Fair Cost-Sharing Methods for Scheduling Jobs on Parallel Machines[⋆]

Yvonne Bleischwitz[1,2] and Burkhard Monien[1]

[1] Faculty of Computer Science, Electrical Engineering and Mathematics, University of Paderborn, Fürstenallee 11, 33102 Paderborn, Germany
{yvonneb, bm}@upb.de
[2] International Graduate School of Dynamic Intelligent Systems

Abstract. We consider the problem of sharing the cost of scheduling n jobs on m parallel machines among a set of agents. In our setting, each agent owns one job and the cost is given by the makespan of the computed assignment. We focus on α-budget-balanced cross-monotonic cost-sharing methods since they guarantee the two substantial mechanism properties α-budget-balance and group-strategyproofness and provide fair cost-shares. For identical jobs on related machines and for arbitrary jobs on identical machines, we give $(m+1)/(2m)$-budget-balanced cross-monotonic cost-sharing methods and show that this is the best approximation possible. As our major result, we prove that the approximation factor for cross-monotonic cost-sharing methods is unbounded for arbitrary jobs and related machines. We therefore develop a cost-sharing method in the $(m+1)/(2m)$-core, a weaker but also fair solution concept. We close with a strategyproof mechanism for the model of arbitrary jobs and related machines that recovers at least $3/5$ of the cost. All given solutions can be computed in polynomial time.

1 Introduction

Motivation and Framework. We consider the scenario, in which a service provider owns a set of machines and receives requests from agents to execute their jobs. Each agent has a non-publicly observable preference for his job to be processed. He submits a bid to the service provider that indicates the amount of money he is willing to pay. If his job is processed, he has to make a payment to the service provider. We refer to a payment as *cost-share*. The utility of an agent expresses his valuation of receiving the service at a certain cost-share. The aim of an agent is to maximize his utility. We assume that agents are selfish. Therefore the provider can generally not rely on receiving *truthful bids*, i.e. bids that equal the private preferences.

In our model, the provider's cost of assigning jobs to his machines is given by the *makespan*, i.e. the time needed until all machines have processed their assigned jobs. The provider's problem is to determine the set of served agents, their cost-shares, and a valid assignment for the served agents. He would like to recover as much of the cost as

[⋆] This work has been partially supported by the German Science Foundation (DFG) priority program 1126 *Algorithms of Large and Complex Networks* under grant MO 285/15-3, and by the European Union within the 6th Framework Programme under contract 001907 (DELIS).

possible. Furthermore, he aims to minimize the makespan for reasons of efficiency and he wants to prevent being manipulated by the agents. To be practicable, his problem has to be computable in polynomial time. Since his scheduling problem is NP-hard in general, he has to apply approximation algorithms. The proposed scenario is of particular importance for commercial computing centers as well as for the evolving commercial grid computing offerings.

The problem of scheduling a set of n jobs on a set of m parallel machines with the objective of minimizing the makespan is an extensively studied problem. The most commonly used models are the models of *related* and *unrelated* machines. In the model of related machines, the completion time of a job on a machine does only depend on its workload and on the speed of the machine, where in the model of unrelated machines, machines have player-specific completion times. Recently, these models have been considered in the context of game theory. In this branch of research, there is no central authority that assigns the jobs, but selfish agents themselves assign their jobs to machines. The objective is to obtain an assignment in Nash equilibrium in which no agent can profit by assigning his job to another machine, given that all other agents leave the assignment of their jobs unchanged.

We recall the provider's problem, which is to determine a set of served agents, their cost-shares, and an assignment for the set of served agents. We can utilize assignment algorithms to compute the assignment but need different tools for determining the set of service-receiving agents and their cost-shares.

The theory of mechanism design proposes *cost-sharing mechanisms* that provide a solution to the problem of choosing the set of served agents and their payments. These mechanisms apply *cost-sharing methods* to determine the cost-shares. Two important fairness properties of cost-sharing methods are *cross-monotonicity* and the *core property*. Cross-monotonic cost-sharing methods require that the cost-share of an agent can only decrease if more agents receive the service. The weaker core property assures, that a coalition is always charged not more than the optimal cost of exclusively assigning the jobs of the coalition. This implies that no coalition is overcharged. Furthermore, a cost-sharing method can be *α-budget-balanced*, which guarantees that the service provider covers an α-fraction of his cost and assures the serviced agents that their collective cost-share is not larger than the cost of an optimal solution. If it additionally satisfies the core-property, we say that it is in the *α-core*. Significant properties of cost-sharing mechanisms are *strategyproofness* and *group-strategyproofness*, demanding that an agent or a group of agents can not improve their utility by submitting untruthful bids. This keeps them from manipulating the service provider. Cross-monotonic cost-sharing methods play a very important role in the design of cost-sharing mechanism, since they can be applied to design group-strategyproof mechanisms [25, 18].

