Coordination Mechanisms, Cost-Sharing, and Approximation Algorithms for Scheduling

Ioannis Caragiannis[1], Vasilis Gkatzelis[2], and Cosimo Vinci[3]

[1] University of Patras, Rion-Patras, Greece
caragian@ceid.upatras.gr
[2] Drexel University, Philadelphia, PA, USA
gkatz@drexel.edu
[3] Gran Sasso Science Institute, L'Aquila, Italy
cosimo.vinci@gssi.it

Abstract. We reveal a connection between coordination mechanisms for unrelated machine scheduling and cost-sharing protocols. Using this connection, we interpret three coordination mechanisms from the recent literature as Shapley-value-based cost-sharing protocols, thus providing a unifying justification regarding why these mechanisms induce potential games. More importantly, this connection provides a template for designing novel coordination mechanisms, as well as approximation algorithms for the underlying optimization problem. The designer need only decide the total cost to be suffered on each machine, and then the Shapley value can be used to induce games guaranteed to possess a potential function; these games can, in turn, be used to design algorithms. To verify the power of this approach, we design a combinatorial algorithm that achieves an approximation guarantee of 1.81 for the problem of minimizing the total weighted completion time for unrelated machines. To the best of our knowledge, this is the best approximation guarantee among combinatorial polynomial-time algorithms for this problem.

1 Introduction

Since the 1950s, the study of scheduling has played a central role in both operations research and computer science. Machine scheduling models have provided a very useful abstraction that has enabled researchers to devise solutions with a wide range of applications. Depending on the context, the "machine" that the schedule is applied to can range from an airport runway serving multiple airplanes, and a classroom used for several courses, to a CPU that needs to process a set of jobs. In all of these examples, the ultimate goal is the efficient utilization of scarce resources. Most of the initial work on machine scheduling focused on designing algorithms that yield efficient schedules, where efficiency is quantified using a measure such as the makespan or the total weighted completion time of the jobs (e.g., see [25]). In the last two decades, motivated by the prevalence of large decentralized environments where the scheduler may have limited information or limited power in enforcing the schedule, many of these machine scheduling problems have been revisited from a game-theoretic point of view.

In a well-studied example of such a problem, each user controls a job that may require different processing times on each machine. These users are self-interested and they can decide which machine their job is assigned to, aiming to minimize its completion time. Each machine, however, is equipped with a decentralized scheduler, a *coordination mechanism* [12], that decides how the jobs assigned to that machine will be scheduled. Given the strategic nature of the users (which we henceforth call players since they are engaged in a game), these policies affect (and, in a sense, coordinate) their behavior. Unlike scheduling algorithms, which have full control over the outcome and can directly output efficient schedules, coordination mechanisms need to provide the appropriate incentives such that the game induced among the users leads to efficient schedules in equilibrium. Therefore, a highly desired property for a coordination mechanism is to induce potential games: these games guarantee the existence of pure Nash equilibria as well as convergence to them after finite sequences of steps. The *price of anarchy* measure can then be used in order to evaluate how inefficient the equilibria of these games may be, compared to the most efficient schedule.

The design of coordination mechanisms has often borrowed results from the literature on scheduling algorithms. For instance, when the efficiency is measured using the total weighted completion time objective, it is known that the best way to schedule the jobs assigned to each machine is based on the Smith's rule policy [33]. This scheduling policy is therefore a natural first candidate to use when designing a coordination mechanism. But, it turns out that the resulting coordination mechanism may induce games that have no pure Nash equilibria [16]. Furthermore, the price of anarchy of this mechanism is 4 [15], which falls short compared to other mechanisms in this setting. Specifically, Cole et al. [15] propose two alternative coordination mechanisms, ProportionalSharing and Rand (defined in detail later), and they show that their price of anarchy is 2.618 and at most 2.133, respectively. Furthermore, both of these mechanisms induce potential games. As a result, the optimal coordination mechanisms need not be similar to the best scheduling algorithms.

A much more surprising fact is that ideas from coordination mechanisms can actually help in designing new scheduling algorithms! In particular, if one can compute, or at least approximate, an equilibrium of a mechanism with good price of anarchy, then this implies a good approximation algorithm. Using this approach, Cole et al. [15] obtained a $2+\epsilon$-approximation algorithm for minimizing the total weighted completion time. They first defined a novel coordination mechanism, Approx, which induces potential games, and then designed a local-search algorithm that computes an assignment of jobs to machines by mimicking the best-response dynamics of the players in the game induced by Approx. Once the desired assignment of jobs to machines was reached, the algorithm then used the Smith's rule policy to schedule the jobs within each machine. What is very interesting about this result is that it uses the game-theoretic analysis to design an, otherwise counter-intuitive, algorithm with appealing performance guarantees. Can the ideas behind the design of Approx be generalized to a template that yields even more efficient combinatorial approximation algorithms?

1.1 Related Work

Since the definition of coordination mechanisms [12], a long list of papers has mostly focused on the design of policies in machine scheduling settings, aiming to minimize the makespan (e.g., [22, 5, 8, 2, 23, 7, 9]) or the total (weighted) completion time (e.g., [16, 15, 20, 1]). Apart from machine scheduling settings, coordination mechanisms have also been proposed for congestion games [14].

A literature that is very closely related to coordination mechanisms aims to design cost-sharing protocols that yield efficient equilibria (e.g., [29, 3, 28, 11, 17, 27, 19, 24, 18, 13]). Two of the most well-known cost-sharing protocols are the *marginal contribution* and the *Shapley value* [31].

