To Be or Not to Be (Served): Cost Sharing Without Indifferences*

Yvonne Bleischwitz¹ Burkhard Monien¹ Florian Schoppmann²†‡

yvonneb@upb.de bm@upb.de fschopp@upb.de

March 18, 2010

¹University of Paderborn, Computer Science, Fürstenallee 11, 33102 Paderborn, Germany
²Stanford University, Computer Science, 353 Serra Mall, Stanford, CA 94305, USA

Abstract. In a cost-sharing problem, finitely many players have unknown valuations for some service, and a mechanism is sought for determining which players to serve and how to distribute the incurred cost. An ideal mechanism is collusion-resistant (elicits truthful valuations even if players could collude), budget-balanced (recovers the cost), economically efficient (trades off cost and valuations), and polynomial-time computable.

We propose egalitarian mechanisms that essentially fulfill all of these properties: They satisfy a stronger notion of collusion-resistance than weak group-strategyproofness, they are exactly budget-balanced, and they are optimally efficient (among all truthful and approximately budget-balanced mechanisms). With a novel technique based on monotonic algorithms, we also achieve polynomial-time computability. We give several applications to scenarios where costs are induced by scheduling or bin-packing problems.

Compared to the known limitations of Moulin (1999) mechanisms, our results show that by sacrificing just a marginal amount of collusion-resistance, our mechanisms allow for vast performance improvements.

JEL Classification. C63, D44

Keywords. Cost sharing, group-strategyproofness, egalitarian method

1 Introduction

In a cost-sharing problem, a non-rivalrous but excludable good (i.e., a service such as inclusion in a schedule or connectivity in a network) is to be made available to \( n \in \mathbb{N} \) players at non-negative prices. Each player \( i \in \{1, \ldots, n\} \) is completely characterized by his valuation \( v_i \in \mathbb{R} \) for receiving the

---

* This work was partially supported by the IST Program of the European Union under contract number IST-15964 (AEOLUS). The first and the last author were supported by fellowships of the International Graduate School of Dynamic Intelligent Systems, Paderborn. A preliminary version of this article appeared in the Proceedings of the 3rd International Workshop on Internet and Network Economics, December 2007.

† Corresponding author

‡ Research done while the author was at the University of Paderborn.
A cost-sharing mechanism is sought that elicits truthful reports of each player’s valuation and then determines both the set of served players \( Q \subseteq \{1, \ldots, n\} \) and a distribution of the service cost \( C(Q) \in \mathbb{R}_{\geq 0} \).

Cost-sharing problems are fundamental in economics and have a broad area of applications; e.g., distributing volume discounts in electronic commerce, sharing the cost of public infrastructure projects, allocating development costs of low-volume built-to-order products, etc.; see also Moulin and Shenker (2001) and their references. The primary constraint when designing cost-sharing mechanisms is **truthfulness**, meaning that each player has an incentive to submit truthful bids to the mechanism. The cost-sharing literature typically requires a particular strong notion of truthfulness in that even collusion (i.e., coordinated wrong-bidding) must never be profitable.

Besides truthfulness, there are three essential goals for the design of cost-sharing mechanisms: The first and most natural is, of course, recovery of the service cost. Together with reasonable bounds on the generated surplus, this property is referred to as (approximate) **budget balance**. As a second goal, a mechanism should satisfy (approximate) **economic efficiency**, i.e., guarantee reasonable bounds on the **social cost** by appropriately trading off the service cost and the excluded players’ valuations. Finally, for practical applications, the outcome of the mechanism must be computable in polynomial time. Not least due to this blend of goals from different perspectives, cost sharing has attracted a great deal of interest also in computer science.

### 1.1 Design Techniques for Cost-Sharing Mechanisms

In the standard cost-sharing model, a player who is served has a utility equal to his valuation minus his payment. If a player is not served, then he will not be charged and his utility is zero. The basic notion of truthfulness, called **strategyproofness** (SP), requires that no player can improve his utility by false bidding when all other bids are kept fixed. Moreover, several concepts of resistance against coordinated manipulation are known in the literature: A mechanism is called **group-strategyproofness** (GSP) if any defection by a coalition that increases some member’s utility inevitably decreases the utility of one of its other members. A weaker notion of collusion resistance is **weak group-strategyproofness** (WGSP) that is fulfilled if any defecting coalition has at least one member whose utility does not strictly improve.

The most universal technique for the design of truthful mechanisms (not just for cost sharing) is the class of Vickrey-Clark-Groves (VCG) mechanisms (Vickrey [1961], Clarke [1971], Groves [1973]). By always picking a set of players so that the sum of the included players’ valuations minus the service cost is maximized and by an appropriate payment scheme such that each player’s payment does not directly depend on his own bid, these mechanisms are truthful (SP) and economically efficient. In fact, Green and Laffont [1977] revealed already in the 1970’s that under general assumptions the VCG mechanisms are the only class of SP mechanisms with these properties.

Unfortunately, VCG mechanisms are not resistant against collusion and fail in general to provide any guarantees for cost recovery, even when ignoring computational complexity (Moulin and Shenker [2001]). Hence, there is an intrinsic conflict: In general, truthful mechanisms cannot guarantee exact budget balance and full economic efficiency at the same time. In fact, Feigenbaum et al. (2003) gave simple cost functions for which not even (relative) approximations of both budget balance and economic efficiency can be achieved at the same time—presuming that economic efficiency is measured in terms of the traditional **social welfare** (sum of the included players’ valuations minus

---

1 The set of possible bids is equal to the set of possible valuations, and utilities are quasi-linear (see Section 2.2).
service cost). Roughgarden and Sundararajan (2009) observed that this impossibility is due to an incompatibility with the mixed-sign social welfare and suggested an alternative measure of economic efficiency: Social cost, defined as the sum of the excluded players’ valuations plus the service cost. This measure is clearly an order-preserving transformation of the social welfare, and the absolute error is always the same under both measures. Roughgarden and Sundararajan proved that measuring economic efficiency in terms of the social cost indeed makes the desired bi-criteria (relative) approximation possible; i.e., there is a large class of mechanisms that guarantee both budget balance and economic efficiency within constant approximation factors $\beta \geq 1$ and $\alpha \geq 1$, abbreviated as $\beta$-BB and $\alpha$-EFF.

Due to the unsuitability of VCG mechanisms for cost-sharing problems there is great need for other general design techniques. However, only few are known up to the present day. Essentially the only one to obtain GSP mechanisms is due to Moulin (1999). Its main ingredient is cross-monotonic cost shares $\xi_i(S)$ that never increase when the set of served players $S$ gets larger. Given such cost shares $\xi_i(S)$, a Moulin mechanism serves the maximal set of players who can afford their corresponding price—due to cross-monotonicity, a unique maximal set always exists. Algorithmically, this set can be found by simulating an iterative ascending auction: At the beginning, all players are included in $S$, and each player $i$ is offered price $\xi_i(S)$. If there is a player who cannot afford this cost share, he is dropped from $S$ and a new iteration begins. The auction terminates once all remaining players can afford their share.