Contribution and Significance. The main contributions of this paper are results on cost-sharing methods that are both α-budget-balanced and cross-monotonic. To the best of our knowledge, this paper is the first to introduce cross-monotonic cost-sharing methods for scheduling jobs on parallel machines. We prove that cross-monotonic cost-sharing methods that are α-budget-balanced do not exist for $\alpha > (m + 1)/(2m)$, not even for identical jobs and machines, and give cross-monotonic methods with factor $\alpha = (m+1)/(2m)$ for arbitrary jobs and identical machines or identical jobs and related machines. For arbitrary jobs and related machines, cross-monotonicity is impracticable. Our results show, that it is impossible to obtain cross-monotonic cost-sharing methods that recover more than a $1/d$-fraction of the cost, and that it is possible to recover a $1/(2d)$-fraction, where d is the number of different workloads.

In order to achieve a better approximation, we design a weaker but also fair cost-sharing method that is in the $(m + 1)/(2m)$-core. In addition, we propose a strategyproof mechanism that recovers at least $3/5$ of the cost and makes no agent pay more than if his job were solely processed. All proposed methods run in polynomial time and compute Nash equilibria.

Related Work. The assignment problem for the model of unrelated [30, 22, 16] and related [8, 10, 12, 14, 15] machines has been extensively studied in the past. We focus on the model of related machines. Hochbaum and Shmoys [15] give a PTAS for this model. In this paper, we frequently apply the LPT algorithm proposed by Graham [12]. LPT is optimal for identical jobs, achieves an approximation ratio of $4/3 - 1/(3m)$ for identical machines [12] and an approximation ratio of $5/3$ for related machines [9]. It is explained in Section 2. For results on computing Nash equilibria, we refer to the surveys of Gairing et al. [11] and Czumaj [4].

Cost-sharing mechanisms have mainly been designed for multicast [7, 6, 1], set cover [5], facility location [5, 27, 23], Steiner trees [19, 18], Steiner forests [20, 21], multicommodity rent-or-buy [3], and single-source rent-or-buy [27, 13]. Penna and Ventre [28] study algorithmic properties of cost-sharing mechanisms that among other properties satisfy group strategyproofness and budget-balance.

Cross-monotonic cost-sharing methods have been investigated for facility location [27, 23], single-source rent-or-buy [27, 13], and Steiner trees and forests [19, 18, 20]. Impossibility results are given by Immorlica et al. [17]. Moulin and Shenker [25] study the relations between group-strategyproofness and cross-monotonicity. One of their central results is a mechanism that is group-strategyproof if it applies a cross-monotonic cost-sharing method. The core is a well studied solution concept that stems from coalitional games with transferable payoffs and has for example been considered by Shapley [29].

Results on scheduling in the mechanism design context exist for other scheduling models. With regard to the fairness concept of the Shapley value, Mishra et al. [24] investigate the case in which there is only one server that can serve only one job at a time. Nisan and Ronen [26] consider unrelated parallel machines. In contrast to our model, machines are owned by agents that submit bids on execution times. They give a strategyproof mechanism that computes an assignment with makespan smaller than m times the optimal makespan and conjecture that this is the best possible. They prove that there is no strategyproof mechanism that computes an assignment with makespan smaller than 2 times the optimal makespan. Archer and Tardos [2] consider the scenario in which agents own related machines and give a strategyproof mechanism whose computed assignment yields a makespan that is smaller than 3 times the optimal makespan.

Road Map. Section 2 gives the basic definitions from mechanism design and defines the scheduling problem. Our results on cross-monotonicity are given in Section 3. Finally, Section 4 mainly focuses on results for the core and gives a strategyproof mechanism.

2 Definitions

Let N be the set of potential customers with $|N| = n$. The set of machines owned by the service provider is denoted by M. Agent $i \in N$ has a private preference $v_i \in \mathbb{R}_{\geq 0}$ for his job to be processed. If his job is processed at a certain cost-share $x_i \in \mathbb{R}_{\geq 0}$ his utility is defined as $u_i = v_i - x_i$. Otherwise, his cost-share and his utility are zero. In his request of being served, he submits a bid b_i that corresponds to the amount of money he is willing to pay. Since he is guided by self-interest, he chooses his bid such as to maximize his utility. The provider experiences a certain cost by scheduling a set of jobs. We assume that this cost is given by the makespan as defined in Section 2.2. His problem is to determine a set of served agents $U \subseteq N$, their cost-shares $x_i(U) \in \mathbb{R}_{\geq 0}$, that recover as much of his cost as possible, and a valid assignment for U.

We give the basic definitions and results on cost sharing methods and mechanisms for an unspecified service in Section 2.1. This section provides solutions on how the provider can extract the agents' real preferences while recovering a certain fraction of his cost. Section 2.2 specifies the service of scheduling the agents' jobs. Throughout the paper, we use $[k], k \in \mathbb{N}$, to denote the set $\{1, \ldots, k\}$ of integers.

