New Efficiency Results for Makespan Cost Sharing ⋆

Yvonne Bleischwitz [a], Florian Schoppmann [a,1]

[a] *University of Paderborn, Department of Computer Science*
Fürstenallee 11, 33102 Paderborn, Germany

Abstract

In the context of scheduling, we study social cost efficiency for a cost-sharing problem in which the service provider's cost is determined by the makespan of the served agents' jobs. For identical machines, we give surprisingly simple cross-monotonic cost-sharing methods that achieve the essentially best efficiency Moulin mechanisms can guarantee. Still, our methods match the budget-balance of previous (yet rather intricate) results. Subsequently, we give a generalization for arbitrary jobs.

Finally, we return to identical jobs in order to perform a fine-grained analysis. We show that the universal worst-case efficiency bounds from [8] are overly pessimistic.

Key words: Scheduling, Mechanism Design, Cost-Sharing, Cross-Monotonicity

1 Introduction and Model

For $n \in \mathbb{N}$, we use the abbreviations $[n] := \{1, \dots, n\}$ and $H_n := \sum_{i=1}^{n} \frac{1}{i}$.

Cost-Sharing. Consider n agents interested in some common service offered by a service provider. Each agent $i \in [n]$ submits a *bid* $b_i \in \mathbb{R}_{\geq 0}$ indicating the amount of money she is willing to pay for the service. Only on the basis of these bids $\boldsymbol{b} := (b_1, \dots, b_n)$, the provider determines a set $Q(\boldsymbol{b}) \subseteq [n]$ of served agents and their payments $x(\boldsymbol{b}) \in \mathbb{R}^n_{\geq 0}$, referred to as *cost shares.* This is accomplished via a commonly-known *cost-sharing mechanism* $(Q \times x) : \mathbb{R}^n_{\geq 0} \to 2^{[n]} \times \mathbb{R}^n_{\geq 0}$. The agents' bids can not be verified to indeed reflect their *true valuations* $v_i \in \mathbb{R}_{\geq 0}$, since agents are assumed to act selfishly such to only maximize their quasi-linear utilities defined as $u_i(\boldsymbol{b}) := v_i - x_i(\boldsymbol{b})$ if $i \in Q(\boldsymbol{b})$ and 0 otherwise. Thus, a major challenge is to create incentives for truthful bidding, even if agents may collude. A mechanism $(Q \times x)$ is *group-strategyproof* (GSP)

⋆ This work was partially supported by the IST Program of the European Union under contract number IST-15964 (AEOLUS).

1 International Graduate School of Dynamic Intelligent Systems

if for every true valuation vector $\boldsymbol{v} \in \mathbb{R}^n_{\geq 0}$ and any coalition $K \subseteq [n]$ there is no bid vector $\boldsymbol{b} \in \mathbb{R}^n_{\geq 0}$, with $b_i = v_i$ for $i \notin K$, such that $u_i(\boldsymbol{b}) \geq u_i(\boldsymbol{v})$ for all $i \in K$ and $u_i(\boldsymbol{b}) > u_i(\boldsymbol{v})$ for at least one $i \in K$.
The only general technique to design GSP cost-sharing mechanisms is due to Moulin [7]. The key ingredient of a Moulin mechanism is a *cross-monotonic* cost-sharing method $\xi : 2^{[n]} \to \mathbb{R}^n_{\geq 0}$ such that for all $A, B \subseteq [n]$ and $i \in A$: $\xi_i(A) \geq \xi_i(A \cup B)$. Given a bid vector $\boldsymbol{b} \in \mathbb{R}^n_{\geq 0}$, $\textsc{moulin}_\xi := (Q \times x)_\xi$ can be computed by a very simple algorithm: Initially, let $Q(\boldsymbol{b}) := [n]$. Then repeatedly eliminate agents from $Q(\boldsymbol{b})$ whose bids are below their current cost shares ($b_i < \xi_i(Q(\boldsymbol{b}))$) until all remaining agents can afford their cost-shares. Let $x(\boldsymbol{b}) := \xi(Q(\boldsymbol{b}))$. If ξ is cross-monotonic then $\textsc{moulin}_\xi$ is GSP [7].
We now focus on the *service cost* $C : 2^{[n]} \to \mathbb{R}_{\geq 0}$, mapping each subset of agents to the cost of serving them. Typically, costs stem from solutions to a combinatorial minimization problem and are defined only implicitely.
Computing an optimal solution with cost C may be computationally intractable. Thus, the service provider resorts to approximate solutions with cost C'. At least a β-fraction of the cost incurred by serving the selected agents should be recovered, while at the same time offering competitive prices. Formally, a mechanism is *β-budget-balanced* (β-BB, $\beta \leq 1$) if for all $\boldsymbol{b} \in \mathbb{R}^n_{\geq 0}$: $\beta \cdot C'(Q(\boldsymbol{b})) \leq \sum_{i \in Q(\boldsymbol{b})} x_i(\boldsymbol{b}) \leq C(Q(\boldsymbol{b}))$.
An economic objective is to design *efficient* mechanisms that trade off the incurred cost and the valuations of the rejected agents as good as possible: The *social cost* of a set $A \subseteq [n]$ with respect to C' and C is $\mathrm{SC}'_{\boldsymbol{v}}(A) := C'(A) + \sum_{i \in [n] \setminus A} v_i$ and $\mathrm{SC}_{\boldsymbol{v}}(A) := C(A) + \sum_{i \in [n] \setminus A} v_i$, respectively. A mechanism is *γ-efficient* (γ-EFF, $\gamma \geq 1$) if for any valuation vector $\boldsymbol{v} \in \mathbb{R}^n_{\geq 0}$ and all $A \subseteq [n]$ it holds that $\mathrm{SC}'_{\boldsymbol{v}}(Q(\boldsymbol{v})) \leq \gamma \cdot \mathrm{SC}_{\boldsymbol{v}}(A)$.