The main benefit of Moulin mechanisms is that they reduce the design of GSP mechanisms to finding cross-monotonic cost-sharing methods, which are solely responsible for the mechanism’s performance. On the negative side, Immorlica et al. (2008) and Roughgarden and Sundararajan (2009, 2006) showed that there are several natural cost-sharing problems for which any Moulin mechanism inevitably suffers from poor budget balance or/and poor economic efficiency.

A family of GSP mechanisms with improved revenue guarantees was given by Bleischwitz et al. (2007b). The main idea is as follows: Given fixed cost shares $\xi_i(S)$, call a set $S$ feasible if all players contained in it can afford their cost share $\xi_i(S)$. Then, choose the set of players that lexicographically maximizes the vector of all players’ utilities, over all feasible sets. In contrast, Moulin mechanisms always choose the feasible set for which all players’ utilities attain their maximum (note here that is only well-defined for cross-monotonic cost shares). Bleischwitz et al. (2007b) applied lexicographic maximization to the special case of symmetric costs; however, at the price of sacrificing efficiency. Generalizing this technique is still an open problem.

Immorlica et al. (2008) and Penna and Ventre (2005) gave another family of cost-sharing mechanisms that are GSP only if players cannot forgo being served. In this work, however, we do assume that players may opt not to participate in order to help others. Consequently, these mechanisms are not GSP according to the definition used in this work.

A general technique for the design of WGSP mechanisms, called acyclic mechanisms, is due to Mehta et al. (2009). Their mechanisms are generalizations of Moulin mechanism and are likewise computed by simulating iterative ascending auctions. However, for any set of remaining players, there is a specific order in which prices are offered to the players. Now, whenever a player cannot afford this offer, a new iteration is started prematurely. This way, lack of cross-monotonicity can be “concealed” from the players and truthfulness be preserved, while the added versatility of acyclic mechanisms allows for improved budget balance and economic efficiency.

---

2 Formally, these mechanisms do not satisfy strong consumer sovereignty (see Section 2.2).
1.2 The Role of Indifferent Players

An immediate implication of SP is that, by unilateral deviation, each player can only influence whether he receives the service, but not his cost share. In detail, for each player $i \in [n]$ and every fixed combination of his competitors’ bids, there has to be some threshold value $\theta_i$ so that $i$ is served for a price of $\theta_i$ when bidding strictly more than $\theta_i$, and $i$ is not served when bidding strictly less. We call a player indifferent if he bids exactly his threshold value. Note that SP does not imply a particular rule for what to do with an indifferent player.

The treatment of indifferent players is a major intricacy for the design of GSP mechanisms with good performance: Since the utility of an indifferent player is zero regardless of whether he is served or not, his utility is completely unaffected by his own bid. Consequently, this player is prone to manipulate and help others either by enforcing his inclusion with a very large bid or by prompting his exclusion with a very low bid.

1.3 Our results

Models Without Indifferences In this work, we study cost sharing where indifferences do not occur. In detail, we discuss three alternative—yet in a precise sense related—models, together with explanations why we regard them as equally reasonable as the “standard model” where indifferences may exist:

i) First, we find it plausible that players’ utilities might not be strictly quasi-linear; instead, when players have the choice between being served for their valuation and not being served at all, they would still prefer the service. For an illustrative example, one might think of auctions here: E.g., one of the coauthors of this article is a collector of antiques and prefers receiving an item also if this requires spending the maximum amount of money he is willing to pay. We call mechanisms that are GSP with respect to these modified utilities as group-strategyproof against service-aware players (SGSP).

ii) Second, even when utilities are quasi-linear as in the standard model, we find it a credible assumption on human behavior that a player would not join a coalition that prevents further service to himself (let alone a coalition that decreases his utility). Under this behavioral assumption, the GSP requirement is unnecessarily strong and could be relaxed so to not imply indifference about losing the service. In detail, we say that a mechanism is weakly group-strategyproof against service-aware players (WSGSP) if any defection by a coalition that increases some member’s utility inevitably either decreases the utility of one of its other members or prevents one of its other members from further service.

iii) Third and last, one could also assert that the case where a player’s valuation equals one of the—only finitely many—prices used by the cost-sharing mechanism is a rare and negligible event (see also Juarez (2007)). Hence, the argument is that a slightly changed model where indifferences cannot occur by definition is sufficient for practical applications. Specifically, we are (only) interested here whether a mechanism is group-strategyproof on the restricted domain of valuations that results from excluding all possible cost shares used by the mechanism.

\[^3\] In the preliminary version \cite{Bleischwitz et al. 2007a} of this work, we called this property “group-strategyproof against collectors” due to the connotations explained before. In this work, we choose a more general term because we think this behavior is in no way limited to collectors and auctions.
Note that, by definition, SGSP implies WSGSP, which in turn implies GSP on the restricted domain of valuations that does not contain the cost shares used by the mechanism. The main benefit of the three models without indifferences is that they allow for cost-sharing mechanisms with much improved budget balance and economic efficiency compared to mechanisms that are required to be GSP on the usual domain of valuations \( \mathbb{R}^n \). Yet, only a small amount of collusion resistance is sacrificed: In particular, even WSGSP is a strictly more robust collusion-resistance property than WGSP.

Techniques for Designing SGSP Mechanisms  We introduce a novel family of SGSP mechanisms by devising an iterative algorithm based on set selection and price functions \( \sigma \) and \( \rho \): In each iteration, a set of players \( S \) not yet assigned a cost share is selected by \( \sigma \) and offered a price specified by \( \rho \). If there are players in \( S \) who cannot afford this price, they are rejected. Otherwise, each player in \( S \) is assigned the price. We call mechanisms induced by our new algorithm egalitarian because the algorithmic idea is reminiscent of Dutta and Ray’s algorithm for computing egalitarian solutions (Dutta and Ray, 1989).

If \( \sigma \) always selects the most cost-efficient set and \( \rho \) the respective price, egalitarian mechanisms guarantee 1-BB for arbitrary costs and additionally \( 2H_n \)-EFF for the large class of subadditive costs (\( H_n \) denotes the \( n \)-th harmonic number). Costs are subadditive when the union of two sets does not cost more than the sum of the two stand-alone costs. Hence, subadditivity seems very natural as it conveys the idea of synergies between players. In particular, our result implies for many natural optimization problems that there are SGSP (and thus WGSP) cost-sharing mechanisms that provide exact budget balance and an economic efficiency that is asymptotically optimal for truthful and (approximately) budget-balanced cost-sharing mechanisms (i.e., \( O(\log n) \)-EFF, see Dobzinski et al. (2008)). This great advantage over Moulin mechanism is, e.g., illustrated by the rooted Steiner tree problem: Here, (optimal) costs are subadditive, which ensures existence of a 1-BB and \( O(\log n) \)-EFF egalitarian mechanism. In contrast, no Moulin mechanism can be better than 2-BB (Könemann et al. 2008, Theorem 7.1) and \( \Omega(\log^2 n) \)-EFF (Roughgarden and Sundararajan, 2006, Theorem 4.2).