2.1 Mechanism Design for Cost-Sharing

Let $c_{\mathcal{A}}(U)$ be the cost of a solution computed by some algorithm $\mathcal{A}$ to provide the service to $U \subseteq N$. In many cases, this algorithm is an approximation algorithm to assure polynomial time. We write $opt(U)$ for the cost of an optimal solution to provide the service to U. For a given set $U \subseteq N$, a *cost-allocation function* $\xi : N \rightarrow \mathbb{R}$ *for the set* $U \subseteq N$ specifies the cost-shares of each $i \in U$. It satisfies $\xi(i) \geq 0$ for all $i \in U$ and $\xi(i) = 0$ for all $i \notin U$. Let $\xi(U) = \sum_{i \in U} \xi(i)$. A *cost-sharing method* is a collection of cost-allocation functions:

Definition 1 (cost-sharing method). *A cost-sharing method x is defined as a function $x : N \times 2^N \rightarrow \mathbb{R}$ satisfying for all $U \subseteq N$, that $x(i,U) \geq 0$ for all $i \in U$ and $x(i,U) = 0$ for all $i \notin U$. We will denote $x(i,U)$ by $x_i(U)$. Let $x(U) = \sum_{i \in U} x_i(U)$.*

Ideally, we would like to have *budget-balance*, i.e. $c_{\mathcal{A}}(U) = x(U) = opt(U)$ for all $U \subseteq N$. In many cases it is not possible to achieve budget-balance if the cost-sharing method is to meet other properties as well, or it is computationally hard to compute. Therefore, this condition is relaxed. A cost-sharing method $x(\cdot)$ is *α-budget-balanced* for $\alpha \leq 1$ if it satisfies $\alpha c_{\mathcal{A}}(U) \leq x(U) \leq opt(U)$ for all $U \subseteq N$. A cost-allocation function for $U \subseteq N$ is *α-budget-balanced*, if the above condition holds for U. Observe, that dividing the cost-shares by α results in cost-shares that guarantee the full coverage of the actual cost and an overall cost-share of less than α^{-1} times the optimal solution. Although this is the more intuitive definition we use the definition given first for reasons of clearness.

Both α-budget-balanced cost-sharing methods and cost-allocation functions can have the property to be in the α-core. Intuitively, no coalition is overcharged:

Definition 2 (the α-core property). *A cost-allocation function $\xi(\cdot)$ for $U \subseteq N$ is in the α-core iff it is α-budget-balanced and for all $U' \subseteq U : \xi(U') \leq opt(U')$. A cost-sharing method $x(\cdot)$ is in the α-core iff for all $U \subseteq N, x(\cdot, U)$ is in the α-core.*

The provider's problem can be solved by a *cost-sharing mechanism* and it's underlying cost-sharing method. A cost sharing mechanism is an algorithm that is given the agents' bids $\{b_i\}_{i \in N}$. It outputs the set of agents $U \subseteq N$ that receive the service and cost-shares $x_i(U) \in \mathbb{R}$ with $0 \leq x_i(U) \leq b_i$ for all $i \in U$ and $x_i(U) = 0$ for all $i \notin U$. Furthermore, it outputs a solution with cost $c_{\mathcal{A}}(U)$ to provide the service to U. We focus on assuring the following mechanism properties:

- **strategyproofness:** Agent $i \in N$ maximizes his utility by bidding $b_i = v_i$.
- **group-strategyproofness:** A coalition $U \subseteq N$ of users cannot collude and submit untruthful bids such that as a result, each of them has at least the same utility and at least one of them has a strictly larger utility compared to the outcome that results if each of them bids truthfully.
- **α-budget-balance, $\alpha \leq 1$:** $\alpha c_{\mathcal{A}}(U) \leq x(U) \leq opt(U)$ holds for the set U of service-receiving agents.

Cross-monotonic cost-sharing methods play a crucial role in the context of how to achieve α-budget-balance and group-strategyproofness.

Definition 3 (cross-monotonicity). *A cost-sharing method $x(\cdot)$ is cross-monotonic if for all $U, U' \subseteq N, U' \subseteq U : x_i(U') \geq x_i(U)\ \forall i \in U'$.*

If the underlying cost sharing method is cross-monotonic and α-budget-balanced, a simple mechanism given by Moulin and Shenker [25] is α-budget-balanced and group-strategyproof [18].

It is easy to see that each α-budget-balanced cross-monotonic cost sharing method is in the α-core. From this we can conclude that if there is no cost-allocation function for some set $U \subseteq N$ in the α-core, then no α-budget-balanced cross-monotonic cost-sharing method can exist. On the other hand, there can be cost-sharing methods that are in the α-core and are not cross-monotonic.

2.2 The Scheduling Problem

Let N be the set of n agents with $|N| = n$. Each agent $i \in N$ owns exactly one job of workload $w_i \in \mathbb{N}$. Therefore, we will use $U \subseteq N$ to denote agents and jobs interchangeably. For $U \subseteq N$, let $W(U) = \sum_{i \in U} w_i$ and $w_{max}(U) = \max_{i \in U} w_i$. Let $d(U)$ denote the number of different workloads in U. Moreover, there is a set M of m machines. Each machine $j \in M$ has speed $s_j \in \mathbb{N}$. We assume that $s_1 \geq \ldots \geq s_m$. For $M' \subseteq M$, let $S(M') = \sum_{j \in M'} s_j$. If all speeds are the same, we say that the machines are identical. Otherwise we call them related. Jobs are identical, if all workloads are the same. Without loss of generality we assume that identical machines and jobs have speeds and workloads of one respectively.

An assignment allocates each job to exactly one machine. For a given assignment, let δ_j be the sum of the workloads of the jobs assigned to machine j. Then the completion time of a job assigned to machine j is (δ_j/s_j). The makespan is defined as $\max_{j \in M}(\delta_j/s_j)$. We call the machines whose completion time is equal to the makespan *makespan machines*. The optimal solution for a set of jobs $U \subseteq N$ is an assignment with minimal makespan, denoted by $opt(U)$.