The Scheduling Problem. We consider as underlying combinatorial minimization problem to schedule a subset of n jobs on m related parallel machines. The service provider administrates the machines and each agent owns exactly one job with processing time $p_i \in \mathbb{N}$. We let $d := |\{p_1, \ldots, p_n\}|$. The speed of machine $j \in [m]$ is denoted as $s_j \in \mathbb{N}$, where $S := \sum_{j \in [m]} s_j$. We say that jobs (machines) are *identical* if $p_i = 1$ for all $i \in [n]$ ($s_j = 1$ for all $j \in [m]$).
For a given assignment, let δ_j be the sum of the processing times of the jobs assigned to machine j. Then, the completion time of a job assigned to j is δ_j / s_j and the makespan is $\max_{j \in [m]} \delta_j / s_j$. We denote the makespan of an optimal assignment for $A \subseteq [n]$ by $C(A)$. For identical jobs, the optimal makespan only depends on the number of jobs. We therefore write $c(|A|) := C(A)$.
Certainly, our model may not very well reflect real-world applications. Agents might not only be interested in their job being processed, but also in its completion time. Furthermore, the makespan as the completion time of the whole system may not reflect the provider's cost properly. However, it has been shown that for some other more natural cost functions like weighted completion time, cross-monotonic cost-sharing methods are impracticable [2]. We thus consider our work to be a first basic step for cost-sharing scheduling scenarios.

Previous Work. Efficiency of cost-sharing mechanisms is traditionally defined as the maximization of the *social welfare*, which is the sum of the receivers' valuations minus the provider's cost. However, results by economists in the 1970's (e.g., [6]) imply that it is essentially impossible for a GSP mechanism to be both 1-BB and social welfare efficient. Furthermore, Feigenbaum et al. [3] showed that even a constant factor approximation of both objectives is impossible. Albeit this negative result, Roughgarden and Sundararajan [8] proved that for the minimization of the social cost, a homogeneous formulation of efficiency, a simultaneous approximation is possible and can be investigated by the *summability* of a cost-sharing method. A cost-sharing method ξ is α-*summable* (α-*SUM*) if for all $A \subseteq [n]$ and every order $a_1, \ldots, a_{|A|}$ of A with $A^i := \{a_1, \ldots, a_i\}$ it holds that $\sum_{i=1}^{|A|} \xi_{a_i}(A^i) \leq \alpha \cdot C(A)$.

Theorem 1 ([8]) *Let ξ be a cross-monotonic cost-sharing method. Let $\alpha \geq 0$ be the smallest and $\beta \leq 1$ be the largest numbers such that ξ is α-SUM and β-BB. Then* MOULIN_ξ *is $(\alpha + \frac{1}{\beta})$-EFF and no better than* $\max\{\alpha, \frac{1}{\beta}\}$*-EFF.*

Brenner and Schäfer [2] introduce a general lower bound on the summability:

Theorem 2 ([2]) *Let $\xi : 2^{[n]} \to \mathbb{R}^n_{\geq 0}$ be a β-BB cost-sharing method. If there are constants $p, q \geq 1$ and a set $A \subseteq [n]$ with $|A| \geq \frac{n}{p}$ such that $C(B) \geq \frac{C(A)}{q}$ for all $\emptyset \neq B \subseteq A$, then ξ is not α-SUM for any $\alpha < \frac{H_{\lceil n/p \rceil} \cdot \beta}{q}$.*

Cross-monotonic cost-sharing methods for scheduling on parallel machines under makespan minimization were first presented by Bleischwitz and Monien [1]. Their methods always guarantee a coverage of $\frac{1}{2d}$ of the makespan cost of the LPT algorithm [5], and even a coverage of $\frac{m+1}{2m}$ if either jobs or machines are identical. On the other hand, they show that cross-monotonic methods cannot be better than $\frac{1}{d}$-BB in general and not better than $\frac{m+1}{2m}$-BB for identical jobs or machines. Brenner and Schäfer [2] modified the cost-sharing method of [1] for *identical* machines such that its summability improved from $\frac{n}{2}$ to H_n.