Afterwards, we show that the family of acyclic mechanisms by Mehta et al. (2009) is in fact also SGSP and thus notably more collusion-resistant than just WGSP. Moreover, we prove that our egalitarian mechanisms constitute a subclass of acyclic mechanisms.

Finally, we observe that SGSP and 1-BB alone is not hard to achieve. Even trivial sequential stand-alone mechanisms that charge all players marginal costs are SGSP and 1-BB. However, for many natural cost-sharing problems with subadditive costs, the economic efficiency of sequential stand-alone mechanisms is poor. Hence, more advanced mechanisms like our egalitarian mechanisms should be used here. On the other hand, when costs are supermodular, sequential stand-alone mechanisms perform very well: Using a result by Brenner and Schäfer (2008) (see the following Section 1.4), we show that they even achieve \( O(1) \)-EFF. Supermodularity means that marginal costs \( C(S \cup \{i\}) - C(S) \) can only increase as \( S \) gets larger. It is a special case of superadditive costs, which means that the union of disjoint sets is always more costly than sum of the stand-alone costs. Intuitively, superadditivity may be seen as the result of congestion.

Efficient Computation  We develop a framework and techniques for coping with the computational complexity of egalitarian mechanisms, especially if the underlying optimization problems are hard. Besides the use of approximation algorithms, the key idea here are “monotonic” costs \( C(S) \) that must not increase when replacing a player \( i \in S \) by some other player \( j \notin S \) with index \( j < i \). In
this case, finding the most cost-efficient set only requires iterating through all possible cardinalities (and not all possible subsets any more). The main issue here is how to pair good (but possibly non-monotonic) approximation algorithms with a monotonic cost function.

Finally, we give applications that underline the power of our new approach. For sharing the makespan cost of scheduling \( n \) jobs on \( m \) parallel machines, we achieve better budget balance than all known GSP mechanisms, while maintaining \( O(\log n)\)-EFF (see Table 1). Moreover, we achieve 1-BB and \( O(\log n)\)-EFF for essentially all makespan scheduling problems that are optimally solvable in polynomial time. Lastly, we also obtain results for the bin packing problem and for problems with supermodular optimal cost functions. The latter includes, e.g., several scheduling problems when the objective is to minimize the sum of completion times.

Table 1: Comparison of techniques for designing polynomial-time computable cost-sharing mechanisms, using the example of minimum-makespan problems

<table>
<thead>
<tr>
<th>Makespan Problem</th>
<th>Technique</th>
<th>Collusion Resistance</th>
<th>BB</th>
<th>EFF</th>
<th>References and Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>arbitrary jobs on related machines ( (Q</td>
<td></td>
<td>C_{max}) )</td>
<td>Moulin mechanisms</td>
<td>GSP</td>
<td>2d</td>
</tr>
<tr>
<td></td>
<td>GSP</td>
<td>2d \cdot (1 + H_n)</td>
<td>( \Omega(n) )</td>
<td>( \Omega(n) )</td>
<td>(Bleischwitz and Schoppmann, 2008)</td>
</tr>
<tr>
<td></td>
<td>GSP</td>
<td>( \sqrt[4]{\frac{17}{4}} \cdot d )</td>
<td>( \Omega(n) )</td>
<td>( \Omega(n) )</td>
<td>(Bleischwitz et al., 2007b)</td>
</tr>
<tr>
<td></td>
<td>Egalitarian mechanisms</td>
<td>SGSP</td>
<td>2</td>
<td>( 4H_n )</td>
<td>Theorem 7.6</td>
</tr>
<tr>
<td>arbitrary jobs on identical machines ( (P</td>
<td></td>
<td>C_{max}) )</td>
<td>Moulin mechanisms</td>
<td>GSP</td>
<td>( \frac{2m}{m+1} )</td>
</tr>
<tr>
<td></td>
<td>GSP</td>
<td>( \frac{2m}{m+1} ) \cdot (1 + H_n)</td>
<td>( \Omega(n) )</td>
<td>( \Omega(n) )</td>
<td>(Brenner and Schäfer, 2007)</td>
</tr>
<tr>
<td></td>
<td>Egalitarian mechanisms</td>
<td>SGSP</td>
<td>1 + ( \varepsilon )</td>
<td>2(1 + ( \varepsilon )) \cdot H_n</td>
<td>Theorem 7.8 running time exponential in ( \frac{1}{\varepsilon} )</td>
</tr>
<tr>
<td></td>
<td>SGSP</td>
<td>( \frac{4}{3} - \frac{1}{3m} )</td>
<td>( 2(\frac{4}{3} - \frac{1}{3m}) \cdot H_n )</td>
<td>Theorem 7.4 practical mechanism</td>
<td></td>
</tr>
<tr>
<td>identical jobs on related machines ( (Q</td>
<td>p_i = p</td>
<td>C_{max}) )</td>
<td>Egalitarian mechanisms</td>
<td>SGSP</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: \( d \) denotes the number of different processing requirements, \( H_n \) denotes the \( n \)-th harmonic number.

### 1.4 Further Related Work

**Applications** Besides general design techniques and obtaining characterization results, most other work on cost sharing has focused on devising “good” cross-monotonic cost-sharing methods and, more recently, “good” acyclic mechanisms. In these works, costs stem from solutions of combinatorial optimization problems, including the minimum spanning tree. [Kent and Skorin-Kapov, 1996], [Jain and...
Vazirani, 2001, 2002), Steiner tree (Jain and Vazirani, 2001; Roughgarden and Sundararajan, 2009, 2006; Mehta et al., 2009), fixed tree multicast (Pei et al., 2001; 2003; Archer et al., 2004), facility location (Pál and Tardos, 2003; Leonardi and Schäfer, 2004; Roughgarden and Sundararajan, 2006; Immorlica et al., 2008; Mehta et al., 2009), rent-or-buy-network design (Pál and Tardos, 2003; Roughgarden and Sundararajan, 2006; Gupta et al., 2008), Steiner forest (Chawla et al., 2006; Gupta et al., 2007; Könemann et al., 2008), edge/vertex/set cover (Immorlica et al., 2008; Mehta et al., 2009), and minimum makespan or minimum sum of completion times when scheduling parallel machines (Bleischwitz and Monien, 2009; Brenner and Schäfer, 2007; Bleischwitz et al., 2007b; Bleischwitz and Schoppmann, 2008; Bleischwitz et al., 2007a; Brenner and Schäfer, 2008). In this list, three results (Mehta et al., 2009; Bleischwitz et al., 2007a; Brenner and Schäfer, 2008) fit into the framework of acyclic mechanisms, all other results develop Moulin mechanisms with cross-monotonic cost shares.