To compute an assignment, we apply Graham's LPT algorithm [12]. LPT processes the jobs in decreasing order and assigns each job to a machine on which it experiences the smallest completion time (taking into account the jobs that have been assigned already). For a set $U \subseteq N$ we use $lpt(U)$ to denote the makespan resulting from LPT, i.e. $lpt(U) = c_{LPT}(U)$. For an assignment for jobs $U \subseteq N$ computed by LPT, let $m(U)$ be the set of machines that jobs are assigned to. The running time of LPT is $O(n)$ for identical jobs and identical machines, $O(n \log m)$ for identical jobs and related machines, and $O(n \log n)$ otherwise. Even though there are better approximation algorithms for the assignment problem [8, 10, 14, 15], our main results cannot be improved by switching to another algorithm. A nice additional property of LPT that we exploit in most proofs is that in each iteration, the current assignment is in Nash equilibrium. An assignment is in Nash equilibrium, if no agent can improve by deviating from the current assignment, i.e. for each job i from the set of served agents $U \subseteq N$ that is assigned to machine $j \in M$ it holds that $(\delta_k + w_i)/s_k \geq \delta_j/s_j$ for all $k \in M \backslash \{j\}$.

There are three LPT specific assignment properties that we will utilize in our proofs. Lemma 1 states these well-known properties.

Lemma 1. *Let $U \subseteq N$ and let $\hat{U} \subseteq U$ be the jobs assigned by LPT until the makespan first occurs, and let $\tau = |m(\hat{U})|$. Then it holds, that:*

1. *For identical machines, $W(U)/m \leq opt(U)$.*
2. *For related machines, $W(\hat{U})/S(m(\hat{U})) \leq opt(\hat{U})$.*
3. *If machines are identical and there are at least two jobs assigned to a makespan machine, then $lpt(U) \leq \frac{2m}{m+1}\frac{W(U)}{m}$.*
4. *If there are at least two jobs assigned to some machine, then*
   - *for related machines: $lpt(U) \leq \frac{2\tau}{\tau+1}\frac{W(\hat{U})}{S(m(\hat{U}))}$.*
   - *for identical jobs: $lpt(U) \leq \frac{2m(U)}{m(U)+1}\frac{|U|}{S(m(U))}$.*

3 Results on Cross-Monotonicity

In this section, we give α-budget-balanced cross-monotonic cost-sharing methods that yield α-budget-balanced group-strategyproof mechanisms if used as input for the mechanism by Moulin and Shenker [25]. All proposed methods rely on solving the assignment problem via LPT. The property that LPT computes a Nash equilibrium is utilized frequently. We say that an algorithm *computes a cost-sharing method* $x(\cdot)$ in time $f(m, n)$ if for each set $U \subseteq N$ the cost-shares $\{x_i(U)\}_{i \in U}$ are computed in time $f(m, n)$. The mechanism by Moulin and Shenker runs in time $O(nf(m, n) + g(m, n))$, where $g(m, n)$ is the running time of LPT. Proofs omitted due to space restrictions are provided in the full version of this paper.

Theorems 1 and 2 propose $(m + 1)/(2m)$-budget-balanced cross-monotonic cost-sharing methods for the scheduling problem with identical jobs and for the scheduling problem with identical machines. Due to Theorem 5 in Section 4, these cross-monotonic methods achieve the best budget-balance factor possible.

Theorem 1. *There is an $\frac{m+1}{2m}$-budget-balanced cross-monotonic cost-sharing method for the scheduling problem with arbitrary jobs and identical machines computable in time $O(n)$.*

Theorem 2. *There is an $\frac{m+1}{2m}$-budget-balanced cross-monotonic cost-sharing method for the scheduling problem with identical jobs and related machines computable in time $O(n \log m)$.*

The central Theorem 4 states, that the approximation factor for cross-monotonic cost-sharing methods is unbounded for arbitrary jobs and related machines. It depends on $d(N)$, the number of different workloads in the set N. By Theorem 3, it is possible to achieve $(2d(N))^{-1}$-budget-balance:

Theorem 3. *There is a $(2d(N))^{-1}$-budget-balanced cross-monotonic cost-sharing method for the scheduling problem with arbitrary jobs and related machines computable in time $O(n \log n)$.*

Theorem 4. *For the scheduling problem with arbitrary jobs and related machines, there is no α-budget-balanced cross-monotonic cost-sharing method for the factor α with $\alpha > (d(N) + \epsilon)^{-1}$, $\forall \epsilon > 0$.*

Proof. We proceed as follows: we fix a set of machines and consider classes of scheduling instances in which the job workloads equal their speeds. Classes are defined by specifying the number of agents and jobs respectively of a certain job workload. For average cost-shares on these instances, we derive properties that are met by all cross-monotonic cost-sharing methods. Afterwards, we derive a bound on α-budget balance where α will be determined later.