Our Contribution. In Section 3, we complement the work of [2] by looking at *related* machines. We give polynomially computable cross-monotonic cost-sharing methods for makespan scheduling that are $\frac{m+1}{2m}$-BB for identical jobs and $\frac{1}{2d}$-BB for arbitrary jobs, matching the bounds from [1]. The strength of our approach lies in the simplicity of the new methods, allowing for first summability results. For identical jobs, our method is H_n-SUM. We conclude from Theorem 2 that this is the best that generally can be achieved. For arbitrary jobs, we show a tight approximate summability for our method.
Although in general, a Moulin mechanism $\text{MOULIN}_{\xi^{\text{IR}}}$ for our cost-sharing method ξ^{IR} for identical jobs (as defined in Section 3) is no better than $\max\{\frac{2m}{m+1}, H_n\}$-EFF [8], we show better tight approximations of the social cost for many cases in Section 4. Roughly speaking, we show that $\text{MOULIN}_{\xi^{\text{IR}}}$ is no worse than 2-EFF if there is sufficiently large demand.

2 LPT and Properties of Minimum Makespan Solutions

Since computing the minimum makespan is NP-hard in general, we assume that the service provider employs Graham's LPT algorithm [5] to compute an allocation. LPT processes the jobs by decreasing processing times and assigns each job to a machine on which it has the smallest completion time (taking into account the jobs that have been assigned already). Ties are resolved in a deterministic way. For a set $A \subseteq [n]$ we use $\mathrm{LPT}(A)$ to denote the makespan resulting from LPT. If all jobs have the same processing time, LPT computes an optimal solution. For arbitrary jobs, LPT achieves an approximation ratio of 5/3 [4]. Its running time is $O(n \log n + nm)$ in general and $O(n \log n + n \log m)$ for identical jobs using a priority queue for job placement.

Lemma 3 *The optimum makespan costs C and c satisfy the properties below:*

(P1) For all $A, B \subseteq [n]$: $C(A \cup B) \leq C(A) + C(B)$
(P2) For all $k, l \in [n]$: $c(k+l) \leq c(k) + c(l)$, especially $\frac{c(2k)}{2k} \leq \frac{c(k)}{k}$
(P3) For all $k \in [n]$: $c(k) \geq \frac{k}{S}$
(P4) For all $k \in [n]$: $c(k \cdot S) = k$
(P5) For all $k \in [n]$ with $k \geq m+1$: $c(k) \leq \frac{k}{S} \cdot \frac{2m}{m+1}$
(P6) For all $a \in [n]$: $\min_{k \in [a]} \frac{c(k)}{k} \geq \frac{1}{S}$
(P7) For all $a \in [n]$: If $l \in [a]$ is maximum with $\frac{c(l)}{l} = \min_{k \in [a]} \frac{c(k)}{k}$, then $c(a) \leq 2c(l)$.

Proof: (P1) and (P2) hold by subadditivity of optimal makespan costs, (P3) and (P4) are trivial observations. To prove (P5), let q be the number of machines on which the makespan occurs (makespan machines). We change this assignment by moving one job from a makespan machine if it can achieve a strictly smaller completion time on another machine. This can be done at most $q-1$ times; otherwise, the initial assignment was not optimal. For a makespan machine j, it is $\delta_j = c(k) \cdot s_j$ and $\delta_{j'} + 1 \geq c(k) \cdot s_{j'}$ $\forall j' \in [m] \backslash \{j\}$. Summation yields $k + m - 1 \geq c(k) \cdot S$. Consequently, $k + (m-1) \cdot \frac{k}{m+1} \geq c(k) \cdot S$ yields the bound. To prove (6), let $\min_{k \in [a]} \frac{c(k)}{k} = \frac{c(l)}{l}$ for an $l \in [a]$. By (P3), $\frac{c(l)}{l} = \frac{1}{S}$. We continue to prove (P7). By (P2), $\frac{c(2l)}{2l} \leq \frac{c(l)}{l}$. If now $a \geq 2l$, obviously l cannot be maximum. Thus, $a < 2l$. By (P2), $c(a) \leq c(2l) \leq 2c(l)$. □

3 Cost-Sharing Methods ξ^{IR} for Identical Jobs on Related Machines and ξ^{AR} for Arbitrary Jobs on Related Machines

Definition 4 *For each set $A \subseteq [n]$, define the method $\xi^{\mathrm{IR}} : 2^{[n]} \to \mathbb{R}^n_{\geq 0}$ by $\xi_i^{\mathrm{IR}}(A) := \min_{k \in [|A|]} \frac{c(k)}{k}$ if $i \in A$ and $\xi_i^{\mathrm{IR}}(A) := 0$ otherwise.*

Theorem 5 *ξ^{IR} is cross-monotonic, $\frac{m+1}{2m}$-BB, and H_n-SUM. For any $A \subseteq [n]$, the cost shares $\{\xi_i^{\text{IR}}(A)\}_{i\in A}$ can be computed in time $O(n \log n + n \log m)$.*

