The system and machines use the same type of objective, namely minimizing the weighted sum of disutility of completion times. We studied the loss, expressed by the Decentralization Cost (DC) and the Price of Anarchy (PoA), of a decentralized system modeled as a non-cooperative game versus the centralized one. Tight bounds show that these inefficiency measures are relatively large in two scenarios of the decentralized system, namely when each machine only considers the completion time of their own part, and when each machine considers the system completion time. We also show that coordination with minimal information requirements is impossible for the first scenario, and is possible for the second scenario only with a particular class of scheduling based, job weighting mechanisms. All the results extend to an environment with incomplete information.

Additional research in this area could use different objective functions, possibly player-specific. Different types of contracts (e.g., revenue sharing) may also be considered.

References