Instances. The considered classes consist of instances with $d = d(N)$ different workloads $1, a, \ldots, a^{d-1}$, with $a \in \mathbb{N}_{>1}$. There are m_j machines having a speed of a^{d-j} with $j \in [d]$. Let $m_1 = 1$ and $m_j = (a-1)\sum_{l=1}^{j-1} m_l a^{j-l}$ for $j \geq 2$. It holds that $m_j = a^2 m_{j-1}$ for $j \geq 3$, which can easily be proved by induction. We use the more complicated formulation that simplifies later arguments. For $j \in [d]$, let N_j be the set of all agents with jobs of workload a^{d-j} and $n_j = |N_j|$. Then, $N = \cup_{j \in [d]} N_j$. For $U \subseteq N$, let $U_j = U \cap N_j$. U_j extracts from U all jobs with workload a^{d-j}. Let the profile $(u_1, \ldots, u_d)$ denote the class of all sets U with $u_j = |U_j|$ for all $j \in [d]$.

Optimal Assignments. First, consider the class $(m_1, \ldots, m_d)$. Obviously, for every instance of this class consisting of the set of jobs U, $opt(U) = 1$. Now change the jth entry to $r_j = am_j$. We show, that $opt(U) = a$ for every instance with the set of jobs U of the class $(m_1, \ldots, r_j, \ldots, m_d)$ for all $j \in [d]$. First, we give an assignment with makespan a. Then we show that it is impossible to obtain a makespan smaller than a.

The assignment is computed as follows. Assign all jobs of workload a^{d-l}, $l \in [d]$ to the machines of speed a^{d-l}. This results in a completion time of one on machines with speed $a^{d-l}, l \neq j$ and a completion time of a on machines with speed a^{d-j}.

Now we show a lower bound for the optimal assignment. Assume, that there exists an assignment with makespan smaller than a. Observe, that all jobs with workload larger than a^{d-j} have to be assigned to the machines with speed larger than a^{d-j}. Now look at the jobs with workload a^{d-j}. They can only be assigned to the machines with speed at least a^{d-j}. At most $(a-1)m_j$ of them can be assigned to the machines with speed a^{d-j}. Now, all jobs of workload larger than a^{d-j} and the m_j remaining jobs of workload a^{d-j} have to be assigned to the machines with speed larger than a^{d-j}. The makespan cannot be smaller than a, because a lower bound for the optimal assignment for these jobs on these machines is given by a:

$$\frac{\sum_{l=1}^{j-1}(m_l a^{d-l}) + m_j a^{d-j}}{\sum_{l=1}^{j-1}(m_l a^{d-l})} = \frac{\sum_{l=1}^{j-1}(m_l a^{d-l}) + (a-1)\sum_{l=1}^{j-1}(m_l a^{d-l})}{\sum_{l=1}^{j-1}(m_l a^{d-l})} = a\ . \quad (1)$$

Cross-Monotonicity. In the following, we assume, that there is a cross-monotonic cost-sharing method $x(\cdot)$. Let $\Gamma(m_1, \ldots, m_d) = \prod_{l=1}^{d} \binom{n_l}{m_l}$. For all instances of the class $(m_1, \ldots, m_d)$, the average cost share of the agents with jobs in $N_k, k \in [d]$ is

$$\chi_k := \chi_k((m_1, \ldots, m_d)) := \Gamma(m_1, \ldots, m_d)^{-1} \sum_{\substack{U \subset N \\ \forall l : |U_l| = m_l}} \sum_{i \in U_k} x_i(U)\ . \quad (2)$$

Now change the jth profile entry to $r_j = a m_j$. Then, the average cost-share for agents with jobs in N_k is $\chi_k((m_1, \ldots, r_j, \ldots, m_d))$. Define $\Gamma = \Gamma(m_1, \ldots, m_d)$ and also $\Gamma_j = \Gamma(m_1, \ldots, r_j, \ldots, m_d)$. We will utilize cross-monotonicity to bound it from above in terms of χ_k.

Consider the set $U_j \subseteq N_j$ with $|U_j| = r_j$ and $U_j \subseteq U \subseteq N$. First, let $k = j$. Every single cost-share of an agent $i \in U_j$ for the set U is not larger than his cost-share for the set $(U \backslash U_j) \cup \{i\} \cup \tilde{U}$, with $\tilde{U} \subset U_j \backslash \{i\}, |\tilde{U}| = m_j - 1$. Especially, it is not larger than the average value of the cost-shares for i for each of these $\binom{r_j - 1}{m_j - 1}$ sets. Therefore an upper bound of $\chi_j((m_1, \ldots, r_j, \ldots, m_d))$ is given by:

$$\chi_j^{(j)} := \Gamma_j^{-1} \sum_{\substack{U \subseteq N \\ \forall l \neq j : |U_l| = m_l \\ |U_j| = r_j}} \sum_{i \in U_j} \sum_{\substack{\tilde{U} \subset U_j \backslash \{i\} \\ |\tilde{U}| = m_j - 1}} \frac{x_i((U \backslash U_j) \cup \{i\} \cup \tilde{U})}{\binom{r_j - 1}{m_j - 1}}\ . \quad (3)$$

Now, let $k \neq j$. Every single cost-share of an agent $i \in U \backslash U_j$ for U is not larger than the cost-share for i for $(U \backslash U_j) \cup \tilde{U}, \tilde{U} \subset U_j, |\tilde{U}| = m_j$. With the same argument as above, the following upper bound of $\chi_k((m_1, \ldots, r_j, \ldots, m_d))$ results:

$$\chi_k^{(j)} := \Gamma_j^{-1} \sum_{\substack{U \subseteq N \\ \forall l \neq j : |U_l| = m_l \\ |U_j| = r_j}} \sum_{i \in U_k} \sum_{\substack{\tilde{U} \subset U_j \\ |\tilde{U}| = m_j}} \frac{x_i((U \backslash U_j) \cup \tilde{U})}{\binom{r_j}{m_j}}\ . \quad (4)$$

We now give a lemma on the relationship between the average cost-shares and their bounds.