Proof: Cross-monotonicity is obvious. Fix $A \subseteq [n]$. We start proving $\frac{m+1}{2m}$-BB. Directly from the definition, it follows that $\sum_{i\in A} \xi_i^{\text{IR}}(A) \leq c(|A|)$. From now on, we assume strict inequality; otherwise, we have 1-BB.
First assume $|A| \leq m$. Let $l < |A|$ be maximum with $\frac{c(l)}{l} = min_{k\in[|A|]} \frac{c(k)}{k}$. (We do not consider $l = |A|$ since we assume strict inequality above, i.e., $\sum_{i\in A} \xi_i^{\text{IR}}(A) < c(|A|)$.) By (P7) and $m \geq l+1$, we get the following bound: $\sum_{i\in A} \xi_i^{\text{IR}}(A) = \frac{|A|}{l} \cdot c(l) \geq \frac{l+1}{2l} \cdot c(|A|) \geq \frac{m}{2m-2} \cdot c(|A|) \geq \frac{m+1}{2m} \cdot c(|A|)$.
Now assume that $|A| \geq m+1$. By (P5), $c(|A|) \leq \frac{|A|}{S} \cdot \frac{2m}{m+1}$. With (P6), we get $\sum_{i\in A} \xi_i^{\text{IR}}(A) \geq \frac{|A|}{S} \geq \frac{m+1}{2m} \cdot c(|A|)$.
For summability, let $a_1, \ldots, a_{|A|}$ be an arbitrary order of A, and let A^i denote the set of the first i elements. Then, $\sum_{i=1}^{|A|} \xi_{a_i}^{\text{IR}}(A^i) \leq \sum_{i=1}^{|A|} \frac{c(i)}{i} \leq H_n \cdot c(|A|)$.

Since LPT on $|A|$ identical jobs simultaneously computes $c(1), \ldots, c(|A|)$, the time to compute $\min_{k\in[|A|]} \frac{c(k)}{k} = \xi_i^{\text{IR}}(A)$ for all $i \in A$ is $O(n \log n + n \log m)$. □

The instance with n identical jobs and n identical machines meets the conditions of Theorem 2 with $p = q = 1$. Hence, by Theorem 1, $O(1)$-BB Moulin mechanisms can in general be no better than $\Theta(\log n)$-EFF. In this sense, H_n-SUM (and H_n-EFF) is the best achievable.

We now apply our idea for identical jobs to arbitrary jobs. As a consequence, budget-balance impairs. However, we achieve $\frac{1}{2d}$-BB while the best that can in general be achieved with cross-monotonic methods is $\frac{1}{d}$-BB [1].

For $A \subseteq [n]$, let $\mathcal{P}(A) := \{p_i \mid i \in A\}$ be the set of different processing times of the jobs in A, and let $A(y) := \{i \in A \mid p_i = y\}$ denote the set of all agents with processing time y.

Definition 6 *For each set $A \subseteq [n]$ define the method $\xi^{\text{AR}} : 2^{[n]} \to \mathbb{R}^n_{\geq 0}$ by $\xi_i^{\text{AR}}(A) := \frac{p_i}{|\mathcal{P}(A)|} \cdot \min_{k\in[|A(p_i)|]} \frac{c(k)}{k}$ if $i \in A$ and $\xi_i^{\text{AR}}(A) := 0$ otherwise.*

Let $\mathcal{P}(A) = \{y_1, \ldots, y_{|\mathcal{P}(A)|}\}$. For $k \in [|\mathcal{P}(A)|]$, let τ_k be the cardinality of the k-th largest set in $\{A(y_l)\}_{l\in[|\mathcal{P}(A)|]}$. Define $\mathcal{H}(A) := \sum_{k=1}^{|\mathcal{P}(A)|} \frac{H_{\tau_k}}{k}$.

Theorem 7 *ξ^{AR} is cross-monotonic, $\frac{1}{2|\mathcal{P}([n])|}$-BB, and $\mathcal{H}([n])$-summable. For any $A \subseteq [n]$, $\{\xi_i^{\text{AR}}(A)\}_{i\in A}$ can be computed in time $O(n \log n + n \log m)$.*