The problem of scheduling jobs on unrelated machines aiming to minimize the total weighted completion time has been studied extensively in the machine scheduling literature (for a detailed list of some of the classic results see [25, Chapter 11]). For instances in the specific machine model of *unrelated* machines that we consider in this paper (formally defined in the following section), until very recently, the best approximation guarantee was a factor or 1.5, obtained using a convex quadratic relaxation of the problem [30, 32]. This has now been marginally improved by Bansal et al. [6] and Li [26] who showed that an approximation factor better than 1.5 is possible: their algorithms achieve a $(1.5 - c)$–approximation for an insignificantly small, yet positive constant c.

1.2 Our Results

In this paper, we study the interplay between coordination mechanisms and scheduling algorithms in more depth, and we also uncover interesting connections between coordination mechanisms and cost-sharing policies. As our first conceptual contribution, we observe (in Section 3) that the three coordination mechanisms ProportionalSharing, Rand, and Approx all follow a common recipe: these mechanisms can be defined as Shapley-value cost-sharing protocols. The Shapley-value is known to induce potential games, hence, the existence of a potential function follows immediately. This suggests a reverse engineering approach in the design of coordination mechanisms: all the designer has to do is to define an appropriate function that maps sets of jobs within each machine to a total cost. Then, the Shapley-value cost-sharing method is used to divide the total cost within each machine to the jobs; the cost charged to each job is then translated into a (weighted) completion time of the particular job.

In the above recipe, there is a feasibility constraint that has to be satisfied for the induced coordination mechanisms to be well-defined: there has to exist a feasible schedule of the jobs such that their completion times yields the desired costs. This is not necessary, however, when designing combinatorial approximation algorithms for the underlying optimization problem. It suffices to define appropriate total cost function and, as in [15], mimic the strategic behavior of (hypothetical) players that experience the Shapley-value share as cost. The last step of using the Smith's rule within each machine to compute the final schedule fixes possible feasibility issues. This template is presented in detail in Section 4.

Our main technical contribution is the analysis of a class of combinatorial approximation algorithms that follow the template above. The best among them achieves 1.81-approximate schedules with respect to the total weighted completion time objective. This improves the algorithmic result of Cole et al. [15] and, to the best of our knowledge, is the best worst-case approximation guarantee by a combinatorial polynomial-time algorithm for the problem. Better results are possible for particular input instances. We complement these results by showing (in Section 5) that none among the three coordination mechanisms ProportionalSharing, Rand, and Approx can be used to obtain approximation guarantees better than 2 using this template. Furthermore, we use this last result to obtain a tight lower bound of 4 on the price of anarchy of Approx.

Section 2 is devoted to preliminary definitions and notation. We conclude in Section 6. Due to lack of space, all proofs have been omitted; they will appear in the final version of the paper.

2 Preliminaries

We consider machine scheduling instances that consist of a set I of m machines and a set J of n jobs. We denote by p_{ij} the processing time of job j on machine i and by w_j the weight of job j. The *Smith ratio* ρ_{ij} is defined as the ratio p_{ij}/w_j. An *assignment* $\boldsymbol{\sigma}$ is a function that assigns each job j to a machine σ_j that processes the job. The standard notation $(\boldsymbol{\sigma}_{-j}, i)$ is used to denote the assignment in which job j is assigned to machine i while the remaining jobs use the same machine they use in $\boldsymbol{\sigma}$. We denote by $J_i(\boldsymbol{\sigma}) := \{j \in J : \sigma_j = i\}$ the set of jobs assigned by $\boldsymbol{\sigma}$ to machine i. In general, we focus on the *unrelated machines* setting, where the processing times p_{ij} can be arbitrary. In the *related machines* setting, each machine i has a speed $s_i > 0$, each job j has a processing requirement p_j, and the processing times are defined as $p_{ij} = p_j/s_i$.

A *coordination mechanism* comprises a set of local scheduling policies, one for each machine. A *scheduling policy* for a machine i determines the schedule of the jobs assigned to machine i by $\boldsymbol{\sigma}$. The scheduling policy for machine i is *local* if it does not depend on the set of jobs assigned to other machines. A coordination mechanism $\mathcal{A}$ simply defines the completion time $c_j^{\mathcal{A}}(\boldsymbol{\sigma})$ of job j on machine $\boldsymbol{\sigma}_j$. As scheduling policies are local, $c_j^{\mathcal{A}}$ can be equivalently defined as a function of machine σ_j and the set of jobs $J_{\sigma_j}(\boldsymbol{\sigma})$ only. The quantity $C^{\mathcal{A}}(\boldsymbol{\sigma}) := \sum_{j \in J} w_j c_j^{\mathcal{A}}(\boldsymbol{\sigma})$ is the *total weighted completion time* (or *total weighted cost*) of assignment $\boldsymbol{\sigma}$. We use the notation $C_i^{\mathcal{A}}(\boldsymbol{\sigma}) := \sum_{j \in J_i(\boldsymbol{\sigma})} w_j c_j^{\mathcal{A}}(\boldsymbol{\sigma})$ to refer to the total weighted completion time of the jobs assigned to machine i, i.e., $C^{\mathcal{A}}(\boldsymbol{\sigma}) = \sum_{i \in I} C_i^{\mathcal{A}}(\boldsymbol{\sigma})$.

For instance, the SmithRule scheduling policy schedules the jobs assigned to machine i in a non-decreasing order of their Smith ratios ρ_{ij}; for resolving ties between pairs of jobs j and k with the same Smith's ratio, a common tie-breaking rule is used in all machines. We use the notation $\rho_{ij} \prec \rho_{ik}$ to denote the fact that either $\rho_{ij} < \rho_{ik}$, or $\rho_{ij} = \rho_{ik}$ but j gets higher priority by the tie-breaking rule. A coordination mechanism can be randomized; in this case, the scheduling

policy within each machine is randomized, i.e., a probability distribution over deterministic policies.