Lemma 2. *$a\chi_j = \chi_j^{(j)}$ and $\chi_k = \chi_k^{(j)}$ for $j \in [d]$ and $k \in [d], k \neq j$.*

Proof. We first look at χ_j and $\chi_j^{(j)}, j \in [d]$. Observe, that both sums are over the same subsets of N_l with m_l elements for $l \in [d] \backslash \{j\}$. It therefore suffices to consider both sums for fixed subsets $U_l \subset N_l, |U_l| = m_l, l \in [d] \backslash \{j\}$ only. Let $U = \cup_{l \in [d] \backslash \{j\}} U_l$. Define:

$$\tilde{\chi}_j := \Gamma^{-1} \sum_{\substack{\tilde{U} \subset N_j \\ |\tilde{U}|=m_j}} \sum_{i \in \tilde{U}} x_i(U \cup \tilde{U}) \text{ and} \tag{5}$$

$$\tilde{\chi}_j^{(j)} := \Gamma_j^{-1} \binom{r_j - 1}{m_j - 1}^{-1} \sum_{\substack{U' \subseteq N_j \\ |U'|=r_j}} \sum_{i \in U'} \sum_{\substack{\tilde{U} \subset U' \setminus \{i\} \\ |\tilde{U}|=m_j-1}} x_i(U \cup \{i\} \cup \tilde{U}) \,. \tag{6}$$

$\tilde{\chi}_j$ and $\tilde{\chi}_j^{(j)}$ are related to each other the same way than χ_j and $\chi_j^{(j)}$. Now,

$$\tilde{\chi}_j^{(j)} = \Gamma_j^{-1} \tbinom{r_j-1}{m_j-1}^{-1} \textstyle\sum_{\substack{U' \subset N_j \\ |U'|=r_j}} \sum_{\substack{\tilde{U} \subset U' \\ |\tilde{U}|=m_j}} \sum_{i \in \tilde{U}} x_i(U \cup \tilde{U}) \tag{7}$$

$$= \Gamma_j^{-1} \tbinom{r_j-1}{m_j-1}^{-1} \tbinom{n_j-m_j}{r_j-m_j} \textstyle\sum_{\substack{\tilde{U} \subset N_j \\ |\tilde{U}|=m_j}} \sum_{i \in \tilde{U}} x_i(U \cup \tilde{U}) \,. \tag{8}$$

Equation (7) is a simple combinatorial observation. To obtain (8), we investigate how often each subset of N_j with m_j elements occurs. For each subset $\tilde{U}$ with m_j elements, to determine a superset $U' \supset \tilde{U}$ with r_j elements, we have $\binom{n_j-m_j}{r_j-m_j}$ possibilities. Combining Equations (5) and (8) we get:

$$\tilde{\chi}_j^{(j)} = \Gamma_j^{-1} \Gamma \binom{r_j - 1}{m_j - 1}^{-1} \binom{n_j - m_j}{r_j - m_j} \tilde{\chi}_j = \frac{r_j}{m_j} \tilde{\chi}_j = a \tilde{\chi}_j \,. \tag{9}$$

Therefore, $a\chi_j = \chi_j^{(j)}$ for $j \in [d]$. With similar argumentation, we can conclude that $\chi_k = \chi_k^{(j)}$ for $j \in [d]$ and $k \in [d] \setminus \{j\}$. □

Budget-Balance. Let us now assume, that $x(\cdot)$ is not only cross-monotonic but also α-budget-balanced. We have seen that the optimal cost for all instances of $(m_1, \ldots, m_d)$ and therefore the average optimal cost is one. With the same argument, the average optimal cost of all instances in class $(m_1, \ldots, r_j, \ldots, m_d)$ for $j \in [d]$ is a. Then we can conclude:

$$\sum_{k=1}^{d} \chi_k \le 1 \text{ and} \sum_{k=1}^{j-1} \chi_k + a\chi_j + \sum_{k=j+1}^{d} \chi_k \ge a\alpha \; \forall j \in [d] \,. \tag{10}$$

Summation of these equations yields $\alpha \le \frac{a-1+d}{da}$. For every $\epsilon > 0$ and a sufficient large a, this results in $\alpha \le 1/(d+\epsilon)$. Note, that it suffices to consider the optimal cost instead of the LPT cost in Equation (10). If an α-fraction of the optimal cost cannot be recovered, in particular it cannot be recovered for a non-optimal cost. □

4 The Core and Other Solution Concepts

Since an α-budget-balanced cross-monotonic cost-sharing method is in the α-core, Theorem 5 tells us that the cost-sharing methods defined in the proofs of Theorems 1 and 2 yield the best approximation factor possible. Theorem 6 provides us with a cost-sharing method in the $(m+1)/(2m)$-core for the scheduling problem with arbitrary jobs and related machines.