Proof: Cross-monotonicity is obvious. Fix $A \subseteq [n]$. To show $\frac{1}{2|\mathcal{P}([n])|}$-BB, we first bound the sum of the cost shares from above by $C(A)$. We have that

$$\sum_{i\in A} \xi_i^{\text{AR}}(A) = \sum_{y\in\mathcal{P}(A)} \frac{y \cdot |A(y)|}{|\mathcal{P}(A)|} \cdot \min_{k\in[|A(y)|]} \frac{c(k)}{k} \leq \frac{1}{|\mathcal{P}(A)|} \sum_{y\in\mathcal{P}(A)} y \cdot c(|A(y)|) \leq C(A),$$

where the last inequality follows from $y \cdot c(|A(y)|) = C(A(y)) \leq C(A)$. Next we give the lower bound with respect to LPT. Applying (P7), we get

$$\sum_{i\in A} \xi_i^{\text{AR}}(A) = \frac{1}{|\mathcal{P}(A)|} \sum_{y\in\mathcal{P}(A)} |A(y)| \cdot y \cdot \min_{k\in[A(y)]} \frac{c(k)}{k} \geq \frac{1}{2|\mathcal{P}(A)|} \sum_{y\in\mathcal{P}(A)} y \cdot c(|A(y)|)$$
$$= \frac{1}{2|\mathcal{P}(A)|} \sum_{y\in\mathcal{P}(A)} C(A(y)) = \frac{1}{2|\mathcal{P}(A)|} \sum_{y\in\mathcal{P}(A)} \text{LPT}(A(y)) \geq \frac{\text{LPT}(A)}{2|\mathcal{P}(A)|} \ .$$

The last inequality is due to the fact that adding all LPT costs for the sets $A(y)$ can never be smaller than the LPT cost of the whole set A.

For summability, let $a_1, \ldots, a_{|A|}$ be an arbitrary order of A, and let A^i denote the set of the first i elements. Furthermore, let $\{y_1, \ldots, y_{|\mathcal{P}(A)|}\} = \mathcal{P}(A)$, in the order in which the processing times first occur in $a_1, \ldots, a_{|A|}$. Then,

$$\sum_{i=1}^{|A|} \xi_{a_i}^{\text{AR}}(A^i) = \sum_{i=1}^{|A|} \frac{p_{a_i}}{|\mathcal{P}(A^i)|} \cdot \min_{k\in[|A^i(p_{a_i})|]} \frac{c(k)}{k} \leq \sum_{l=1}^{|\mathcal{P}(A)|} \frac{y_l}{l} \sum_{i=1}^{|A(y_l)|} \min_{k\in[i]} \frac{c(k)}{k} \ .$$

This estimate holds, since the middle term is maximized when jobs of the same processing time are contiguous in the order. This is due to the term $|\mathcal{P}(A^i)|$, which in that way is always the smallest possible number. The remaining computation is straightforward:

$$\sum_{i=1}^{|A|} \xi_{a_i}^{\text{AR}}(A^i) \leq \sum_{l=1}^{|\mathcal{P}(A)|} \frac{1}{l} \sum_{i=1}^{|A(y_l)|} \frac{y_l \cdot c(i)}{i} \leq \sum_{l=1}^{|\mathcal{P}(A)|} \frac{H_{|A(y_l)|}}{l} \cdot C(A) \leq \mathcal{H}(A) \cdot C(A) \ ,$$

where we utilize that for all $l \in [|\mathcal{P}(A)|]$ and all $i \in [|A(y_l)|]$ it holds that $y_l \cdot c(i) \leq y_l \cdot c(|A(y_l)|) = C(A(y_l)) \leq C(A)$.

The time to compute $\{\xi_i^{\text{AR}}(A)\}_{i\in A}$ is determined by the computation time of $c(1), \ldots, c(|A|)$ (for *identical* jobs). We have already seen for ξ^{IR} that this can be accomplished in time $O(n \log n + n \log m)$. □

If $|\mathcal{P}([n])| \in \{1, n\}$, then we achieve H_n-summability. In the following, we show that the summability bound $\mathcal{H}([n])$ is tight for ξ^{AR}.

The following instance meets the bound from Theorem 7: Let $d \in \mathbb{N}_{>0}$ be the number of different processing times. For $k \in [d]$ and $q \in \mathbb{N}_{>0}$, let there be n_k jobs with processing time $y_k = q + k - 1$. Let $n = \sum_{k=1}^{d} n_k$. It holds that $|\mathcal{P}([n])| = d$. Let there be $m = n$ machines of speed q. We assume that processing times are increasing, i.e., $p_1 \leq \cdots \leq p_n$. Then, $[i]$ corresponds to the set of the first i elements of $[n]$. As $q \to \infty$, we get $C([n]) = 1$ and

$$\sum_{i=1}^{n} \xi_i^{\text{AR}}([i]) = 1 \cdot \left(\sum_{j=1}^{n_1} \frac{q}{j \cdot q} \right) + \ldots + \frac{1}{d} \cdot \left(\sum_{j=1}^{n_d} \frac{q + d - 1}{j \cdot q} \right) = \sum_{k=1}^{d} \frac{H_{n_k}}{k} \cdot C([n]) \ .$$

4 Efficiency Guarantees of MOULIN$_{\xi^{\mathrm{IR}}}$

Although in general, MOULIN$_{\xi^{\mathrm{IR}}}$ is not better than $\max\{\frac{2m}{m+1}, H_n\}$-EFF [8], we show better approximations for many cases with *identical* jobs. Theorem 8 states that only if MOULIN$_{\xi^{\mathrm{IR}}}$ gives the service to less than S agents, the worst case performance of $1 + H_{\min\{n,S\}}$ may occur.