Subadditive Costs and Economic Efficiency

Dobzinski et al. (2008) studied the scenario of a so-called excludable public good (Moulin, 1999; Deb and Razzolini, 1999b) where \( C(S) = 1 \) if \( S \neq \emptyset \) and \( C(\emptyset) = 0 \). They showed for any such cost-sharing problem with \( n \) players that no SP and \( \beta \)-BB (\( \beta \geq 1 \)) cost-sharing mechanism can guarantee social cost better than \( \Omega(\log n) \) times the optimal social cost. The excludable-public-good case is a special instance of many natural cost-sharing problems with subadditive optimal costs. This includes, e.g., makespan, facility location, and rooted Steiner tree problems. Consequently, \( \Omega(\log n) \)-EFF is a lower bound for all these cost-sharing problems.

Superadditive Costs and Singleton Mechanisms

Brenner and Schäfer (2008) studied several scheduling cost-sharing problems where the cost of a schedule is defined as the sum of all players’ completion times. All problems they considered have superadditive optimal costs (meaning that the union of disjoint sets is always more costly than sum of the stand-alone costs), and the excludable-public-good case case therefore cannot occur. In fact, Brenner and Schäfer (2008) gave a simple subclass of acyclic mechanisms, called singleton mechanisms, that guarantee 1-BB and constant-factor approximations of the social cost (independent of the number of players), i.e., \( O(1) \)-EFF.

Intuitively, singleton mechanisms can be specified by a complete binary tree with \( n \) levels: Every node is labeled with a player, and every path from the source to a leaf contains each player exactly once. Now applying the mechanism corresponds to finding a path from the root to a leaf: Start at the root and initialize \( Q \) as the empty set. Now proceed as follows: If the player at the current node can afford his marginal cost \( C(Q \cup \{i\}) - C(Q) \), charge him this price, add him to \( Q \), and go to the right successor node. Otherwise, go to the left successor.

Indifference Rules

Both Moulin and acyclic mechanisms treat indifferent players in an extreme way, in that they are always served. This property is referred to as upper continuity. The only known general technique for designing truthful (GSP or not) cost-sharing mechanisms that do not satisfy upper continuity is the one due to Bleischwitz et al. (2007b). For GSP mechanisms, an interesting characterization of upper continuity is due to Immorlica et al. (2008): If a GSP mechanism is upper-continuous then it has cross-monotonic cost shares. Consequently, it is a Moulin mechanism.
1.5 Roadmap

In Section 2, all preliminaries are given, including our notational conventions and a formal definition of the cost-sharing model. In Section 3, we discuss three models of cost sharing without indifferences and establish that they are essentially equivalent. Afterwards, in Section 4, we develop the novel family of egalitarian mechanisms. In Section 5, we show that all egalitarian mechanisms are acyclic, and that in fact all acyclic mechanisms are SGSP. In Section 6, we develop a framework for designing egalitarian mechanisms whose outcome can be computed in polynomial time. In Section 7, we apply this framework to various scheduling problems and bin packing.

2 The Model

2.1 Notation

For \( n, m \in \mathbb{N}_0 \), let \( \{n \ldots m\} := \{n, n+1, \ldots, m\} \) and \( [n] := \{1 \ldots n\} \). Given a vector \( x \), we denote its components by \( x = (x_1, x_2, \ldots) \). Two vectors \( x, y \) of the same dimension are called \( K\)-variants if \( x_i = y_i \) for all \( i \notin K \). In this case, we write \( y = (x\_\ldots \_\ldots k, y_k) \). Given a finite set \( S \subseteq \mathbb{N} \) and \( k \in \mathbb{N} \), we define \( \text{MIN}_k S \) as the set of the \( k \) smallest elements in \( S \). Here, \( |S| \leq k \) implies \( \text{MIN}_k S = S \).

We write \( x \preceq y \) to denote that in every component, \( x \) is no larger than \( y \). Moreover, \( x < y \) if \( x \preceq y \) and \( x \neq y \). Similarly, \( x \ll y \) if \( x \) is strictly smaller than \( y \) in every component.

To simplify notation, we will sometimes omit curly brackets around singleton sets when it is unambiguous to do so. E.g., we will write \( i\)-variants instead of \( \{i\}\)-variants or \( K \cup i \) instead of \( K \cup \{i\} \). Finally, we denote the \( n \)-th harmonic number by \( H_n = \sum_{i=1}^{n} \frac{1}{i} \).

2.2 Cost-Sharing Problems and Mechanisms

A cost-sharing problem with \( n \in \mathbb{N} \) players is specified by a cost function \( C : 2^{[n]} \rightarrow \mathbb{R}_{\geq 0} \) that associates all possible sets of served players to the incurred service cost. A set of served players \( Q^* \subseteq [n] \) together with a cost distribution \( x^* \in \mathbb{R}^n \) is called an outcome. We denote player \( i \)'s true valuation for being served by \( v_i \in \mathbb{R} \). Unless otherwise stated, we assume quasi-linear utilities, i.e., player \( i \)'s utility for outcome \((Q^*, x^*)\) is \( v_i \cdot q_i^* - x_i \) where \( q_i^* \in \{0, 1\}, q_i^* = 1 \iff i \in Q^* \).

**Definition 2.1.** A cost-sharing mechanism \( M = (Q, x) \) consists of a pair of functions \( Q : \mathbb{R}^n \rightarrow 2^{[n]} \) and \( x : \mathbb{R}^n \rightarrow \mathbb{R}^n \) that associate any bid vector \( b \) to an outcome \((Q(b), x(b))\).

Sometimes the set notation will be inconvenient, and we therefore implicitly define \( q : \mathbb{R}^n \rightarrow \{0, 1\}^n \) by \( q_i(b) = 1 \iff i \in Q(b) \). Given a cost-sharing mechanism \( M = (Q, x) \), we write \( M_i(b) := (q_i(b), x_i(b)) \) and define \( u_i(b | v_i) := v_i \cdot q_i(b) - x_i(b) \). When utilities are quasi-linear, \( u_i(b | v_i) \) is hence player \( i \)'s utility for outcome \((Q(b), x(b))\). Provided that there is no confusion about the true valuation \( v_i \), we simply write \( u_i(b) \) instead of \( u_i(b | v_i) \).\(^4\) We let \( M(b) := (M_1(b), \ldots, M_n(b)) \) and \( u(b) := (u_1(b), \ldots, u_n(b)) \).

Unless otherwise noted, we will always require three standard axiomatic properties:

- **No positive transfers** (NPT): Players never get paid, i.e., \( x_i(b) \geq 0 \).
- **Voluntary participation** (VP): When served, players never pay more than they bid; otherwise, they are charged nothing, i.e., if \( i \in Q(b) \) then \( x_i(b) \leq b_i \), else \( x_i(b) = 0 \).