Theorem 5. *For the scheduling problem with identical jobs and machines, there is no cost-sharing method in the α-core for $\alpha > (m+1)/(2m)$.*

Proof. We show that for $\alpha > (m+1)/(2m)$, there is no cost-allocation function in the α-core for the set U with $|U| = m+1$. Let $U' \subset U, |U'| = m$. Assume, there is a cost-allocation function $\xi(\cdot) : N \rightarrow \mathbb{R}$ for the set U in the α-core. Since we have that $\sum_{i \in U'} \xi(i) \leq \mathit{opt}(U') = 1$, there is an agent $k \in U'$ with $\xi(k) \leq 1/m$. Then,

$$\sum_{i \in U} \xi(i) = \xi(k) + \sum_{i \in U \backslash \{k\}} \xi(i) \leq 1/m + \mathit{opt}(U \backslash \{k\}) = 1/m + 1 \, . \tag{11}$$

From $2\alpha \leq \sum_{i \in U} \xi(i) \leq (1+m)/m$, we can conclude that $\alpha \leq (m+1)/(2m)$. □

Theorem 6. *There is a cost-sharing method in the $(m+1)/(2m)$-core of the scheduling problem with arbitrary jobs and related machines computable in time $O(n \log n)$.*

Proof. Let $U \subseteq N$. Let $\hat{U} \subset U$ be the set of jobs, that LPT assigns until the makespan is reached and let $\tau = |m(\hat{U})|$. Furthermore, we denote by $m_{opt}(U)$ the machines that an optimal assignment uses to assign the set U.

To define the cost-sharing method, we look at two different cases. In the first case $\tau < |\hat{U}|$, i.e. if the makespan first occurs, there is at least one machine that is assigned more than one job. Then, define $x_i(U) = w_i / S(m(\hat{U}))$ for all $i \in \hat{U}$ and $x_i(U) = 0$ for all $i \notin \hat{U}$. In the second case $\tau = |\hat{U}|$, i.e. if the makespan first occurs, LPT has assigned at most one job to each machine. Let $\tau \geq 3$. We will omit the proof of the subcase $\tau \in \{1, 2\}$ due to space restrictions . We define $A(U) = S(m(\hat{U}))\mathit{opt}(U) - W(\hat{U})$. Let $x_i(U) = 0$ for $i \notin \hat{U}$. For $i \in \hat{U}$, let

$$x_i(U) = \begin{cases} \frac{w_i}{S(m(\hat{U}))} & \text{if } A(U) < \frac{\tau-1}{\tau+1} W(\hat{U}) \\ \frac{w_i}{S(m(\hat{U})) - s_\tau} & \text{otherwise} \, . \end{cases}$$

First observe, that LPT determines the running time. We have to show for the given cost-sharing method $x(\cdot)$ that $x(\cdot, U)$ is in the α-core for all $U \subseteq N$. We start with the first case in which $\tau < |\hat{U}|$. $x(U)$ is smaller than $\mathit{opt}(U)$, since by Lemma 1 it holds, that $x(U) = W(\hat{U})/S(m(\hat{U})) \leq \mathit{opt}(\hat{U}) \leq \mathit{opt}(U)$. Lemma 1 also provides the approximation factor. Next, we show the core condition. Let $U' \subseteq U$. From the proof of Lemma 1 we can conclude, that $m(\hat{U}) \supseteq m_{opt}(\hat{U})$. Therefore,

$$\sum_{i \in U'} x_i(U) = \frac{W(U' \cap \hat{U})}{S(m(\hat{U}))} \leq \frac{W(U' \cap \hat{U})}{S(m_{opt}(\hat{U}))} \leq \frac{W(U' \cap \hat{U})}{S(m_{opt}(U' \cap \hat{U}))} \leq \mathit{opt}(U') \, . \tag{12}$$

Consider the second case, $\tau = |\hat{U}|$. Then, $\mathit{lpt}(U) = \mathit{opt}(U) = w_\tau / s_\tau$. Due to space restrictions, we omit the case $A(U) < (\tau - 1)/(\tau + 1) W(\hat{U})$ and for the remaining case only show the core condition. If $A(U) \geq (\tau - 1)/(\tau + 1) W(\hat{U})$, then $x(U) = W(\hat{U})/(S(m(\hat{U})) - s_\tau)$.

Let $U' \subseteq U$. If $U' = \hat{U}$, then $\sum_{i \in U'} x_i(U) = x(U) \leq \mathit{opt}(U) = \mathit{opt}(U')$. Otherwise, $(U' \cap \hat{U}) \subset \hat{U}$. Since $\mathit{lpt}(U) = \mathit{opt}(U)$, it holds that $m(\hat{U}) = m_{opt}(\hat{U})$ and

$m_{opt}(U' \cap \hat{U}) \subseteq m_{opt}(\hat{U}) \setminus \{\tau\}$, since an optimal assignment for a proper subset of $\hat{U}$ does not use the machine τ anymore. Thus,

$$\sum_{i \in U'} x_i(U) = \frac{W(U' \cap \hat{U})}{S(m_{opt}(\hat{U})) - s_\tau} \leq \frac{W(U' \cap \hat{U})}{S(m_{opt}(U' \cap \hat{U}))} \leq opt(U') \, . \qquad (13)$$

□

Finally, we state Theorem 7, whose proof is available in the full version.

