Coordination of the Decentralized Concurrent Open-Shop

Yossi Bukchin*  Eran Hanany*  Yigal Gerchak*

July 2021

Abstract

In concurrent open-shop, several jobs have to be completed, where each job consists of multiple components that are processed simultaneously by different dedicated machines. We assume that the components are sequenced on each machine in a decentralized manner, and analyze the resulting coordination problem under the objective of minimizing the weighted sum of disutility of completion times. The decentralized system is modeled as a non-cooperative game for two environments: (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. Tight bounds are provided on the inefficiency that might occur in the decentralized system, showing potentially severe efficiency loss in both environments. We propose and investigate scheduling based, coordinating job weighting mechanisms that use concise information, showing impossibility in the local completion times environment and possibility using the related weights mechanism in the global completion times environment. 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.

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

1 Introduction

Operations often involve jobs that are broken up into components and executed in parallel by dedicated machines, one component for each job and machine, 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 open-shop problem. The problem was motivated via assembly systems, 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.

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 setting each machine will act based on its own cost structure and incentives. The resulting 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 decentral-
ization, and accordingly propose managerial strategies for coordination. Specifically, we consider
a contract signed between the system manager and the machines based on the natural objective
of minimizing the weighted sum of disutility of completion times. According to the contract, the
system manager announces the weights of components, and each machine is paid a decreasing func-
tion of the weighted sum of disutility of completion times. Such contracting affects the incentives
of the machines. Note that 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.

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 may be to minimize
the weighted sum of completion times, and the contract between the plant and the suppliers may
specify the payment accordingly. 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 be paid based on possibly different 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, 2017).

In the second application, different dedicated teams perform maintenance operation for air-
planes, 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 airplane,
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 op-
eration should determine the airplane component weights for each team in order to measure and
incentivize their operation. A unified view is shown in Figure 1, namely the job weights are dic-
tated to the system by the customer, while possibly different component weights are dictated to
the machines by the system. The decision regarding the 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 environments 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 environments may arise in practice, depending on the contract signed between the system manager and each machine. Contracts based on the local completion times environment create a natural separation between the machines, as each machine is evaluated based on their own performance alone. A global completion times contract, which creates interdependence between the machines, 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, 2013).

For clarity of the presentation, our model is initially analyzed under the assumption of complete information, namely that all parameters are commonly known to all the machines and the system manager. Nevertheless, most of our findings do not require complete information. 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. 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 environments. 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 mechanisms. In such a mechanism, the system manager determines the 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 using concise information, in the sense that the component weights for each machine
will not depend on the specific processing times or disutility function. For the local completion times environment 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 weighting mechanisms that use concise information as consisting of only the related weights mechanism. In this mechanism, the job weights are identical for all machines up to multiplication by a constant of proportionality. A related weights mechanism can therefore be thought of as a cost sharing arrangement.

Finally, we extend the analysis to a setting 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 of the 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, Framinan et al. 2019 and Ahmadian et al. (2021).

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 already for three machines. Roemer (2006) has summed up the evolving literature stream of the concurrent open-shop and showed that the problem is strongly NP-Hard even for two machines with unit weights. When weights are associated with the jobs, Mastrolilli et al. (2010) provided approximation results including a primal dual 2-approximation algorithm.

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 Decentralization Cost (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. Atay et al. (2021) take a cooperative game theoretical approach to open shop scheduling problems to minimize the weighted sum of completion times, provide a core allocation rule for unit processing times and weights, and counterexamples for the existence of such allocations for general open shop problems. See also Selvarajah and Steiner (2009), Manoj
et al. (2012), Vairaktarakis (2013) and Hamers et al. (2019) for related decentralized scheduling problems.

To conclude, the literature on concurrent open-shop scheduling mostly concerns centralized systems. Almost all existing research on decentralized scheduling systems concentrates on variants of parallel machines (including identical, related and unrelated machines). The contribution of our paper is the analysis of a decentralized concurrent open-shop. Under complete and incomplete information, we provide tight bounds on the inefficiency that might occur without intervention, and propose scheduling based, coordinating job weighting mechanisms that use concise information.

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 uses concise information. Section 4 extends all the results to a general setting in which there is incomplete information about the processing times. Section 5 concludes. All proofs are collected in the Appendix.