\(^4\) Since the true valuation \( v_i \) is an “optional” argument, we separate it by “|” for better readability.
- **Consumer sovereignty (CS):** Each player can bid in a way so that he is served, regardless of the other players’ bids; i.e., there is a $b^\infty \in \mathbb{R}_{\geq 0}$ such that if $b_i \geq b^\infty$ then $i \in Q(b)$.

VP and NPT imply that players may opt not to participate. Technically, these players submit a negative bid. This property in conjunction with CS is sometimes referred to as **strong CS**. It strengthens the collusion-resistance requirements and rules out otherwise implausible and undesirable mechanisms [Immorlica et al., 2008].

### 2.3 Non-Manipulability

The basic notion of truthfulness is **strategyproofness (SP).** It requires a mechanism $M$ to guarantee that for all possible valuation vectors $v \in \mathbb{R}^n$, all players $i \in [n]$, and all $i$-variants $b$ of $v$ it holds that $u_i(b) \leq u_i(v)$. In this work, as well as in many related works on cost sharing, a stronger notion is required that also ensures resistance against coordinated manipulation.

**Definition 2.2.** A cost-sharing mechanism $M$ is **group-strategyproof (GSP)** if for all true valuations $v \in \mathbb{R}^n$ and all non-empty coalitions $K \subseteq [n]$ there is no $K$-variant $b$ of $v$ with $u_K(b) > u_K(v)$.

We say a non-empty coalition $K \subseteq [n]$ is GSP-successful at $v$ (or simply a successful coalition) if there is some $K$-variant $b$ of $v$ so that the coalition improves by deviating, i.e., $u_K(b) > u_K(v)$. With the corresponding modifications, we will use this terminology also for other kinds of collusion resistance.

An interesting property of GSP is that it implies a separability condition in that cost shares only depend on the set of served players and not on the bids [Moulin, 1999]:

**Definition 2.3.** A cost-sharing method is a function $\xi : 2^{[n]} \rightarrow \mathbb{R}_{\geq 0}^n$ that associates each set of players to a cost distribution. We say that a cost-sharing mechanism $M = (Q, x)$ is separable if there exists a cost-sharing method $\xi$ so that $x = \xi \circ Q$, i.e., for all $b \in \mathbb{R}^n : x(b) = \xi(Q(b))$.

A weaker notion of collusion resistance is obtained by strengthening the requirements for successful coalitions:

**Definition 2.4.** A cost-sharing mechanism $M$ is weakly group-strategyproof (WGSP) if for all true valuations $v \in \mathbb{R}^n$ and all non-empty coalitions $K \subseteq [n]$ there is no $K$-variant $b$ of $v$ with $u_K(b) \succ u_K(v)$.

Besides the coalitional variants of strategyproofness, there are several other concepts of non-manipulability. In this work, we consider a property introduced by **Satterthwaite and Sonnenschein (1981):** If a single player changes his bid in a way so that his own outcome does not change, then all other players should also get the same outcome as before.

**Definition 2.5 (Satterthwaite and Sonnenschein (1981)).** A cost-sharing mechanism $M$ is (outcome) non-bossy (ONB) if for all players $i \in [n]$ and all $i$-variants $b, b' \in \mathbb{R}^n$ it holds that $M_i(b) \neq M_i(b')$ or $M(b) = M(b')$. 


2.4 Dealing with Indifferences

A general rule of thumb for the design of truthful mechanisms is that a player’s payment must not depend directly on his own bid. In particular, the following simple proposition is well-known and a standard fact (see, e.g., \cite{deb1999a}).

**Proposition 2.6 (Threshold Property).** A cost-sharing mechanism \( M = (Q, x) \) is SP if and only if the following holds: For all \( i \in [n] \) and all \( b_{-i} \in \mathbb{R}^{[n] \setminus i} \), there is a non-negative threshold value \( \theta_i(b_{-i}) \) so that if \( b_i > \theta_i(b_{-i}) \) then \( i \in Q(b) \), if \( b_i < \theta_i(b_{-i}) \) then \( i \notin Q(b) \), and if \( i \in Q(b) \) then \( x_i(b) = \theta_i(b_{-i}) \).

We call a player \( i \) indifferent at \( b \) if \( b_i = \theta_i(b_{-i}) \), i.e., player \( i \) bids exactly his threshold value. Clearly, the threshold property leaves open how to handle indifferent player. Two extreme options are to always serve players who bid their respective threshold value or to always reject them. In fact, with the exception of the cost-sharing mechanisms by \cite{bleischwitz2007b}, this is what all known general techniques for designing truthful (GSP or not) cost-sharing mechanisms do. Formally, these two extremes are captured by the notions upper- and lower-continuity.

**Definition 2.7.** A cost-sharing mechanism \( M = (Q, x) \) is called upper-continuous if for all players \( i \) and all bid vectors \( b \) the following holds: If \( i \in Q(b_{-i}, z) \) for all \( z > b_i \) then also \( i \in Q(b) \). Likewise, \( M \) is called lower-continuous if the following holds: If \( i \notin Q(b_{-i}, z) \) for all \( z < b_i \) then also \( i \notin Q(b) \).

2.5 Budget Balance and Economic Efficiency

In typical applications, cost functions are implicitly defined by combinatorial optimization problems, i.e., \( C(S) \) is the value of a minimum-cost solution for the problem instance that corresponds to the set of served players \( S \). Due to the NP-hardness of many natural problems, usually only approximations with cost \( C'(S) \geq C(S) \) can be computed in polynomial time, unless \( P = NP \). Still, the budget of the mechanism should be reasonably balanced:

**Definition 2.8.** A mechanism \( M = (Q, x) \) is \( \beta \)-budget-balanced (\( \beta \)-BB) with regard to actual costs \( C' \) and optimal costs \( C \) if for all bid vectors \( b \) it holds that

\[
C'(Q(b)) \leq \sum_{i=1}^{n} x_i(b) \leq \beta \cdot C(Q(b)),
\]

where \( \beta \geq 1 \) is a constant (independent of \( b \)).

We define \( \beta \)-BB in the corresponding way also for cost-sharing methods. Since the definition of budget balance is meaningless otherwise, we will always assume and only consider problems with \( C(\emptyset) = 0 \).

For economic efficiency, the service cost and the rejected players’ valuations should be traded off as good as possible. A measure for this trade-off is the social cost function \( SC : 2^{[n]} \to \mathbb{R}_{\geq 0} \). Given actual costs \( C' \) and true valuations \( v \), social costs are defined by \( SC(S) := C'(S) + \sum_{i \notin S} \max\{v_i, 0\} \).