Assuming that each job is controlled by a self-interested player who decides the machine on which her job will be assigned, a coordination mechanism naturally defines a strategic game among the players. Each player has any machine as possible strategy and her cost is simply the (expected) weighted completion time of her job as defined by the coordination mechanism. Hence, in an assignment $\boldsymbol{\sigma}$, the cost of player j when the coordination mechanism $\mathcal{A}$ is used is simply $w_j c_j^{\mathcal{A}}(\boldsymbol{\sigma})$. Then, the assignment $\boldsymbol{\sigma}$ is a *pure Nash equilibrium* if no player has any incentive to deviate from her strategy in $\boldsymbol{\sigma}$.

A desirable property from a coordination mechanism is to define *potential games*. This means that there exists a potential function $\Phi^{\mathcal{A}}(\cdot)$ that is defined over assignments with the property that for every pair of assignment $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma'}$ that differ only in the strategy of a single player j, it holds $\Phi^{\mathcal{A}}(\boldsymbol{\sigma}) - \Phi^{\mathcal{A}}(\boldsymbol{\sigma'}) = w_j c_j^{\mathcal{A}}(\boldsymbol{\sigma}) - w_j c_j^{\mathcal{A}}(\boldsymbol{\sigma'})$. The existence of a potential function implies that a pure Nash equilibrium not only exists but can also be found after a finite sequence of improving deviations (e.g., best-response deviations) by the players.

The total weighted completion time is a natural *social cost* measure that can be used to assess the quality of equilibria. The price of anarchy of a scheduling instance is then defined as the worst-case ratio of $C^{\mathcal{A}}(\boldsymbol{\sigma})/\mathrm{OPT}$, where σ is a pure Nash equilibrium of the game induced by mechanism $\mathcal{A}$ on instance M and OPT denotes the minimum value of total weighted completion time over all possible assignments and schedules (i.e., including schedules that are not produced by $\mathcal{A}$) of jobs to machines.

Beyond machine scheduling settings, *cost-sharing protocols* define ways in which a total cost is to be shared among a set N of agents who are competing for a collection of resources. Each resource i is characterized by a cost function $C_i : 2^N \to \mathbb{R}_+$, which quantifies the total cost that this resource would suffer, depending on the subset of the agents that end up using it. For the system to support itself, the agents using a resource need to contribute some amount such that their total contributions cover the total cost that they cause. The decision regarding how the cost f_j that each agent j needs to contributed is defined by a cost-sharing protocol. If the cost-shares that this protocol charges to the agents $S \subseteq N$ using a resource i always add up to exactly the total cost $C_i(S)$, then the cost-sharing protocol is called *budget-balanced*. Under some circumstances, the cost-sharing protocol may also charge the agents more than the total cost that they generated, which is often referred to as *over-charging*.

Two of the best-known cost-sharing protocols are the *marginal contribution* and the *Shapley value* [31]. According to the marginal contribution, the cost-share of each agent $j \in S$ is equal to $C_i(S) - C_i(S \setminus \{j\})$, i.e., equal to the increase in the total cost due to the presence of j. As long as the cost functions, $C_i(\cdot)$ are supermodular, these cost-shares cover the total cost, but they may be overcharging the agents. On the other hand, the Shapley value is budget-balanced and the cost-share of each agent j is equal to the expected value of the following process: order the agents of S uniformly at random, let $S_{<j}$ be the

subset of these agents that lie before j in the ordering, and charge j a cost of $C_i(S_{<j} \cup \{j\}) - C_i(S_{<j})$. In other words, the agents arrive in that random order and each one of them is charged for her marginal contribution with respect to the agents that have arrived up to that point.

We remark that cost-sharing protocols can be naturally defined on machine scheduling instances with each machine corresponding to a resource and with each job corresponding to an agent. This is exactly the analogy we consider in the next section.

3 Coordination Mechanisms as Cost-Sharing Protocols

In this section we revisit coordination mechanisms that led to surprisingly good price of anarchy guarantees through the lens of cost-sharing protocols. In particular, we reveal that these mechanisms can be interpreted as Shapley value cost-sharing protocols of appropriate total cost functions. This connection directly explains why these mechanisms induce potential games, and it sets the stage for a framework that enables the design of novel local-search approximation algorithms, discussed in Section 4. In this section we first discuss the coordination mechanisms analyzed in [15], and then we prove how all of these mechanisms can be viewed as cost-sharing protocols.

3.1 Coordination Mechanisms

Arguably the most natural scheduling policy for the problem of minimizing the total weighted completion time is the SmithRule (SR). This is a deterministic policy which schedules the jobs assigned to each machine i without preemption, in non-decreasing order with respect to their smith ratios ρ_{ij}. When the weights of all the jobs are equal, this reduces to the shortest–first policy and, for any given assignment $\boldsymbol{\sigma}$, the SmithRule policy is known to minimize the total weighted completion time [33]. Formally, the weighted completion time of each job j under the SmithRule policy is:

$$w_j c_j^{\mathsf{SR}}(\boldsymbol{\sigma}) = \sum_{\substack{k \in J_i(\boldsymbol{\sigma}) \\ \rho_{ik} \prec \rho_{ij}}} w_j p_{ik} + w_j p_{ij}$$

Cole et al. [15] analyzed the game induced by the SmithRule coordination mechanism and showed that its price of anarchy is exactly 4. They also showed that, any mechanism that uses a deterministic policy that orders the jobs without preemption has a price of anarchy of 4.