Theorem 8 *Let $\boldsymbol{v} \in \mathbb{R}^n_{\geq 0}$, $\mu = |Q(\boldsymbol{v})|$ be the number of agents selected by* MOULIN$_{\xi^{\mathrm{IR}}}$ $=: (Q \times x)_{\xi^{\mathrm{IR}}}$ *and σ be the largest cardinality of all sets that minimize the social cost $\mathrm{SC}_{\boldsymbol{v}}$. Then for all $A \subseteq [n]$ it holds that $\mathrm{SC}_{\boldsymbol{v}}(Q(\boldsymbol{v})) \leq \gamma \cdot \mathrm{SC}_{\boldsymbol{v}}(A)$, where the values of γ are given for specific conditions in the table below:*

case	1	2	3	4	5
condition	$\sigma < \mu$ *and* $\mu \geq m+1$	$\sigma < \mu$ *and* $\mu < m+1$	$\sigma > \mu$ *and* $\mu \geq S$	$\sigma > \mu$ *and* $\mu < S$	$\sigma = \mu$
γ	$\frac{2m}{m+1}$	$\frac{2m-2}{m}$	$2 - \frac{\lfloor \mu/S \rfloor}{\lfloor \mu/S \rfloor + 1}$	$1 + H_{\min\{n,S\}}$	1

To discuss these cases, we first give examples which also show that the results from Theorem 8 are tight. For simplicity, we write $(a)^n := (a, \ldots, a) \in \mathbb{R}^n$.

- *Case 1*: Consider m identical machines and $n = m + 1$ jobs. It is $\boldsymbol{x} = (1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{m}, \frac{1}{m})$. Let $\boldsymbol{v} := (\frac{1}{m})^n$. It holds that $\mu = m + 1$ and $\sigma = m$. The social costs are $\mathrm{SC}_{\boldsymbol{v}}([\mu]) = 2$ and $\mathrm{SC}_{\boldsymbol{v}}([\sigma]) = 1 + \frac{1}{m}$. Whereas we have 1-BB for 1 to m agents ($c[i] = 1 \forall i \in [m]$), the cost increase of 1 when moving from m to $m + 1$ agents cannot be recovered by the cross-monotonic cost-shares.
- *Case 2*: Consider $m \geq 2$ machines, where $m - 1$ machines have speed 2 and 1 machine has speed 1. Let there be $n = m$ jobs and let $\boldsymbol{v} := (\frac{1}{2(m-1)})^n$. It is $\boldsymbol{x} = (\frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \ldots, \frac{1}{2(m-1)}, \frac{1}{2(m-1)})$. It holds that $\mu = m$ and $\sigma = m - 1$. The social costs are $\mathrm{SC}_{\boldsymbol{v}}([\mu]) = 1$ and $\mathrm{SC}_{\boldsymbol{v}}([\sigma]) = \frac{1}{2} + \frac{1}{2(m-1)}$. As in case 1, the lack of budget balance in favor of cross-monotonicity causes inefficiency.
- *Case 3*: Consider m identical machines and $n = 2m$ jobs. Furthermore, let $\boldsymbol{v} := (1)^{(m+1)}(\frac{1}{m} - \varepsilon)^{(m-1)}$. It is $\boldsymbol{x} = (1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{m})(\frac{1}{m})^m$. It holds that $\mu = m + 1$, $\sigma = 2m$, $\mathrm{SC}_{\boldsymbol{v}}([\mu]) = 2 + (m - 1)(\frac{1}{m} - \varepsilon)$, and $\mathrm{SC}_{\boldsymbol{v}}([\sigma]) = 2$. Despite 1-BB for $2m$ agents, the fact that some valuations are just below the cost-shares causes efficiency loss. For $m \to \infty$ and $\varepsilon \to 0$ it is $\gamma \to \frac{3}{2}$.
- *Case 4*: Consider m identical machines and $n = m$ jobs, $\boldsymbol{x} = (1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{m})$. Let $\boldsymbol{v} := (1 - \varepsilon, \frac{1}{2} - \varepsilon, \frac{1}{3} - \varepsilon, \ldots, \frac{1}{m} - \varepsilon)$. it holds that $\mu = 0$ and $\sigma = m$. For $\varepsilon \to 0$, the social costs are $\mathrm{SC}_{\boldsymbol{v}}([\mu]) = H_n$ and $\mathrm{SC}_{\boldsymbol{v}}([\sigma]) = 1$. We have the same source of inefficiency as in case 3.