2 Model under complete information

Our model of Concurrent Open-Shop Game (COSG) considers a system processing the jobs in $\mathcal{J} = \{1, \ldots, J\}$ for $J \geq 2$. Each job $j \in \mathcal{J}$ consists of components $m = 1, \ldots, M$ that are processed in parallel by machines $\mathcal{M} = \{1, \ldots, M\}$ for $M \geq 2$, 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 $S_m = (S_{jm})_j$ 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 $S = (S_m)_m$ 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 consider a contract signed between the system manager and the machines based on the natural objective of minimizing the weighted sum of disutility of completion times. According to the contract, the system manager announces the weights of components, and each machine is paid a decreasing function of the weighted sum of disutility of completion times. Such contracting affects the incentives of the machines. The contract also determines the completion times that enter into the objective functions of the machines. While the system manager always considers the system completion time of each job, $C_j(S)$, the job completion times from the perspective of the machines may be defined differently. Let $C_{jm}^m(S)$ 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 according to the contract. We consider the following two possible environments, as dictated by the contract.

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

Let $f : \mathbb{R}_+ \to \mathbb{R}_+$ 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 $m$, the component of job $j$ has a weight $w_j^m \in (0, 1)$, which represents the allocation
of the job’s importance across its components, where $\sum_{m} w_{jm} = w_{j}$. As explained above, it is reasonable to assume that the job weights $w_{j}$ are given to the system by the end customer, whereas the component weights $w_{jm}$ 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 summarize the elements of the model so far, for each decentralized environment, either with local or with global completion times, an instance of COSG is given by $[(p_{jm})_{j,m}, (w_{jm})_{j,m}, f]$. In this section and in Section 3 we assume that these model parameters are commonly known to all the machines and the system manager (this assumption is relaxed in Section 4).

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, $F^{m}(S) \equiv \sum_{j} w_{jm} f[C^{m}_{j}(S_{m}, S_{-m})]$, 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 environments.

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 $\frac{F(S^{W})}{F(S^{O})}$. Note that these inefficiency measures are justified since the same objective, the system’s, is used in both the numerator and denominator. 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 environments defined above, namely local vs. global completion times. The local completion times environment is the outcome of a natural contract, as each machine is evaluated according to their own performance, irrespective of the other machines. In contrast, the global completion times environment has the potential to be better aligned with the system’s objective. Such a contract 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 Aydinliyim, 2017). 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 environments the machine’s problem is clearly equivalent to the system’s problem. The following simple, 2-job, 2-machine example compares the environments 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 component weights for the machines are $w_1^1 = 0.51, w_2^2 = 0.09$ for job 1 and $w_1^1 = 0.30, w_2^2 = 0.10$ 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 the left table, and for global completion times in the right table. Note that in the global completion times environment, the system’s objective value is equal to the sum of the objectives values of the machines.
For the local completion times environment, both machine 1 and machine 2 have a strictly dominant strategy: for machine 1 it is to set the permutation (2, 1), and for machine 2 it is to set 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 3.20. The decentralized setting 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 environment, 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 environments, 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 environments to generate the exact opposite comparison. For instance, this would occur in this example if we just swapped the component weights of job 2 to $w_2^1 = 0.10, w_2^2 = 0.30$, with no further changes.
With this modification, the local completion times environment 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 \(\frac{46}{32} = 1.4375\) 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 environments. 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 component weights from the perspective of the machines. The 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 component weights are dictated to the machines by the system manager.

Therefore, the job weighting mechanism determines, for any job weights \((w_j)_j\), the component weights \(w^m_j\) of each job \(j\) from the perspective of each machine \(m\), where \(\sum_m w^m_j = w_j\) for all \(j\). One natural candidate for such a mechanism is the following.

**Definition 4.** Given an instance \([(p_{jm})_{j,m}, (w^m_j)_{j,m}, f]\), 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., \( w^m_j = \frac{p_{jm}}{\sum_m p_{jm}} w_j \).

One may view the proportional mechanism as a default arrangement according to which the machines are paid proportionally to the their processing times, thus the 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 environment, whereby the 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 component weights for the machines are determined by the proportional mechanism, namely \( w_1^1 = 0.125, w_2^1 = 0.375 \) for job 1 and \( w_1^2 = 0.375, w_2^2 = 0.125 \) 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.

| local | (1,2) | (2,1) | | global | (1,2) | (2,1) |
|-------|------|------| |       |------|------|
| (1,2) | 1.625,1.625 (3.5) | 1.625,1.625 (4.0) | | (1,2) | 1.875,1.625 (3.5) | 2.000,2.000 (4.0) |
| (2,1) | 1.625,1.625 (4.0) | 1.625,1.625 (3.5) | | (2,1) | 2.000,2.000 (4.0) | 1.625,1.875 (3.5) |

For the local completion times environment (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 environment (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 environments 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 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>1.833,1.167 (3.0)</td>
<td>2.000,1.500 (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 environment 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 $\frac{3.5}{3.0} \approx 1.1667$. 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. Intuitively, this follows because any non-permutation schedule can be weakly improved by postponing a job with higher system completion time on a machine where this job is located before another job with lower system completion time. For completeness of the presentation, we give here their result.