To achieve a price of anarchy better than 4, Cole et al. [15] analyzed a deterministic but preemptive scheduling policy called ProportionalSharing (PS), and they showed that its price of anarchy is 2.618. According to this policy, all jobs are scheduled in parallel, with each job j receiving a fraction of machine j's processing time that is proportional to its weight, i.e., $w_j / \sum_{k \in J_i} w_k$. The completion times of this scheduling policy are actually equivalent to scheduling the

jobs in a nonpreemptive fashion, just as SmithRule would, but then delaying the completion of each job by an appropriate amount. Although these delays increase the social cost of any given assignment, Cole et al. [15] show that this changes the equilibrium structure of the induced game, thus leading to significantly more efficient equilibria. The delay that a job j suffers beyond its SmithRule completion time is equal to the externality that it causes to jobs that SmithRule schedules after them on the same machine, i.e., $\sum_{k:\rho_{ik} \succ \rho_{ij}} w_k p_{ij}$. Formally, the weighted completion time of job j under the ProportionalSharing policy is:

$$w_j c_j^{\mathsf{PS}}(\boldsymbol{\sigma}) = \sum_{\substack{k \in J_i(\boldsymbol{\sigma}) \\ \rho_{ik} \prec \rho_{ij}}} w_j p_{ik} + \sum_{\substack{k \in J_i(\boldsymbol{\sigma}) \\ \rho_{ik} \succ \rho_{ij}}} w_k p_{ij} + w_j p_{ij}$$

Rand (R) is a randomized scheduling policy according to which, given two jobs j and k assigned to the same machine i, the probability that job k is processed before job j is equal to $q_{kj} := \frac{\rho_{ij}}{\rho_{ij}+\rho_{ik}}$. Once the order of the jobs is randomly generated, the jobs are scheduled one after the other, without preemption. Thus:

$$w_j c_j^{\mathsf{R}}(\boldsymbol{\sigma}) = \sum_{k \in J_i(\boldsymbol{\sigma}) \setminus \{j\}} q_{kj} w_j p_{ik} + w_j p_{ij} = \sum_{k \in J_i(\boldsymbol{\sigma}) \setminus \{j\}} \frac{\rho_{ij}}{\rho_{ij} + \rho_{ik}} w_j p_{ik} + w_j p_{ij}$$

Cole et al. [15] showed that this randomized mechanism has a price of anarchy of 2.133.

Finally, Approx (A) is a deterministic scheduling policy that Cole et al. [15] defined not aiming for an improved price of anarchy bound, but rather for the design of an approximation algorithm for the underlying optimization problem. In particular, the completion times defined by this policy are equal to those of the ProportionalSharing policy, but with the addition of delays for each job j on machine i by exactly p_{ij}.

$$w_j c_j^{\mathsf{A}}(\boldsymbol{\sigma}) = \sum_{\substack{k \in J_i(\boldsymbol{\sigma}) \\ \rho_{ik} \prec \rho_{ij}}} w_j p_{ik} + \sum_{\substack{k \in J_i(\boldsymbol{\sigma}) \\ \rho_{ik} \succ \rho_{ij}}} w_k p_{ij} + 2 w_j p_{ij}$$

The price of anarchy of this mechanism is at most 4, and it can be used to define a polynomial time approximation algorithm for the underlying optimization problem with an approximation factor of 2.

3.2 Cost-Sharing Protocols

In the cost-sharing literature, the group of agents using some resource generates a cost on that resource. This cost depends on the nature of the resource at hand, as well as the set of agents using it, and the goal of the cost-sharing protocol is to share this cost among the users of the resource. At first glance, this is unlike the coordination mechanisms defined above, whose scheduling policies define the costs of the agents directly rather than sharing some well-defined cost. However, there is a natural connection between cost-sharing protocols and coordination

mechanisms. Given an assignment $\boldsymbol{\sigma}$, one can assume that this assignment causes a (social) cost of at least $C_i^{\mathsf{SR}}(\boldsymbol{\sigma})$ on each machine i, since SmithRule is the policy that minimizes that social cost. Hence, the SmithRule policy can be interpreted as a cost-sharing protocol that decides how this social cost is to be shared among the agents. In fact, this is a budget-balanced cost-sharing protocol since the cost that SmithRule divides among the agents adds up to exactly $C_i^{\mathsf{SR}}(\boldsymbol{\sigma})$. More generally, using this approach, we can interpret any coordination mechanism $\mathcal{A}$ that induces a social cost $C_i^{\mathcal{A}}(\boldsymbol{\sigma})$ on machine i as a cost-sharing protocols that decides how this social cost needs to be divided among the jobs using machine i. The following lemmas reveal that, rather surprisingly, ProportionalSharing, Rand, and Approx are *all* using the exact same cost-sharing protocol to divide their total cost: the Shapley value!

$$f_j(\boldsymbol{\sigma}) = \sum_{S \subseteq J_i(\boldsymbol{\sigma}) \setminus \{j\}} \Pr[S, J_i(\boldsymbol{\sigma})](C_i(S \cup \{j\}) - C_i(S)),$$

where $\Pr[S, J_i(\boldsymbol{\sigma})]$ is the probability that the set of jobs that lie before j in the random ordering is exactly the set S.

Lemma 1. ProportionalSharing, Rand, *and* Approx *are all equivalent to the Shapley-value cost-sharing of cost functions* $C^{\mathsf{PS}}(\boldsymbol{\sigma})$, $C^{\mathsf{R}}(\boldsymbol{\sigma})$, *and* $2C^{\mathsf{SR}}(\boldsymbol{\sigma})$ *respectively.*

One important implication of this lemma is that the connection to the Shapley value cost-sharing directly implies that the induced games are potential games, and it also directly implies what the potential function is. Without this observation, Cole et al. [15] had to provide three separate proofs to prove that these games always possess pure Nash equilibria.

More importantly, another implication of this lemma is that it provides a general way of designing coordination mechanisms that are guaranteed to induce potential games: define a cost function $C_i^{\alpha}(\boldsymbol{\sigma})$ for each machine i, which can depend arbitrarily on the set of jobs assigned to that machine, and then use the Shapley value in order to define the cost-share that each job should suffer. It is important to emphasize that a critical difference between cost-sharing protocols and coordination mechanisms is that the latter are restricted by feasibility constraints implied by the scheduling model. That is, there has to exist some feasible schedule for processing the jobs such that each job's cost is equal to what is dictated by the cost-sharing protocol. For instance, in our model, SmithRule is the only budget-balanced mechanism that is also feasible, since any other feasible way of scheduling the jobs would necessarily lead to a higher social cost. An interesting future research direction would be to understand what constraints on the cost function C_i^{α} can guarantee the feasibility of the Shapley value cost-sharing schedule. However, the main result of this paper shows that, even when the induced mechanisms do not yield feasible schedules, they can be used to design novel approximation algorithms for the underlying optimization problem.