Theorem 7. *There is a* $3/5$*-budget-balanced strategyproof cost-sharing mechanism for the scheduling game with arbitrary jobs and related machines. It is* $3m/(4m-1)$*-budget-balanced for the scheduling game with identical jobs and related machines and* 1*-budget-balanced for identical jobs and identical machines. Its running time is the running time of* LPT.

Acknowledgements. We would like to thank Rainer Feldmann, Martin Gairing, and Karsten Tiemann for many helpful discussions.

References

1. A. Archer, J. Feigenbaum, A. Krishnamurthy, and R. Sami. Approximation and collusion in multicast cost sharing. *Games and Economic Behaviour*, 47:36–71, 2004.
2. A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. *Proceedings of the 42th IEEE Symposium on Foundations of Computer Science*, pages 482–491, 2001.
3. L. Beccetti, J. Könemann, S. Leonardi, and M. Pál. Sharing the cost more efficiently: improved approximation for multicommodity rent-or-buy. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 375–384, 2005.
4. A. Czumaj. Selfish Routing on the Internet. *Chapter 42 in Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, 2004.
5. N. Devanur, M. Mihail, and V. Vazirani. Strategyproof cost sharing mechanisms for set cover and facility location problems. In *Proceedings of ACM Conference on Electronic Commerce*, pages 108–114, 2003.
6. J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker. Hardness results for multicast cost sharing. *Theoretical Computer Science*, 304(1-3):215–236, 2003.
7. J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63:21–41, 2001.
8. D. Friesen. Tighter bounds for the multifit processor scheduling algorithm. *SIAM Journal on Computing*, 13(1):170–181, 1984.
9. D. Friesen. Tighter bounds for lpt scheduling on uniform processors. *SIAM Journal on Computing*, 16(3):554–560, 1987.
10. D. Friesen and M. Langston. Bounds for multifit scheduling on uniform processors. *SIAM Journal on Computing*, 12(1):60–70, 1983.
11. M. Gairing, T. Lücking, B. Monien, and K. Tiemann. Nash Equilibria, the Price of Anarchy and the Fully Mixed Nash Equilibrium Conjecture. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *LNCS*, pages 51–65, 2005.
12. R. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 17(2):416–429, 1969.

13. A. Gupta, A. Srinivasan, and E. Tardos. Cost-Sharing Mechanisms for Network Design. *Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 3122:139–152, 2004.
14. D. Hochbaum and D. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34(1):144–162, 1987.
15. D. Hochbaum and D. Shmoys. A polynomial approximation scheme for scheuduling on uniform processors: using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.
16. E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the Association for Computing Machinery*, 23(2):317–327, 1976.
17. N. Immorlica, M. Mahdian, and V. Mirrokni. Limitations of cross-monotonic cost sharing schemes. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 602–611, 2005.
18. K. Jain and V. Vazirani. Applications of approximate algorithms to cooperative games. In *Proceedings of the 33th Annual ACM Symposium on Theory of Computing*, pages 364–372, 2001.
19. K. Kent and D. Skorin-Kapov. Population monotonic cost allocation on msts. In *Operational Research Proceedings KOI*, pages 43–48, 1996.
20. J. Könemann, S. Leonardi, and G. Schäfer. A group-strategyproof mechanism for steiner forests. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 612–619, 2005.
21. J. Könemann, S. Leonardi, G. Schäfer, and S. van Zwam. From primal-dual to cost shares and back: a stronger LP relaxation for the steiner forest problem. In *Proceedings of the 32th Int. Colloquium on Automata, Languages, and Programming*, pages 930–942, 2005.
22. J. K. Lenstra, D. B. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science (FOCS'87)*, pages 217–224, 1987.
23. S. Leonardi and G. Schäfer. Cross-monotonic cost-sharing methods for connected facility location games. In *ACM Conference on Electronic Commerce*, pages 224–243, 2004.
24. D. Mishra and B. Rangarajan. Cost sharing in a job scheduling problem using the shapley value. In *Proceedings of the 6th ACM Conference on Electronic Commerce*, pages 232–239, 2005.
25. H. Moulin and S. Shenker. Strategyproof sharing of submodular costs: budget balance versus efficiency. *Economic Theory*, 18:511–533, 2001.
26. N. Nisan and A. Ronen. Algorithmic Mechanism Design. *Games and Economic Behaviour*, 35:166–196, 2001. Extended abstract appeard at STOC'99.
27. M. Pál and E. Tardos. Group strategyproof mechanisms via primal-dual algorithms. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 584–593, 2003.
28. P. Penna and C. Ventre. The Algorithmic Structure of Group Strategyproof Budget-Balanced Cost-Sharing Mechanisms. In *Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science, to appear*, 2006.
29. L. S. Shapley. On balanced sets and cores. *Naval Research Logistics Quarterly*, 14:453–460, 1967.
30. E. V. Shchepin and N. Vakhania. An optimal rounding gives a better approximation for scheduling unrelated machines. *Operations Research Letters*, 33:127–133, 2005.