**Definition 2.9.** A mechanism \( M = (Q, x) \) is \( \alpha \)-efficient (\( \alpha \)-EFF) with regard to actual costs \( C' \) and optimal costs \( C \) if for all true valuations \( v \) it holds that

\[
SC(Q(v)) \leq \alpha \cdot \min_{P \subseteq [n]} \left\{ C(P) + \sum_{i \notin P} \max\{v_i, 0\} \right\},
\]

where \( \alpha \geq 1 \) is a constant (independent of \( v \)).
Note here that there are two potential sources for loss of economic efficiency: First, the selected set $S$ may be suboptimal; and second, the actual cost $C'(S)$ may be too high.

In the sections where polynomial-time computability is not an issue, we implicitly assume that the actual costs $C'$ coincide with the optimal costs $C$. When there is no confusion, we will usually only write $\beta$-BB and $\alpha$-EFF (and omit the “with regard to”).

2.6 Special Cost Functions

In many cases, costs exhibit a special structure that can be exploited when designing cost-sharing mechanisms. In this work, we discuss costs with the following properties:

- **Symmetric costs**: Costs depend only on the number of served players. That is, for any two sets $S, T \subseteq [n]$ with $|S| = |T| : C(S) = C(T)$.

- **Subadditive costs**: The cost of the union of two sets is never more than the sum of the stand-alone costs. That is, for any two sets $S, T \subseteq [n] : C(S \cup T) \leq C(S) + C(T)$.

- **Superadditive costs**: The cost of the union of two disjoint sets is never less than the sum of the stand-alone costs. That is, for any two sets $S, T \subseteq [n], S \cap T = \emptyset : C(S \cup T) \geq C(S) + C(T)$.

- **Submodular costs**: The marginal costs of adding players to some set $S$ are non-increasing in the size of $S$. That is, for all players $i \in [n]$ and any two sets $S \subseteq T \subseteq [n] : C(T \cup \{i\}) - C(T) \leq C(S \cup \{i\}) - C(S)$. It can be shown that this condition is equivalent to that for all $S, T \subseteq [n] : C(S \cup T) + C(S \cap T) \leq C(S) + C(T)$.

- **Supermodular costs**: Marginal costs are non-decreasing, i.e., for any two sets $S \subseteq T : C(T \cup \{i\}) - C(T) \geq C(S \cup \{i\}) - C(S)$. Equivalently, for all $S, T \subseteq [n] : C(S \cup T) + C(S \cap T) \geq C(S) + C(T)$.

3 Cost-Sharing Without Indifferences

GSP is a very strong axiom. Not only does it imply that players have full information of all others’ valuations and essentially unbounded ability to communicate and make bindings agreements. Even more, it implies that any player would agree to abandon a dominant strategy—telling the truth—even if he did not benefit from the deviation himself. In fact, he would still do so if this deviation prevents further service to himself. We assert that there are scenarios where GSP is thus unnecessarily strong.

In this section, we formally define three new collusion-resistance properties without indifferences. Afterwards, we will show how they are related and that, in a precise sense, there are some natural constraints under which all three models are equivalent. Recall that relaxing the collusion-resistance requirements implies weaker coordination capabilities (or simply less willingness to cooperate), and is done by strengthening the assumptions on successful coalitions. For instance, WGSP implies that players would only be joining a coalition if this also strictly benefits themselves.

3.1 Definitions

**Service-Aware Players** It seems plausible that utilities are not strictly quasi-linear: When players have the choice between being served for their valuation and not being served at all, they might
still prefer the service. Formally, we say that players are service-aware if a player $i$ prefers outcome $(Q^*, x^*)$ over $(Q', x')$ if and only if $q_i^* \cdot v_i - x_i^* > q_i' \cdot v_i - x_i'$ or $(q_i^* \cdot v_i - x_i = q_i' \cdot v_i - x_i'$ and $q_i^* > q_i'$).

In order to more easily compare our results, we do not define a new utility function but instead incorporate service-awareness into a new variant of collusion resistance.

**Definition 3.1.** A mechanism $M$ is group-strategyproof against service-aware players (SGSP) if for all true valuations $v \in \mathbb{R}^n$ and all non-empty coalitions $K \subseteq [n]$ there is no $K$-variant $b$ of $v$ with $(u_K(b) > u_K(v)$ and $q_K(b) \geq q_K(v))$ or $(u_K(b) \geq u_K(v)$ and $q_K(b) > q_K(v))$.

**Limited Coalition Formation** Alternatively, we propose a new notion of coalition-resistance in between GSP and WGSP. The main motivation is the behavioral assumption that players are not willing to sacrifice being served for no personal reward, i.e., players are not indifferent to losing the service. We believe that this behavior is very plausible for human beings.

**Definition 3.2.** A mechanism $M$ is weakly group-strategyproof against service-aware players (WSGSP) if for all true valuations $v \in \mathbb{R}^n$ and all non-empty coalitions $K \subseteq [n]$ there is no $K$-variant $b$ of $v$ with $u_K(b) > u_K(v)$ and $q_K(b) \geq q_K(v)$.

**Restricted Valuation Domain** A third model without indifferences was proposed by Juarez (2007). Here, the domain of valuations is restricted so that players’ valuations are always different to the cost shares used by the mechanism. Formally, this can be done as follows: Let $M$ be a cost-sharing mechanism with cost-sharing method $\xi$. For each player $i$, let $D_i := \mathbb{R} \setminus \bigcup_{S \subseteq [n]} \xi_i(S)$. That is, $D_i$ contains all reals except player $i$’s possible payments. Note that $D_i$ is still dense in $\mathbb{R}$. Define $D := D_1 \times \cdots \times D_n$. When valuations equal to some payment are a rare and in practice a negligible event, the argument is that one could just as plausibly focus only on the restriction $M|_D = (Q|_D, x|_D)$, where $Q|_D : D \to 2^{[n]}$ and $x|_D : D \to \mathbb{R}_{\geq 0}$.

**Definition 3.3.** A mechanism $M$ is group-strategyproof on the restricted domain of valuations $D \subseteq \mathbb{R}^n$ if for all true valuations $v \in D$ and all non-empty coalitions $K \subseteq [n]$ there is no $K$-variant $b \in D$ of $v$ with $u_K(b) > u_K(v)$.

### 3.2 Equivalence of Models Without Indifferences

We show that for any mechanism that is GSP on the restricted domain $D$ there is a canonical SGSP mechanism (on $\mathbb{R}^n$). Conversely, even WSGSP (on $\mathbb{R}^n$) implies GSP on the restricted domain $D$. Specifically, we show that all three models are equivalent once we require upper continuity, non-bossiness, and the threshold property.

**Lemma 3.4.** Let $M$ be a SGSP cost-sharing mechanism. Then $M$ is upper-continuous.

**Proof.** By way of contradiction, assume that $M = (Q, x)$ is not upper-continuous. Hence, there is a true valuation vector $v$ and a player $i$ so that $i \notin Q(v)$ whereas for all $z > v_i$ it holds that $i \in Q(v_{-i}, z)$. Due to the threshold property, it holds that $x(v_{-i}, z) = v_i$ for all $z > v_i$. Hence, $\{i\}$ is a successful SGSP-coalition; a contradiction.