4 Approximation Algorithms via Cost-Sharing

Even if the mechanism designer could control which machine each job is assigned to, the problem of computing a feasible schedule that minimizes the total weighted completion time is known to be APX-hard [21]. The best approximation factor guarantee was only very recently improved beyond 1.5, using elaborate lift-and-round techniques on convex programming relaxations, to reach an approximation factor no less than 1.4999 [6]. The machine scheduling literature has traditionally emphasized the importance of *combinatorial* algorithms which, unlike convex program-based solutions, provide more intuition regarding the structure of the problem at hand. To the best of our knowledge, the combinatorial algorithm providing the best approximation guarantees for this problem, prior to our work, yields a factor of $2 + \epsilon$ for some arbitrarily small constant $\epsilon > 0$ [15].

In light of the connection between coordination mechanisms and cost-sharing identified in the previous section, and using the fact that the Shapley value cost-sharing protocol is guaranteed to induce potential games, we now propose a general framework for designing combinatorial approximation algorithms for this problem, using mechanisms based on the Shapley value. The following steps, explained in more detail later on, provide a high-level description of our framework for designing such algorithms:

1. For each machine i and each subset of jobs J_i define some cost $C_i(J_i)$
2. Using the Shapley value protocol, define the cost $f_j(J_i)$ for each $j \in J_i$
3. In a polynomial number of best-response deviations, reach an assignment $\boldsymbol{\sigma}$
4. Assign the jobs according to $\boldsymbol{\sigma}$ and process them using the SmithRule order

The key ingredient in this framework is that, using the Shapley value to divide the chosen costs, $C_i(J_i)$, is guaranteed to induce a potential game. This implies that any sequence of best response dynamics will eventually lead to an assignment that is a pure Nash equilibrium of this game. This may require an exponential number of such deviations but, as we show, we can guarantee that a polynomial number of deviations suffices for this approach to reach an assignment that is an approximate equilibrium. As a result, if our choice of $C_i(J_i)$ gives rise to high quality (approximate) equilibria, then this framework yields a high quality polynomial time algorithm[4].

The $(2 + \epsilon)$-approximation algorithm proposed by Cole et al. [15] for this problem can be directly presented as an instantiation of this framework. Their algorithm uses Approx to define the costs $C_i(J_i) := C_i^{\mathsf{A}}(J_i)$ in the first step and, unbeknownst to them, this mechanism divides these costs among the jobs according to the Shapley value protocol, as we showed in Lemma 1. Then, they prove that the induced game is, in fact, a potential game, and they use this

[4] Note that computing the Shapley value cost-shares can be non-trivial, or even intractable, if the $C_i(J_i)$ is arbitrarily general, but most of the natural choices for this function provide sufficient structure for the shares to be readily computable, as verified in the previous and the following section.

to show that a polynomial number of deviations will lead to an assignment $\boldsymbol{\sigma}$ whose social cost is essentially no worse than that of any equilibrium of the game induced by Approx. Since the price of anarchy of Approx is at most 4, the assignment $\boldsymbol{\sigma}$ that this algorithm computes satisfies $C^{\mathsf{A}}(\boldsymbol{\sigma}) \leq (4+\epsilon)C^{\mathsf{SR}}(\boldsymbol{\sigma}^*)$. But, since $C^{\mathsf{A}}(\boldsymbol{\sigma}) = 2C^{\mathsf{SR}}(\boldsymbol{\sigma})$, this implies that $C^{\mathsf{SR}}(\boldsymbol{\sigma}) \leq (2+\epsilon)C^{\mathsf{SR}}(\boldsymbol{\sigma}^*)$, thus verifying the approximation factor. A more natural presentation of the same algorithm would instead use the optimal social cost, $C_i(J_i) := C_i^{\mathsf{SR}}(\boldsymbol{\sigma})$, in the first step; we refer to the mechanism that shares this optimal cost using the Shapley value as ShapleyValue. Lemma 1 implies that the game induced by ShapleyValue is essentially the same as the one induced by Approx: for every possible assignment $\boldsymbol{\sigma}$, the cost of every job j in Approx is exactly twice its cost in ShapleyValue, i.e., $f_j^{\mathsf{A}}(\boldsymbol{\sigma}) = 2f_j^{\mathsf{SR}}(\boldsymbol{\sigma})$. As a result, the set of equilibrium assignments is exactly the same, and the fact that the price of anarchy of Approx is at most 4 implies that the price of anarchy of ShapleyValue is at most 2. Note that there may not exist feasible schedules that yield completion times compatible with the costs of ShapleyValue, but this is irrelevant since this mechanism is used only as a guide for designing an approximation algorithm.

The obvious open question is whether we can leverage the systematic approach suggested by this framework in order to design improved approximation algorithms. In the following section we show that neither Approx, Rand, or ProportionalSharing can be used to get a better approximation algorithm. However, in the rest of this section, we analyze all the mechanisms that are convex combinations of ShapleyValue and ProportionalSharing and, using our framework, we propose new approximation algorithms, one of which leads to an improved approximation factor of 1.81.