Proof of Theorem 8: The main idea of this proof is to order the agents from 1 to n such that $v_1 \geq \ldots \geq v_n$. Then, $Q(\boldsymbol{v}) = [\mu]$, and $[\sigma]$ is the maximum set of agents that minimizes $\mathrm{SC}_{\boldsymbol{v}}$. In the following, we determine the ratio between

$\mathrm{SC}_{\boldsymbol{v}}([\mu])$ and $\mathrm{SC}_{\boldsymbol{v}}([\sigma])$. MOULIN$_{\xi^{\mathrm{IR}}}$ is obviously 1-EFF for $\mu = \sigma$. We define $\boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$ by $x_k := \min_{l\in[k]} \frac{c(l)}{l}$; x_k denotes $\xi_1^{\mathrm{IR}}([k])$. We frequently use that $v_i \geq x_\mu$ for all $i \in [\mu]$ (because MOULIN$_{\xi^{\mathrm{IR}}}$ has decided to serve these agents).

- *Case 1:* With (P5), $\mathrm{SC}_{\boldsymbol{v}}([\mu]) \leq \frac{\mu}{S} \cdot \frac{2m}{m+1} + \sum_{i=\mu+1}^n v_i$. Applying (P6), we get $\mathrm{SC}_{\boldsymbol{v}}([\sigma]) = c(\sigma) + \sum_{i=\sigma+1}^{\mu} v_i + \sum_{i=\mu+1}^n v_i \geq \frac{\sigma}{S} + \frac{\mu-\sigma}{S} + \sum_{i=\mu+1}^n v_i$.
- *Case 2:* Let $l \in [\mu]$ be maximum with $x_\mu = \frac{c(l)}{l}$. It is $\sigma > 0$, since $\sigma = 0$ together with $\mathrm{SC}_{\boldsymbol{v}}([l]) = c(l) + \sum_{i=l+1}^n v_i = l \cdot x_\mu + \sum_{i=l+1}^n v_i \leq \sum_{i=1}^n v_i = \mathrm{SC}_{\boldsymbol{v}}([\sigma])$ contradicts the assumption that σ is maximum. Furthermore, $m = 1$ implies $\mu = 1$ and $\sigma = 0$. From now on we thus assume $\sigma > 0$ and $m > 1$.

  We start showing that $\sigma \geq l$. Assume $\sigma < l$. Using $\frac{c(l)}{l} = x_l \leq x_\sigma \leq \frac{c(\sigma)}{\sigma}$, we get $\mathrm{SC}_{\boldsymbol{v}}([\sigma]) = c(\sigma) + \sum_{i=\sigma+1}^n v_i \geq \frac{\sigma}{l} \cdot c(l) + \frac{l-\sigma}{l} \cdot c(l) + \sum_{l+1}^n v_i = \mathrm{SC}_{\boldsymbol{v}}([l])$, a contradiction to σ being maximum.

  We now show $2\sigma > \mu$. Otherwise, $\mu \geq 2l$ and $\frac{c(2l)}{2l} \leq \frac{c(l)}{l}$ (see (P2)) contradicts the maximality of l. Specifically, by (P2), $2c(\sigma) \geq c(2\sigma) > c(\mu)$.

  We also observe that $\mu x_\mu = \frac{\mu}{l} \cdot c(l) \geq \frac{\mu}{2l} \cdot c(\mu) \geq \frac{l+1}{2l} \cdot c(\mu) \geq \frac{m}{2m-2} \cdot c(\mu)$, which follows from (P7), $\mu > \sigma \geq l$, and $m \geq \mu \geq l+1$.

  Now, with $\sum_{i=\sigma+1}^{\mu} v_i \geq (\mu - \sigma) \cdot x_\mu \geq \frac{\mu-\sigma}{\mu} \cdot \frac{m}{2m-2} \cdot c(\mu) \geq \frac{\mu-\sigma}{2m-2} \cdot c(\mu)$, it is

$$\frac{\mathrm{SC}_{\boldsymbol{v}}([\mu])}{\mathrm{SC}_{\boldsymbol{v}}([\sigma])} \leq \frac{c(\mu) + \sum_{i=\mu+1}^n v_i}{\frac{1}{2} \cdot c(\mu) + \frac{\mu-\sigma}{2m-2} \cdot c(\mu) + \sum_{\mu+1}^n v_i} \leq \frac{1}{\frac{1}{2} + \frac{\mu-\sigma}{2m-2}} \leq \frac{2m-2}{m} .$$

- *Case 3:* By definition, $x_S \leq \frac{c(S)}{S} = \frac{1}{S}$. Together with (P6) we get that for all $i \geq S$, it holds that $x_i = \frac{1}{S}$. Furthermore, for all $i > \mu$ we have that $v_i < \frac{1}{S}$ (since they did not receive service). Together with $\sigma > \mu$ and (P3) we get

$$\frac{\mathrm{SC}_{\boldsymbol{v}}([\mu])}{\mathrm{SC}_{\boldsymbol{v}}([\sigma])} \leq \frac{c(\sigma) + \sum_{\mu+1}^{\sigma} v_i + \sum_{\sigma+1}^n v_i}{c(\sigma) + \sum_{\sigma+1}^n v_i} < 1 + \frac{(\sigma - \mu) \cdot S}{S \cdot \sigma} = 2 - \frac{\mu}{\sigma} .$$