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

### 3 Inefficiency bounds and coordination under complete information

We begin our analysis by providing bounds for the DC and PoA that may occur in each of the environments 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 component
weights $w^m_j$ 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 $(w_j)_j$ and disutility functions $f$ this bound is tight, i.e. there exists an instance approaching this bound.

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 $\sum_j \frac{w_j}{\min_j w_j}$. 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 $f$.

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 component
weights \( w_j^m \) of each job \( j \) from the perspective of each machine \( m \), with \( \sum_m w_j^m = w_j \) 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 use concise information, 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 the DCIs equal to 1 for all processing times \((p_{jm})_{j,m}\), job weights \((w_j)_j\) and disutility functions \(f\). A job weighting mechanism uses concise information if for all job weights \((w_j)_j\), the allocated component weights \((w_j^m)_{j,m}\) do not depend on the processing times \((p_{jm})_{j,m}\) or the disutility function \(f\).

As shown in Example 2, the proportional mechanism fails to be coordinating. Moreover, by definition it also fails to use concise information because it directly depends on the processing times. We will therefore pay attention instead to the following simple job weighting mechanism that does use concise information. This mechanism assigns weights to machines and determines the component weights as a product of the job weight and the machine weight. As we will show, the mechanism will achieve the desired coordination under the global completion time environment.

**Definition 6.** Given jobs weights \( w_j \) for all \( j \), a job weighting mechanism \( w_j^m \) is said to be a related weights mechanism if, for some \( w^m \) for each \( m \), the component weights satisfy \( w_j^m = w_j w^m \) for all \( j,m \).

Note that any distribution of weights \( w^m \) for each \( m \) defines some related weights mechanism, and since \( \sum_j w_j = 1 \), for any related weights mechanism with weights \( w^m \) for each \( m \) we have \( \sum_m w^m = 1 \).

The related weights mechanism is coordinating when restricting attention to instances of the “related machines” type, i.e. when processing times are the same on all machines up to scaling (multiplication by a positive machine dependent constant). Intuitively, such instances create aligned incentives for the machine and the system under either local or global completion times. Unfortunately, this does not hold for general instances. Our first result on job weighting mechanisms concerns the impossibility of achieving the desired coordination under the local completion times environment.
**Theorem 2.** *In a decentralized system with local completion times, there does not exist a coordinating job weighting mechanism that uses concise 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 uses concise 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 weights mechanism.

Theorem 2 shows that for the case of local completion times, a job weighting mechanism that achieves efficiency for all instances necessarily uses 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 environment, 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 weights mechanism. Furthermore, a job weighting mechanism that uses concise information is coordinating if and only if it is a related weights mechanism.*

Intuitively, the first part of Theorem 3 follows since any related weights 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 job weighting mechanism that uses concise information and is not of the related weights type, for which there necessarily exist two jobs with relative job weights that are different from their relative 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 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 $f$.

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

4 Model and analysis 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 concise 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 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 a setting with incomplete information, in which 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.

In order to incorporate incomplete information about the processing times, we generalize the model of Section 2. Suppose there is an underlying stochastic process that determines the distribution of processing times. Specifically, there is a finite number, $n \geq 1$, of possible processing
time scenarios, and scenario $1 \leq i \leq n$ has probability $q_i$, with $\sum_{i=1}^{n} q_i = 1$. In scenario $i$, the processing time of job $j$ on machine $m$ is $p_{jm}^i$. This notation is without loss of generality, as it allows a machine $m$ to have equal processing times across scenarios. The probabilities $q_i$ are commonly known to all the machines and to the system manager. Additionally, in scenario $i$, each machine $m$ is informed only of the vector of their own processing times $p_m^i = (p_{jm}^i)_j$. The scenarios are therefore a standard way to model the uncertainty each machine faces about the processing times of other machines. An instance of our COSG with incomplete information is given by $[(p_{jm}^i)_{i,j,m}, (q_i), (w_{jm}^m)_{j,m,f}]$. Let $P_m = \{p_m \mid \exists i, p_m = p_{jm}^i\}$ be the set of all distinct processing time vectors for machine $m$. Given processing time vector $p_m \in P_m$, machine $m$ assigns conditional probabilities, $q(p_m')_{m' \neq m|p_m} = \frac{\sum_{i}(1|p_{jm}^i = p_{jm}^m, m') q_i}{\sum_{i}(1|p_{jm}^m = p_{jm}^m) q_i}$, to the processing times $(p_{jm'})_{m' \neq m}$ of the other machines. 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 $\overline{S}_m = (\overline{S}_{mpm})_{p_m} = (\overline{S}_{jmpm})_{j,p_m}$. A schedule profile $\overline{S} = (\overline{S}_m)_m$ specifies the strategy $\overline{S}_m$ for all machines.