4.1 Approximation Algorithms

Cost Functions. We embark on our search for improved approximation algorithms by considering a class of cost functions that are combinations of the costs implied by ShapleyValue and those implied by ProportionalSharing. We parameterize this class of mechanisms using a value $\beta > 0$, and let $\mathsf{H}(\beta)$ be the mechanism whose social cost on each machine i is

$$C^{\mathsf{H}(\beta)}(\boldsymbol{\sigma}) = 2\beta C^{\mathsf{SR}}(\boldsymbol{\sigma}) - (2\beta - 1)\eta(\boldsymbol{\sigma}),$$

where $\eta(\boldsymbol{\sigma}) = \sum_{i \in I} \sum_{j \in J_i(\boldsymbol{\sigma})} w_j p_{ij}$ is an abbreviation that we use extensively in the following. It is easy to verify that $C^{\mathsf{H}(\beta)}(\boldsymbol{\sigma})$ can be expressed as the combination $\kappa C^{\mathsf{SR}}(\boldsymbol{\sigma}) + \lambda C^{\mathsf{PS}}(\boldsymbol{\sigma})$ for appropriate constants κ and λ. In particular, ProportionalSharing and ShapleyValue are the mechanisms $\mathsf{H}(1)$ and $\mathsf{H}(1/2)$ of the above class. If we share this cost among the set of jobs J_i using machine i, the cost-share of player $j \in J_i$ is

$$f_j^{\mathsf{H}(\beta)}(J_i) = \beta \sum_{\substack{k \in J_i \\ \rho_{ik} \prec \rho_{ij}}} w_j p_{ik} + \beta \sum_{\substack{k \in J_i \\ \rho_{ik} \succ \rho_{ij}}} w_k p_{ij} + w_j p_{ij}.$$

Since we use the Shapley value for cost-sharing, the induced potential game has the following potential function: $\Phi^{\mathsf{H}(\beta)}(\boldsymbol{\sigma}) = \frac{1}{2} C^{\mathsf{H}(\beta)}(\boldsymbol{\sigma}) + \frac{1}{2}\eta(\boldsymbol{\sigma})$.

ϵ-approximate ϵ-equilibrium. The algorithms that we propose compute an ϵ-approximate ϵ-equilibrium of the game induced by $\mathsf{H}(\beta)$. For the definition of the ϵ-approximate ϵ-equilibrium, we follow the notation of [4]. Given an assignment $\boldsymbol{\sigma}$ and a player j, denote by σ'_j a best-response strategy of j. Hence, σ'_j minimizes $f_j^{\mathsf{H}(\beta)}(\boldsymbol{\sigma}_{-j}, \cdot)$ over all strategies of player j. Let $\Delta_j(\boldsymbol{\sigma}) = f_j^{\mathsf{H}(\beta)}(\boldsymbol{\sigma}) - f_j^{\mathsf{H}(\beta)}(\boldsymbol{\sigma}_{-j}, \cdot)$ denote the maximum decrease in her cost player j can gain when deviating from her strategy in $\boldsymbol{\sigma}$, and $\Delta(\boldsymbol{\sigma}) = \sum_{j \in J} \Delta_j(\boldsymbol{\sigma})$ denote the sum of these quantities over all players. If Q denotes the set of players j who can improve their cost by more than an ϵ fraction, i.e., $\Delta_j(\boldsymbol{\sigma}) > \epsilon f_j^{\mathsf{H}(\beta)}(\boldsymbol{\sigma})$, then we say that $\boldsymbol{\sigma}$ is an ϵ-approximate ϵ-equilibrium if the total relative benefit that these players can accrue via unilateral deviations is $\sum_{j \in Q} \Delta_j(\boldsymbol{\sigma}) \leq \epsilon C^{\mathsf{H}(\beta)}(\boldsymbol{\sigma})$.

Algorithm $\mathsf{A}(\boldsymbol{\beta}, \boldsymbol{\epsilon})$. Given parameter values for β and ϵ, the $\mathsf{A}(\beta, \epsilon)$ algorithm simulates a restricted sequence of best-response play in the game induced by $\mathsf{H}(\beta)$ until an ϵ-approximate ϵ-equilibrium assignment $\boldsymbol{\sigma}$ is reached. The restricted sequence begins with an arbitrary assignment and continues while an ϵ-approximate ϵ-equilibrium has not been computed. In each step, among all players who have a deviation that can improve their cost by at least a factor of ϵ, the player who can improve her cost the most is picked to follow a best-response strategy. Once the assignment $\boldsymbol{\sigma}$ is reached, the algorithm terminates by scheduling the jobs within each machine in this final assignment according to $\mathsf{SmithRule}$.

Adapting the results of Awerbuch et al. [4] in our setting (the important condition that allows this adaptation is that $\Phi^{\mathsf{H}(\beta)}(\boldsymbol{\sigma}) \leq C^{\mathsf{H}(\beta)}(\boldsymbol{\sigma})$), we obtain that the above algorithm is guaranteed to find an ϵ-approximate ϵ-equilibrium after $O\left(\frac{n}{\epsilon} \ln \frac{\Phi^{\mathsf{H}(\beta)}(\boldsymbol{\sigma}_0)}{\Phi_{\min}^{\mathsf{H}(\beta)}}\right)$ player moves. Here, n is the number of players, $\boldsymbol{\sigma}_0$ denotes the initial state and $\Phi_{\min}^{\mathsf{H}(\beta)}$ is the globally minimum value of the potential function. Crucially, the above running time is polynomial in terms of the number of bits in the representation of the scheduling instance and $1/\epsilon$.