---

5 We remark that defining such a new utility function is very well possible: For instance, the real-valued utility of some service-aware player $i$ for outcome $(Q^*, x^*)$ could be defined as $v_i - x_i^* + \varepsilon$ if both $i \in Q^*$ and $x_i^* \leq v_i$, and defined as $q_i^* \cdot v_i - x_i^*$ otherwise, for some constant $\varepsilon > 0$. 

---
Lemma 3.5. Let $M$ be a WSGSP and upper-continuous cost-sharing mechanism. Then $M$ is (outcome) non-bossy.

Proof. Denote $M = (Q, x)$. Let $v \in \mathbb{R}^n$ contain the true valuations, let $i \in [n]$, and let $b$ be an $i$-variant. As intermediate steps, we prove the following implications:

i) $q_i(b) = q_i(v) \implies u(b) = u(v)$

ii) $q_i(b) = q_i(v) = 1 \implies Q(b) = Q(v)$

iii) $q_i(b) = q_i(v) = 0$ and $b_i > v_i \implies Q(b) = Q(v)$

iv) $q_i(b) = q_i(v) = 0$ and $b_i < v_i \implies Q(b) = Q(v)$

Note that the threshold property implies that $M_i(b) = M_i(v)$ is equivalent to $q_i(b) = q_i(v)$. Implication [i] is clearly fulfilled because of WSGSP.

To see the other implications, suppose that $q_i(b) = q_i(v)$ indeed holds. By way of contradiction, assume there is a player $j \neq i$ such that, w.l.o.g., $j \notin Q(v)$ and $j \in Q(b)$, i.e., $x_j(b) = v_j$. Let $v^*$ be a $j$-variant of $v$ with $v_j < v_j^* < \theta_j(v_{-j})$. Such a $v_j^*$ exists due to upper continuity. The threshold property implies $M_j(v^*) = M_j(v)$. We also define $b^* := (b_{-j}, v_j^*)$. Hence, $b_j^* = v_j^* > v_j = b_j$ and again the threshold property implies $M_j(b^*) = M_j(b)$. We can now verify the remaining three implications:

[i] Case $q_i(b) = q_i(v) = 1$:

Then $u_i(b) = u_i(v) = u_i(v^*)$ where the last equality is due to [i]. Hence, the coalition $\{i, j\}$ can help player $j$ at $v^*$ by bidding as in $b$. This is a contradiction.

[iii] Case $q_i(b) = q_i(v) = 0$ and $b_i > v_i$:

Since $M_j(b^*) = M_j(b)$ and $j \in Q(b)$, we have $M(b^*) = M(b)$ due to implication [ii]; so, in particular, $i \notin Q(b^*)$. Since $b_j^* = b_i > v_i = v_j^*$, the threshold property implies also $i \notin Q(v^*)$. Consequently, player $i$ can help player $j$ at $v^*$ by bidding $b_i$; a contradiction.

[iv] Case $q_i(b) = q_i(v) = 0$ and $b_i < v_i$:

As in the previous case, we have $i \notin Q(b^*)$ due to implication [ii]. Moreover, $M_j(v^*) = M_j(v)$ and $v_j^* > v_j$, so $q_i(v^*) = q_i(v) = 0$ due to [iii]. Hence, player $i$ can again help player $j$ at $v^*$ by bidding $b_i$; a contradiction.

Consequently, $q_i(b) = q_i(v)$ implies $q(b) = q(v)$. Together with implication [i], this completes the proof. \qed

Theorem 3.6. Let $M$ be a cost-sharing mechanism with cost-sharing method $\xi$, and let $D$ be the restricted domain of valuations as in Section 3.1. Then, $M$ is SGSP if and only if $M\|_D$ is GSP, $M$ is upper-continuous, $M$ is (outcome) non-bossy, and $M$ fulfills the threshold property.

Proof. Sufficiency ("$\Rightarrow$") is straightforward: Upper continuity and outcome non-bossiness follow from Lemmata 3.4 and 3.5. Moreover, SGSP clearly implies SP. It also implies GSP on the restricted domain of valuations $D$: By way of contradiction, suppose there are true valuations $v \in D$, a
non-empty coalition $K$, and a $K$-variant $b \in D$ of $v$ with $u_K(b) > u_K(v)$. Then, it must hold that also $q(b) \geq q(v)$. Consequently, $K$ is SGSP-successful, a contradiction.

In the rest of the proof, we verify necessity (“$\Leftarrow$”). Let $v \in \mathbb{R}^n$ contain the true valuations, let $K$ be a non-empty coalition, and let $b \in \mathbb{R}^n$ be a $K$-variant of $v$. By way of contradiction, assume that $(u_K(b) > u_K(b)$ and $q_K(b) \geq q_K(v))$ or $(u_K(b) \geq u_K(b) and q_K(b) > q_K(v)$, i.e., $K$ is a SGSP-successful coalition. We show that both $v$ and $b$ can be transformed into $K$-variants $v', b' \in D$ so that $u_i(b \mid v_i') \geq u_i(v \mid v_i')$ for all $i \in K$, with at least one strict inequality. We give an algorithmic argument: First, initialize $v':=v$ and $b':=b$. Then, for every player $i = 1, \ldots, n$, increase both $v_i'$ and $b_i'$ by the same amount $\varepsilon > 0$ so that still $v_i', b_i' \in D$ and neither $M_i(v')$ nor $M_i(b')$ changes. Such an $\varepsilon$ exists because $D_i$ is dense in $\mathbb{R}$ and because $M$ is upper-continuous. Due to (outcome) non-bossiness, $M_{-i}(v')$ and $M_{-i}(b')$ do no change, either.

In the end, both $v' \in D$ and $b' \in D$. Moreover, $M(v) = M(v')$, $M(b) = M(b')$, and $v'$ and $b'$ are $K$-variants. It holds for all players $i \in K$ that $u_i(b' \mid v_i') \geq u_i(v' \mid v_i')$. As a last step, we show that there is at least one player $i \in K$ with $u_i(b' \mid v_i') > u_i(v' \mid v_i')$. This is obvious if there is a player $i \in K$ with $u_i(b \mid v_i) > u_i(v \mid v_i)$. If this is not the case, then there is a player $i \in Q(b) \setminus Q(v)$ with $x_i(b) = v_i$. Now, since $v_i' > v_i$, it follows that $u_i(b' \mid v_i') = v_i' - x_i(b') = v_i' - x_i(b) > 0 = u_i(v \mid v_i')$. This completes the contradiction.

Corollary 3.7. Let $M$ be a cost-sharing mechanism. Then $M$ is SGSP if and only if $M$ is WSGSP and upper-continuous.

Proof. This follows from Lemma 3.5 and Theorem 3.6.