  Let $p \in \mathbb{N}_{\geq 1}, q \in \{0\} \cup [S-1]$ such that $\mu = p \cdot S + q$. By (P3) and (P4), $\mathrm{SC}_{\boldsymbol{v}}([(p+1) \cdot S]) < p+1 + \frac{\sigma - (p+1) \cdot S}{S} + \sum_{i=\sigma+1}^n v_i \leq c(\sigma) + \sum_{i=\sigma+1}^n v_i = \mathrm{SC}_{\boldsymbol{v}}([\sigma])$. Consequently, $\sigma \leq (p+1) \cdot S$, since $\sigma > (p+1) \cdot S$ contradicts the maximality of σ. Now, we have that $\frac{\mathrm{SC}_{\boldsymbol{v}}([\mu])}{\mathrm{SC}_{\boldsymbol{v}}([\sigma])} \leq 2 - \frac{\lfloor \frac{\mu}{S} \rfloor}{\lfloor \frac{\mu}{S} \rfloor + 1}$.
- *Case 4:* For $i > \mu$ we have that $v_i < x_i \leq \frac{c(i)}{i}$ and therefore it follows that $\sum_{i=\mu+1}^{\sigma} v_i < \sum_{i=\mu+1}^{\sigma} \frac{c(i)}{i} \leq \left(\sum_{i=1}^{\sigma} \frac{1}{i}\right) \cdot c(\sigma) = H_\sigma \cdot c(\sigma)$. Now,

$$\frac{\mathrm{SC}_{\boldsymbol{v}}([\mu])}{\mathrm{SC}_{\boldsymbol{v}}([\sigma])} < \frac{c(\sigma) + H_\sigma \cdot c(\sigma) + \sum_{i=\sigma+1}^n v_i}{c(\sigma) + \sum_{i=\sigma+1}^n v_i} \leq 1 + H_\sigma .$$

  Furthermore, it holds that $\sigma \leq S$. Otherwise,

$$c(S) + \sum_{i=S+1}^n v_i < 1 + \frac{\sigma - S}{S} + \sum_{i=\sigma+1}^n v_i \leq c(\sigma) + \sum_{i=\sigma+1}^n v_i ,$$

  contradicting $[\sigma]$ minimizing social cost. Thus, $\frac{\mathrm{SC}_{\boldsymbol{v}}([\mu])}{\mathrm{SC}_{\boldsymbol{v}}([\sigma])} \leq 1 + H_{\min\{n,S\}}$. □

5 Conclusion and Open Problems

In our opinion, our fine-grained analysis of efficiency loss pursues an interesting direction. Whereas for identical machines we identified cases with a small efficiency loss, we left it for future work to investigate how likely these cases may actually arise. Furthermore, we consider it interesting to find out, if such an analysis is promising for the problem of scheduling arbitrary jobs on parallel machines and/or for other cost-sharing problems.
Another problem is to improve both the approximate efficiency and budget-balance for the problem of scheduling arbitrary jobs on parallel machines – or to show that this is impossible.
Furthermore, alternatives to the makespan scheduling cost function as well as other scheduling models should be considered. There are many open issues, e.g., agents might as well be interested in the completion times of their jobs, and the provider may be interested in the number of resources expended.
Finally, one could go beyond cross-monotonicity and develop alternatives to Moulin mechanisms with better approximate budget-balance and/or efficiency.

Acknowledgements We would like to thank the two anonymous referees for their helpful comments.

References

[1] Y. Bleischwitz, B. Monien, Fair cost-sharing methods for scheduling jobs on parallel machines, in: Proceedings of the 6th Italian Conference on Algorithms and Complexity, vol. 3998 of LNCS, 2006.

[2] J. Brenner, G. Schäfer, Cost sharing methods for makespan and completion time scheduling, in: Proceedings of the 24th International Symposium on Theoretical Aspects of Computer Science, vol. 4393 of LNCS, 2007.

[3] J. Feigenbaum, A. Krishnamurthy, R. Sami, S. Shenker, Hardness results for multicast cost sharing, Theoretical Computer Science 304 (1-3) (2003) 215–236.

[4] D. K. Friesen, Tighter bounds for LPT scheduling on uniform processors, SIAM Journal on Computing 16 (3) (1987) 554–560.

[5] R. Graham, Bounds on multiprocessing timing anomalies, SIAM Journal of Applied Mathematics 17 (2) (1969) 416–429.

[6] J. Green, J.-J. Laffont, Characterizations of satisfactory mechanisms for the revelation of preferences for public goods, Econometrica 45 (2) (1977) 427–438. URL `http://www.jstor.org/stable/1911219`

[7] H. Moulin, Incremental cost sharing: Characterization by coalition strategy-proofness, Social Choice and Welfare 16 (2) (1999) 279–320.

[8] T. Roughgarden, M. Sundararajan, New trade-offs in cost-sharing mechanisms, in: Proceedings of the 38th ACM Symposium on Theory of Computing, 2006.