Our main goal in this section is to show that ϵ-approximate ϵ-equilibria of the games induced by $\mathsf{H}(\beta)$ (for appropriate values of β) are efficient in terms of their $C^{\mathsf{SR}}(\cdot)$ cost. In turn, this will imply good approximation guarantees for algorithm $\mathsf{A}(\beta, \epsilon)$. As a result, we propose two algorithms that provide different approximation guarantees: the first one (for $\beta \approx 0.591$) yields a 1.81-approximation for any instance (Theorem 1), and the second (for $\beta = 2/3$) yields an approximation that converges to 1 as $\eta(\boldsymbol{\sigma}^*)/C^{\mathsf{SR}}(\boldsymbol{\sigma}^*)$ converges to zero (Theorem 2)

Theorem 1. *Let $\epsilon \in (0, 1/12]$ and $\beta = \frac{9+\sqrt{5}}{19} \approx 0.591$. Algorithm $\mathsf{A}(\beta, \epsilon)$ runs in polynomial time and computes an assignment $\boldsymbol{\sigma}$ such that $C^{\mathsf{SR}}(\boldsymbol{\sigma}) \leq \left(\frac{5+\sqrt{5}}{4} + 8\epsilon\right) C^{\mathsf{SR}}(\boldsymbol{\sigma}^*) \approx (1.809 + 8\epsilon) C^{\mathsf{SR}}(\boldsymbol{\sigma}^*)$, where $\boldsymbol{\sigma}^*$ is the assignment that minimizes the total weighted completion time of the scheduling instance on input.*

Finally, Theorem 2 shows that the cost of the solution obtained by $\mathsf{H}(2/3)$ approaches optimality as the ratio $\eta(\boldsymbol{\sigma}^*)/C^{\mathsf{SR}}(\boldsymbol{\sigma}^*)$ tends to 0.

Theorem 2. *Let* $\epsilon \in (0, 1/12]$ *and* $\beta = 2/3$. *Algorithm* $\mathsf{A}(\beta, \epsilon)$ *runs in polynomial time and computes an assignment* $\boldsymbol{\sigma}$ *with* $C^{\mathsf{SR}}(\boldsymbol{\sigma}) \leq (1 + 6\epsilon)\left(C^{\mathsf{SR}}(\boldsymbol{\sigma}^*) + \eta(\boldsymbol{\sigma}^*)\right)$, *where* $\boldsymbol{\sigma}^*$ *is the assignment that minimizes the total weighted completion time of the scheduling instance on input.*

5 Lower Bounds for Approx, Rand, and ProportionalSharing

In this section we show that, even for the special class of related machines, neither one of the coordination mechanisms studied in [15] can be used in order to design an algorithm with an approximation factor better than 2. In particular, we show that each one of these mechanisms possesses an equilibrium assignment $\boldsymbol{\sigma}$ whose optimal social cost is at least two times $C^{SR}(\boldsymbol{\sigma}^*)$. For the SmithRule policy this was already shown in [15], since its price of anarchy is exactly 4. For the ProportionalSharing policy, [10, Theorem 15] provides an instance showing that its price of anarchy is at least 2.618, i.e., there exists an equilibrium $\boldsymbol{\sigma}$ such that $C^{\mathsf{PS}}(\boldsymbol{\sigma}) \geq 2.618 C^{\mathsf{SR}}(\boldsymbol{\sigma}^*)$. Although this inequality is not the desired lower bound, the equilibrium assignment $\boldsymbol{\sigma}$ in this instance happens to assign a single job on each machine, and hence $C^{\mathsf{SR}}(\boldsymbol{\sigma}) = C^{\mathsf{PS}}(\boldsymbol{\sigma}) \geq 2.618 C^{\mathsf{SR}}(\boldsymbol{\sigma}^*)$, so ProportionalSharing cannot yield an algorithm with an approximation factor better than 2.618.

The main result of this section is a construction verifying that Approx and Rand cannot lead to an improved approximation algorithm either. In particular, we prove that the price of anarchy of ShapleyValue is at least 2, which implies the existence of some assignment $\boldsymbol{\sigma}$ which is an equilibrium for ShapleyValue and satisfies $C^{\mathsf{SR}}(\boldsymbol{\sigma}) \geq 2C^{\mathsf{SR}}(\boldsymbol{\sigma}^*)$. But, as we already observed in the previous sections, any equilibrium of ShapleyValue is also an equilibrium of Approx. Furthermore, our lower bound construction uses jobs with $p_{ij} = w_j/s_i$ on each machine i, where s_i is the speed of machine i. Since $\rho_{ij} = \rho_{ik}$ for every machine i and every pair of jobs j and k, we have $f_j^{\mathsf{SR}}(\sigma) = f_j^{\mathsf{R}}(\sigma)$, i.e., the game induced by the protocol also coincides with the game induced by Rand on these instances. As a result any equilibrium of ShapleyValue for these instances is also an equilibrium of Rand, and our lower bound construction kills two birds with one stone, verifying the limitations of both Approx and Rand.

It is worth noting that, apart from the implications regarding our inability to design improved approximation algorithms using these policies, this also yields a lower bound of 2 on the price of anarchy of Rand, which improves upon the best previously known lower bound of 5/3 [15].

Theorem 3. *The price of anarchy of* ShapleyValue *is at least* 2, *even on related machine instances.*

6 Conclusion

The main contribution of this paper is a framework for the systematic design of coordination mechanisms and approximation algorithms. Leveraging previous work on cost-sharing protocols, this framework produce mechanisms that

always possess pure Nash equilibria, and approximation algorithms that terminate in a polynomial number of steps. To verify that this framework can lead to novel results, we provide a combinatorial approximation algorithm for machine scheduling on unrelated machines. Although we focus on the unweighted Shapley value, an even richer family of mechanisms and algorithms can be designed using generalized weighted Shapley value variants [19]. Our results call for a better understanding of the conditions under which the Shapley value cost-shares can be implemented via a (randomized) feasible schedule; using this understanding one could design coordination mechanisms whose price of anarchy bounds outperform the currently best known bound of 2.133 from Rand. Furthermore, our framework could be applied more broady to design mechanisms and algorithms for general resource selection games.