Given a schedule profile $\overline{S}$, the realized schedule in scenario $i$ is $\overline{S}^i = (\overline{S}_{mpm})_{p_m}$. Consequently, the realized completion time of job $j$ in machine $m$ is $C_{jm}(\overline{S}^i) = \sum_j \{j|\overline{S}_{jmpm} \leq \overline{S}_{jmpm}^i\} p_{jm}^i$, and the realized system completion time of job $j$ is $C_j(\overline{S}^i) = \max_m C_{jm}(\overline{S}^i)$. The realized completion time of job $j$ from the perspective of machine $m$ is then $C_{jm}^m(\overline{S}^i) = C_{jm}(\overline{S}^i)$ under local completion times and $C_{jm}^m(\overline{S}^i) = C_j(\overline{S}^i)$ under global completion times.

The objective function of the system is to process according to the schedule profile $\overline{S}$ which minimizes the expected weighted sum of disutility of completion times, $\overline{F}(S) \equiv \sum_i q_i \sum_j w_j f[C_j(S^i)]$, i.e., the schedule profile that solves $\min_S \overline{F}(S)$. Denote such a system optimal centralized schedule profile by $\overline{S}^O$. 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, $\overline{S}_m$, so as to minimize their own conditional expected weighted sum of disutility of completion times according to the conditional probabilities $q(p_{mj})_{m' \neq m|p_m}$ given their knowledge of their own processing times $p_m$, and taking as given the schedule profile parts of the other machines. To simplify the presentation, we will use the equivalent ex-ante (without conditioning) objective for machine $m$, namely to minimize the expected weighted sum of disutility of completion times, $\overline{F}^m(S) \equiv \sum_i q_i \sum_j w_j^m f[C_j^m(\overline{S}^i)]$, taking as given the schedule profile parts of the other machines $\overline{S}_{-m} = (\overline{S}_1, \overline{S}_2, ..., \overline{S}_{m-1}, \overline{S}_{m+1}, ..., \overline{S}_M)$.
i.e., the schedule strategy that solves \( \min_{\mathcal{S}_m} F_1^m(\mathcal{S}) \). Therefore, due to the incomplete information, this leads to a (Bayesian) Nash equilibrium denoted by \( \mathcal{S}^E \), where machine \( m \)'s part is denoted by \( \mathcal{S}^E_m \).

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

The following example demonstrates our model in a setting 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 scenarios \( i=1,2 \), with probabilities \( q_1 = \frac{1}{3} \) and \( q_2 = \frac{2}{3} \). In scenario 1, the processing times for job 1 are \( p_{11}^1 = 3, p_{12}^1 = 4 \), and for job 2 are \( p_{21}^1 = 1, p_{22}^1 = 6 \), whereas in scenario 2 they are \( p_{11}^2 = 5, p_{12}^2 = 4, p_{21}^2 = 2, p_{22}^2 = 6 \). Thus in each scenario, each machine knows their own processing times, but machine 1 also knows the scenario independent processing times of machine 2, while machine 2 faces uncertainty about the processing times of machine 1 which do vary across the scenarios. The component weights for the machines are \( w_1^1_1 = \frac{1}{12}, w_1^2 = \frac{3}{12} \) 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 in each scenario, as the processing times are different across the two scenarios. Thus each machine \( m \) has 4 possible schedule strategies \( \mathcal{S}_m \).

The following tables present for scenario 1, for each possible realized schedule \( \mathcal{S}^1 = (\mathcal{S}^1_{(3,4)}, \mathcal{S}^1_{(2,1)}) \), each machine’s realized objective value, \( F^1(\mathcal{S}^1), F^2(\mathcal{S}^1) \), and in parenthesis the system’s realized objective value, \( F(\mathcal{S}^1) \); this is given for local completion times in left table, and for global completion times in the right 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>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>

Similarly, the realized objective values in scenario 2 for each possible realized schedule \( \mathcal{S}^2 = (\mathcal{S}^2_{(5,4)}, \mathcal{S}^2_{(2,6)}) \) are the following.

20
Finally, recall that the machines do not know the realized scenario, and may only choose depending on their private information about their own processing times. Therefore for each schedule profile $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$, where $\mathcal{S}_1 = \{\mathcal{S}_{1p_1}, \mathcal{S}_{1p_2}\}$ and $\mathcal{S}_2 = \{\mathcal{S}_{2p_2}\}$ are the schedule strategies consisting of the permutation chosen by machine $m = 1, 2$, respectively, for scenarios $i = 1, 2$, the following tables present each machine’s objective value, $F^m(\mathcal{S})$, and in parenthesis the system’s objective value, $F(\mathcal{S})$.

<table>
<thead>
<tr>
<th></th>
<th>(1,2)</th>
<th>(2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>3.33,3.50 (8.33)</td>
<td>3.33,4.00 (8.00)</td>
</tr>
<tr>
<td>(2,1)</td>
<td>1.42,3.50 (9.00)</td>
<td>1.42,4.00 (7.33)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(1,2)</th>
<th>(2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>4.58,3.75 (8.33)</td>
<td>3.75,4.25 (8.00)</td>
</tr>
<tr>
<td>(2,1)</td>
<td>4.75,4.25 (9.00)</td>
<td>3.33,4.00 (7.33)</td>
</tr>
</tbody>
</table>

For the local completion times environment, both machine 1 and machine 2 have a strictly dominant strategy: for machine 1 it is setting the permutation $(2, 1)$ irrespective of their processing times, and for machine 2 it is the permutation $(1, 2)$. 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 in scenario 1, thus leading to two optimal schedule profiles (marked in bold). The resulting DC and PoA are equal to $\frac{8.67}{7.33} \approx 1.1818$. For the global completion times environment, machine 2 has the same weakly dominant strategy as in the local completion times environment, to which machine 1 best responds strictly with the permutation $(1, 2)$ in scenario 2 and is indifferent between their choices in scenario 1, thus generating two equilibria; there are two additional strictly preferred equilibria for the system in which machine 1 sets the weakly dominated strategy, the permutation $(2, 1)$, to which machine 1 best responds strictly with the permutation $(2, 1)$ in scenario 2 and is again indifferent between their choices in scenario 1. The resulting DC is equal to 1 and the PoA is equal to $\frac{8.22}{7.33} \approx 1.1212$. 

21
Example 3 has an optimal permutation schedule profile, namely $\bar{S}$ for which the permutation of jobs $\bar{S}_{mp_m}$ is identical for each machine $m$ and each processing times $p_m$ realized for this machine (in Example 3 it is the permutation $(2,1)$). Proposition 1 guarantees the existence of a scenario specific optimal permutation schedule under complete information for each scenario separately. An optimal permutation schedule profile under incomplete information, as in Example 3, would exist whenever there exists a scenario specific optimal permutation schedule which is the same in all scenarios. 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.

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

**Definition 7.** A job weighting mechanism is coordinating if the DC is equal to 1 for all processing times $(p_{jm})_{j,m}$, probabilities $(q_i)_{i}$, job weights $(w_j)_{j}$ and disutility functions $f$. A job weighting mechanism uses concise information if for all job weights $(w_j)_{j}$, the allocated component weights $(w^m_{jm})_{j,m}$ do not depend on the processing times $(p_{jm})_{j,m}$, probabilities $(q_i)_{i}$ 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 identical 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 concise information under global completion times to be possible only under a related weights mechanism (as the special case was sufficient for these conclusions). Finally, the cost sharing nature of any related weights mechanism allows the coordination result under incomplete information and global completion times.
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 environments 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 use of concise information is impossible for the first environment, and is possible for the second environment only with a particular class of scheduling based, job weighting mechanisms. All the results extend to a setting 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.

A Appendix

Proof of Theorem 1. Fix any instance \([(p_{jm})_{j,m}, (w^m_j)_{j,m}, f]\) such that \(\sum_m w^m_j = w_j\) for all \(j\) for the given job weights \(w_j\). For any schedule \(S\), the system makespan is

\[
\mu \equiv \max_j C_j(S) = \max_m \max_j C_{jm}(S) = \max_m \sum_j p_{jm},
\]

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 \(\sum_j p_{jm^L} = \mu\).

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. \( C_j^L(S^O) = C_j^{\mu}(S^O) = \sum_j p_{jm\mu} = \mu \). Then,

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

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. Therefore the DC and PoA are bounded from above by \( \frac{1}{\min_j w_j} \).

To see that for all job weights \((w_j)\) 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 \( p_{J'1} = 1 \), \( p_{11} = \frac{w_J}{\varepsilon} - M \) and \( p_{m'm'} = p_{m'1} = 0 \), and set \( w_{J'}^1 = \lfloor 1 - \frac{f(p_{J'1})}{f(1 + p_{J'1})} \rfloor \), \( w_{J'}^1 = w_J - (M - 1) \varepsilon \), \( w_{m'}^m = \frac{w_J - w_{J'}^1}{M-1} \) and \( w_{m'}^m = \varepsilon \). In this case, \( C_{jm'}(S) = 0 \) for any \( j, S \) and \( m' > 1 \). Therefore \( C_j(S) = C_j^1(S) = C_{J1}(S) \) depends only on \( S_1 \). Additionally, any machine \( m' > 1 \) is indifferent between all permutations, either because \( C_{jm'}^m(S) = C_{jm'}^m(S) \) for local completion times, or because \( C_{jm'}^m(S) = C_j(S) \) for global completion times. For any \( 1 \leq k \leq J \), let \( S^k \) be the permutation where for all \( m, S_{km}^k = k \), 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 \( S_1 \) 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 \( 1 < k \leq J \),