References

1. Abed, F., Correa, J.R., Huang, C.C.: Optimal coordination mechanisms for multi-job scheduling games. In: Proceedings of the 22nd Annual European Sympoisum on Algorithms (ESA). pp. 13–24 (2014)
2. Abed, F., Huang, C.C.: Preemptive coordination mechanisms for unrelated machines. In: Proceedings of the 20th Annual European Sympoisum on Algorithms (ESA), pp. 12–23. Springer (2012)
3. Anshelevich, E., Dasgupta, A., Kleinberg, J.M., Tardos, É., Wexler, T., Roughgarden, T.: The price of stability for network design with fair cost allocation. SIAM J. Comput. 38(4), 1602–1623 (2008)
4. Awerbuch, B., Azar, Y., Epstein, A., Mirrokni, V.S., Skopalik, A.: Fast convergence to nearly optimal solutions in potential games. In: Proceedings of the 9th ACM Conference on Electronic Commerce (EC). pp. 264–273 (2008)
5. Azar, Y., Fleischer, L., Jain, K., Mirrokni, V.S., Svitkina, Z.: Optimal coordination mechanisms for unrelated machine scheduling. Operations Research 63(3), 489–500 (2015)
6. Bansal, N., Srinivasan, A., Svensson, O.: Lift-and-round to improve weighted completion time on unrelated machines. In: Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC). pp. 156–167 (2016)
7. Bhattacharya, S., Im, S., Kulkarni, J., Munagala, K.: Coordination mechanisms from (almost) all scheduling policies. In: Proceedings of the 5th Conference on Innovations in Theoretical Computer Science (ITCS). pp. 121–134 (2014)
8. Caragiannis, I.: Efficient coordination mechanisms for unrelated machine scheduling. Algorithmica 66(3), 512–540 (2013)
9. Caragiannis, I., Fanelli, A.: An almost ideal coordination mechanism for unrelated machine scheduling. In: Proceedings of the 9th International Symposium on Algorithmic Game Theory (SAGT). pp. 315–326 (2016)
10. Caragiannis, I., Flammini, M., Kaklamanis, C., Kanellopoulos, P., Moscardelli, L.: Tight bounds for selfish and greedy load balancing. Algorithmica 61(3), 606–637 (2011)
11. Chen, H., Roughgarden, T., Valiant, G.: Designing network protocols for good equilibria. SIAM J. Comput. 39(5), 1799–1832 (2010)
12. Christodoulou, G., Koutsoupias, E., Nanavati, A.: Coordination mechanisms. Theoretical Computer Science 410(36), 3327–3336 (2009)

13. Christodoulou, G., Gkatzelis, V., Sgouritsa, A.: Cost-sharing methods for scheduling games under uncertainty. In: Proceedings of the 18th ACM Conference on Economics and Computation (EC). pp. 441–458 (2017)
14. Christodoulou, G., Mehlhorn, K., Pyrga, E.: Improving the price of anarchy for selfish routing via coordination mechanisms. Algorithmica 69(3), 619–640 (2014)
15. Cole, R., Correa, J.R., Gkatzelis, V., Mirrokni, V., Olver, N.: Decentralized utilitarian mechanisms for scheduling games. Games and Economic Behavior 92, 306–326 (2014)
16. Correa, J.R., Queyranne, M.: Efficiency of equilibria in restricted uniform machine scheduling with total weighted completion time as social cost. Naval Research Logistics (NRL) 59(5), 384–395 (2012)
17. von Falkenhausen, P., Harks, T.: Optimal cost sharing for resource selection games. Math. Oper. Res. 38(1), 184–208 (2013)
18. Gkatzelis, V., Kollias, K., Roughgarden, T.: Optimal cost-sharing in general resource selection games. Operations Research 64(6), 1230–1238 (2016)
19. Gopalakrishnan, R., Marden, J.R., Wierman, A.: Potential games are *necessary* to ensure pure Nash equilibria in cost sharing games. Math. Oper. Res. 39(4), 1252–1296 (2014)
20. Hoeksma, R., Uetz, M.: The price of anarchy for minsum related machine scheduling. In: Workshop on Approximation and Online Algorithms. pp. 261–273 (2011)
21. Hoogeveen, H., Schuurman, P., Woeginger, G.J.: Non-approximability results for scheduling problems with minsum criteria. INFORMS Journal on Computing 13(2), 157–168 (2001)
22. Immorlica, N., Li, L.E., Mirrokni, V.S., Schulz, A.S.: Coordination mechanisms for selfish scheduling. Theoretical Computer Science 410(17), 1589–1598 (2009)
23. Kollias, K.: Nonpreemptive coordination mechanisms for identical machines. Theory of Computing Systems 53(3), 424–440 (2013)
24. Kollias, K., Roughgarden, T.: Restoring pure equilibria to weighted congestion games. ACM Trans. Economics and Comput. 3(4), 21:1–21:24 (2015)
25. Leung, J.Y. (ed.): Handbook of Scheduling - Algorithms, Models, and Performance Analysis. Chapman and Hall/CRC (2004)
26. Li, S.: Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. CoRR abs/1707.08039 (2017)
27. Marden, J.R., Wierman, A.: Distributed welfare games. Operations Research 61(1), 155–168 (2013)
28. Mosk-Aoyama, D., Roughgarden, T.: Worst-case efficiency analysis of queueing disciplines. In: Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP), pp. 546–557. Springer (2009)
29. Moulin, H.: The price of anarchy of serial, average and incremental cost sharing. Economic Theory 36(3), 379–405 (2008)
30. Sethuraman, J., Squillante, M.: Optimal scheduling of multiclass parallel machines. In: Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 963–964 (1999)
31. Shapley, L.S.: Additive and non-additive set functions. Princeton University (1953)
32. Skutella, M.: Convex quadratic and semidefinite programming relaxations in scheduling. J. ACM 48(2), 206–242 (2001)
33. Smith, W.: Various optimizers for single stage production. Naval Res. Logist. Quart. 3(1-2), 59–66 (1956)