\[
F^1(S^k) = \sum_j w_j^1 f[C_j^1(S^k)] = \sum_j w_j^1 f[C_{J1}(S^k)] = \sum_{j' = 1}^{k-1} w_j^1 f(j') + w_j^1 f(k-1+p_{J1}) + \sum_{j' = k}^{J-1} w_j^1 f(j'+p_{J1}) > w_j^1 f(p_{J1}) + \sum_{j' = 1}^{J-1} w_j^1 f(j'+p_{J1}) = \sum_j w_j^1 f[C_j^1(S^1)] = F^1(S^1)
\]
if and only if
\[ \sum_{j'=1}^{k-1} \frac{f(j') - f(j' + p_{J1})}{f(1 + p_{J1})} + w_j f(k - 1 + p_{J1}) - f(p_{J1}) \frac{w_j f(1 + p_{J1})}{w_j f(p_{J1})} > 0. \tag{A.1} \]

Since \( k \geq 2 \) and \( f(x) \) is non-negative and increasing in \( x \), the left-hand side of (A.1) is bounded from below by
\[ -\sum_{j'=1}^{k-1} \prod_{i=1}^{j'-1} \frac{f(i+1+p_{J1})}{f(i+p_{J1})} + \frac{w_j}{1 - f(1+p_{J1})}, \]
which approaches \(+\infty\) when \( \varepsilon \to 0 \) because \( \lim_{\varepsilon \to 0} w_j^1 = w_J > 0 \), \( \lim_{\varepsilon \to 0} p_{J1} = +\infty \) and \( \lim_{x \to +\infty} \frac{f(x)}{f(x+1)} = 1 \). Therefore (A.1) holds for sufficiently small \( \varepsilon > 0 \). 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
\[ \sum_{j'=1}^{k-1} w_j f(j') - \frac{f(j')}{f(J-1+p_{J1})} + \frac{w_J f(k - 1 + p_{J1})}{f(J-1+p_{J1})} + \sum_{j'=k}^{J-1} w_j f(j' + p_{J1}) - \frac{f(j' + p_{J1})}{f(J-1+p_{J1})} > 1. \tag{A.2} \]

The left-hand side of (A.2) approaches \( 1 + \frac{1}{w_J} \sum_{j'=k}^{J-1} w_j > 1 \) when \( \varepsilon \to 0 \) because, as above, \( \lim_{\varepsilon \to 0} p_{J1} = +\infty \) and \( \lim_{x \to +\infty} \frac{f(x)}{f(x+1)} = 1 \). Therefore inequality (A.2) holds for sufficiently small \( \varepsilon > 0 \). This leads to the DC value of
\[ \frac{w_J f(p_{J1}) + \sum_{j'=1}^{J-1} w_j f(j' + p_{J1})}{\sum_{j'=1}^{J-1} w_j f(j') + w_J f(J - 1 + p_{J1})}, \]
which approaches \( \frac{1}{\min_j w_j} \) as \( \varepsilon \to 0 \).

\( \square \)

**Proof of Theorem 2.** It is sufficient to show that all job weighting mechanisms that use concise information fail to generate DC equal to 1 for some instance with given job weights. We show this failure for an instance in which two jobs, say jobs \( J - 1 \) and \( J \), have equal job weights, i.e. \( w_{J-1} = w_J \). In fact, we show this holds for any disutility function \( f \). Consider the following two mutually exclusive and exhaustive cases: (i) any related weights mechanism, i.e. \( w_j^m = w_j w^m \) for
all \( j, m \), and (ii) any job weighting mechanism that is not of the related weights type. Begin with case (i). Consider any disutility function \( f \), and for all \( j' < J - 1 \) 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 \( j' < J - 1 \) and \( w_{J-1}^1 f(p_{(J-1)1}) + w_{J1}^1 f(p_{(J-1)1} + p_{J1}) = w_{J1}^1 [f(1) + f(3)] < w_{J1}^2 [f(1) + f(3)] = w_{J-1}^1 f(p_{(J-1)1} + p_{J1}) + w_{J1}^1 f(p_{J1}) \), any permutation \( S_1 = (..., J - 1, J) \) is weakly dominant for machine 1 (these permutations are strategically equivalent) and all other permutations are strictly dominated. Similarly, any permutation \( S_2 = (..., J, J - 1) \) is weakly dominant for machine 2 and all other permutations are strictly dominated. Therefore any equilibrium schedule, including the best \( S^B \), has \( S_1, S_2 \) with \( F(S^B) = w_{J-1} f(\max\{1,1 + 2\}) + w_{J} f(\max\{1 + 2, 1\}) = 2 w_{J} f(3) \). Additionally, any permutation schedules \((..., J - 1, J)\) and \((..., J, J - 1)\) are optimal with \( F(S^O) = w_{J-1} f(\max\{1,2\}) + w_{J} f(\max\{1 + 2, 2 + 1\}) = w_{J} [f(2) + f(3)] \). Thus \( \text{PoA} = \text{DC} = \frac{2 f(3)}{f(1) + f(3)} > 1 \).

Now consider case (ii). Since \( w_{J-1} = w_{J} \), and this case considers any job weighting mechanism other than the related weights, 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 \( j' < J - 1 \) and \( m' > 2 \), set \( p_{j'1} = p_{j'2} = 0 \), \( p_{(J-1)1} = p_{J1} = p_{(J-1)2} = p_{J2} = 1 \) and \( p_{j'm'} = p_{(J-1)m'} = p_{Jm'} = 0 \). In this case, as in case (i), 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 \( p_{j'1} = p_{j'2} = 0 \) for all \( j' < J - 1 \), \( w_{J-1}^1 [f(1) + w_{J}^1 f(2)] < w_{J-1}^1 f(2) + w_{J}^1 f(1) \) 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 and all other permutations are strictly dominated. Therefore any equilibrium schedule, including the best \( S^B \), has \( S_1, S_2 \) with \( F(S^B) = w_{J-1} f(\max\{1,1 + 1\}) + w_{J} f(\max\{1 + 1, 1\}) = 2 w_{J} f(2) \). 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 \( \text{PoA} = \text{DC} = \frac{2 f(2)}{f(1) + f(2)} > 1 \).

**Proof of Theorem 3.** Fix any processing times \((p_{jm})_{j,m}\), job weights \((w_j)\), and disutility functions \( f \), and let \( S^O \) be a corresponding optimal schedule. Consider a related weights 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 \(S_m\),

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

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 any optimal schedule is also an equilibrium, and the DC is 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 \(\sum_j w^m_j f[C^m_j(S_m, S^O_{-m})] \leq \sum_j w^m_j f[C^m_j(S_m, S^O_{-m})]\) for each \(m\) and \(S_m\) remains valid
for a range of component weights \(w^m_j\) around \(w_j w^m\) for all \(j, m\). Thus the DC remains equal to 1
for all job weighting mechanisms assigning component weights in this range.

It remains to show that for all job weighting mechanisms that use concise information and all
job weights \((w_j)_j\), if the DC is equal to 1 for all processing times \((p_{jm})_{j,m}\) and disutility functions
\(f\), then necessarily \(w^m_j = w_j w^m\) for all \(j, m\). Fix job weights \((w_j)_j\), and suppose that \(w^m_j\) is not
given by a related weights mechanism, so without loss of generality \(\frac{w^4_j}{w_1} \neq \frac{w^4_j}{w_2}\). Consider an instance
defined with any disutility function \(f\), and with \(p_{11} = 1\), \(p_{21}\) satisfying 
\(\frac{f(1+p_{21})-f(1)}{f(1+p_{21})-f(p_{21})} = \frac{1}{2} (\frac{w^4_2}{w_1} + \frac{w^4_1}{w_2})\),
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 \(\sum_j w^m_j f[C^m_j(S_1, S_{-1})]\) is \(w^m_1 f(1) + w^m_2 f(1+p_{21})\) for
\(S_1 = \ldots 1, 2\) and \(w^m_1 f(1+p_{21}) + w^m_2 f(p_{21})\) for \(S_1 = \ldots 2, 1\). It is sufficient to show that either some
schedule with \(S_1 = \ldots 1, 2\) is strictly better for the system than all schedules with \(S_1 = \ldots 2, 1\)
and machine 1 strictly prefers to deviate to some schedule \(S_1 = \ldots 2, 1\), or some schedule with
\(S_1 = \ldots 2, 1\) is strictly better for the system than all schedules with \(S_1 = \ldots 1, 2\) and machine 1
strictly prefers to deviate to some schedule \(S_1 = \ldots 1, 2\). Equivalently, it is sufficient to establish
the negativity of
\[
(\sum_j w_j f[C_j(\ldots, 1, 2)] - \sum_j w_j f[C_j(\ldots, 2, 1)]) (\sum_j w_j f[C_j(\ldots, 1, 2)] - \sum_j w_j f[C_j(\ldots, 2, 1)]).
\]

Substituting using the specification of \(p_{21}\) and simplifying, this is equal to
\[
[w_2(f(1 + p_{21}) - f(p_{21})) - w_1(f(1 + p_{21}) - f(1))] [w_2^1(f(1 + p_{21}) - f(p_{21})) - w_1^1(f(1 + p_{21}) - f(1))] \\
= (w_2 - w_1) \frac{f(1 + p_{21}) - f(1)}{f(1 + p_{21}) - f(p_{21})} [w_2^1 - w_1^1] \frac{f(1 + p_{21}) - f(1)}{f(1 + p_{21}) - f(p_{21})} [f(1 + p_{21}) - f(p_{21})]^2 \\
= - \frac{(w_1 w_2^1 - w_2 w_1^1)^2}{4 w_1^1 w_2} \cdot [f(1 + p_{21}) - f(p_{21})]^2,
\]
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. \(\square\)

**Proof of Proposition 2.** Consider two jobs with equal weights \(w_1 = w_2 = \frac{1}{2}\), two machines, and two equally likely scenarios, \(q_1 = q_2 = \frac{1}{2}\). There is uncertainty only about the processing time of job 1 on machine 1, which is \(p_{11}^1 = 1\) in scenario 1 and \(p_{11}^2 = 3\) in scenario 2. The other processing times are \(p_{21}^1 = 2\) and \(p_{11}^2 = p_{22}^1 = 1\) 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 \(\overline{S}_{1(1,2)}^O = (1, 2)\) in scenario 1 and \(\overline{S}_{1(3,2)}^O = (2, 1)\) in scenario 2, and machine 2 choosing \(\overline{S}_2 = (1, 2)\) 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. \(\square\)

**Proof of Theorem 4.** 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 scenario 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 uses concise information must be a related weights 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 \([(p_{jm}^i)_{i,j,m}, (q_i)_{i}, (w_m^j)_{j,m}, f]\) such that \(\sum_m w_m^j = w_j\) for all \(j\) for the given job weights \(w_j\). For any schedule profile \(S\), the realized system makespan in scenario \(i\) is

\[
\mu_i \equiv \max_j C_j(S^i) = \max_m \max_j C_{jm}(S^i) = \max_m \sum_j p_{jm}^i,
\]

thus is independent of \(S\). Consider an optimal schedule profile \(S^O\), and let \(m^L_i\) be the machine that determines the realized system makespan according to \(S^O\) in scenario \(i\), i.e. for which \(\sum_j p_{jm^L_i}^i = \mu_i\). For the last job on machine \(m^L_i\) according to \(S^O\) in scenario \(i\), i.e. the job \(j^L_i\) for which \(S^O,i(j^L_i)(m^L_i)(p_{m^L_i}^j) = J\), the completion time of \(j^L_i\) in the system and on this machine are both equal to the realized system makespan, i.e. \(C_{j^L_i}(S^O,i) = C_{j^L_i m^L_i}(S^O,i) = \sum_j p_{jm^L_i}^i = \mu_i\). Then,

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

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. Therefore the DC and PoA are bounded from above by \(\frac{1}{\min_j w_j}\).

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

\[
F^m(\bar{S}_m, \bar{S}_m - m) = \sum_i q_i \sum_j w^m_j f[C^m_j(\bar{S}_m, \bar{S}_m - m)] = w^m \sum_i q_i \sum_j w_j f[C_j(\bar{S}_m^O, \bar{S}_m - m)]
\]

\[
\leq w^m \sum_i q_i \sum_j w_j f[C_j(\bar{S}_m^i, \bar{S}_m - m)] = \sum_i q_i \sum_j w^m_j f[C^m_j(\bar{S}_m^i, \bar{S}_m - m)] = F^m(\bar{S}_m, \bar{S}_m - m),
\]

where the inequality follows from optimality of \( \bar{S}^O \). 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 any optimal schedule is also an equilibrium, and the DC is equal to 1. Furthermore, since fixing any schedule \( S \), \( \sum_i q_i \sum_j w^m_j f[C^m_j(\bar{S}_m, \bar{S}_m - m)] \) is still continuous in each \( w^m_j \), the set of inequalities \( \sum_i q_i \sum_j w^m_j f[C^m_j(\bar{S}_m^O, \bar{S}_m - m)] \leq \sum_i q_i \sum_j w^m_j f[C^m_j(\bar{S}_m^i, \bar{S}_m - m)] \) for each \( m \) and \( \bar{S}_m \) remains valid for a range of component weights \( w^m_j \) around \( w_j w^m \) for all \( j, m \). Thus the DC remains equal to 1 for all job weighting mechanisms assigning component weights in this range.

\[
\quad
\]

**References**