Figure 1 gives an overview of the previous implications. In the following, we give examples showing that the converse directions do not hold true.

![Hierarchy of collusion-resistance properties](image)

**Figure 1**: Hierarchy of collusion-resistance properties

Lemma 3.8. There are cost-sharing mechanisms that are

i) GSP but not SGSP,

ii) SGSP but not GSP, or

iii) WGSP and upper-continuous but not GSP on $D$ (where $D \subset \mathbb{R}^n$ is the restricted domain of valuations as in Section 3.1).
Proof. 

i) Any lower-continuous GSP cost-sharing mechanism is not SGSP.

ii) Let there be \( n = 2 \) players and define \( M = (Q, x) \) as follows: The cost-sharing method is 
\[ \xi_1(\{1\}) = \xi_2(\{2\}) = 1 \text{ and } \xi(\{1, 2\}) = (2, 1). \] 
Moreover, 
\[ Q(b) := \begin{cases} 
\{1, 2\} & \text{if } b_1 \geq 2 \text{ and } b_2 \geq 1 
\{1\} & \text{if } b_1 \geq 1 \text{ and } b_2 < 1 
\{2\} & \text{if } b_1 < 2 \text{ and } b_2 \geq 1 
\emptyset & \text{otherwise.} 
\end{cases} \]
For an illustration, see Figure 2a. Note first that \( M \) is upper-continuous and satisfies the threshold property (Proposition 2.6). Moreover, for player 2, this threshold value is constantly \( \theta_2(b_1) = 1 \). The threshold value for player 1 is \( \theta_1(b_2) = 2 \) if player 2 gets the service (i.e., \( b_2 \geq 1 \)) and 1 otherwise; i.e., the threshold value only changes if player 2 loses the service. This proves SGSP.

\( M \) is not GSP: Assume \( v = (2, 1) \). Then player 2 could help player 1 by bidding strictly less than 1.

iii) Let there be \( n = 2 \) players and define \( M' = (Q', x') \) as follows: The cost-sharing method is 
\[ \xi_1(\{1\}) = \xi_2(\{2\}) = 2 \text{ and } \xi(\{1, 2\}) = (3, 1). \] 
Moreover, 
\[ Q'(b) := \begin{cases} 
\{1, 2\} & \text{if } b_1 \geq 3 \text{ and } b_2 \geq 1 
\{1\} & \text{if } b_1 \geq 2 \text{ and } b_2 < 1 
\{2\} & \text{if } b_1 < 3 \text{ and } b_2 \geq 2 
\emptyset & \text{otherwise.} 
\end{cases} \]
For an illustration, see Figure 2b. Note first that \( M' \) is SP as the threshold property is fulfilled. Moreover, \( M' \) is upper-continuous. Now, \( M' \) always chooses a \( b \)-feasible set that maximizes player 2’s utility. Hence, both players can never form a successful WGSP-coalition together. This proves WGSP.

Suppose now \( v = (2.5, 1.5) \). Then, player 2 could help player 1 by bidding strictly less than 1, e.g., let \( b = (2.5, 0.5) \). It holds that \( v, b \in D \), so \( M' \) is not GSP on the restricted domain \( D \).
3.3 WSGSP Implies Separability

We show in the following that already WSGSP is sufficient for separability; hence, collusion resistance according to any of the three models without indifference implies existence of a cost-sharing method. A corresponding result for GSP mechanisms is due to Moulin (1999). However, our Theorem 3.9 is stronger, since GSP is relaxed to WSGSP, and strong CS is relaxed to CS. In particular, our result continues to hold if the domain of valuations is restricted to non-negative bids.

**Theorem 3.9.** Let \( M \) be a WSGSP mechanism. Then, \( M \) is separable. This result holds even if the domain of valuations is restricted to \( \mathbb{R}^n_{\geq 0} \) (i.e., even if \( M \) does not satisfy strong CS).

**Proof.** Let \( D \) denote the domain of valuations. We establish the result both for \( D = \mathbb{R}^n \) and \( D = \mathbb{R}^n_{\geq 0} \). Define a cost-sharing method \( \xi : 2^n \rightarrow \mathbb{R}^n_{\geq 0} \) as follows: Denote \( b^- := -1 \) if \( D = \mathbb{R}^n \) and \( b^- := 0 \) if \( D = \mathbb{R}^n_{\geq 0} \). Now, define \( \hat{b} : 2^n \rightarrow D \) by \( \hat{b}(S) := b^\infty \) if \( i \in S \) and \( \hat{b}(S) := b^- \) otherwise. Then, let \( \xi(S) := x(\hat{b}(S)) \). In order to prove the theorem, it is sufficient to show that for any true valuations \( v \) it holds that \( x(v) = \xi(Q(v)) \). We do this by induction over \( m \in [n] \):

Claim (Induction Hypothesis). Suppose \( \emptyset \neq S \subseteq [n] \) with \( |S| \leq m \), and \( b \in \mathbb{R}^n \) is an \( S \)-variant of \( v \) such that \( b_i = b^\infty \) if \( i \in S \cap Q(v) \), \( b_i = b^- \) if \( i \in S \setminus Q(v) \), and \( b_i = v_i \) otherwise. Then \( u(v) = u(b) \), and for all \( i \in S \) with \( v_i > 0 \) or \( i \in Q(v) \) it holds that \( M_i(v) = M_i(b) \).

**Proof (of claim).** The base case is \( m = 1 \): Suppose \( S = \{i\} \) and \( b \) is as in the induction hypothesis. Due to the threshold property, \( M_i(b) = M_i(v) \). Hence, also \( u(b) = u(v) \) due to WSGSP. Otherwise, player \( i \) could help some other player either by bidding \( v_i \) if the true valuation vector was \( b \) or \( b_i \) if it was \( v \).

For the induction step “\( (m-1) \rightarrow m \)”, assume the induction hypothesis holds up to \( m - 1 \). Suppose \( S \subseteq [n] \) with \( |S| = m \) and \( b \) is as in the induction hypothesis. Define \( S' := \{ j \in S \mid v_j > 0 \text{ or } j \in Q(v) \} \). If \( D = \mathbb{R}^n \) then consider the coalition \( K := S \), otherwise if \( D = \mathbb{R}^n_{\geq 0} \), let \( K := S' \). In the latter case it holds for all \( j \in S \setminus K \) that \( v_j = b_j = 0 \). The proof of the claim proceeds in several steps:

i) \( u_S(b) \leq u_S(v) \)

This holds because otherwise there is a player \( i \in S \) with \( u_i(b) > u_i(v) = u_i(b_{-i}, v_i) \). Here the equality is due to the induction hypothesis. This contradicts SP.

ii) For all \( i \in K : (i \in Q(b) \iff i \in Q(v)) \)

Let \( i \in K \). Obviously, if \( i \in Q(v) \), then \( i \in Q(b) \) because \( b_i = b^\infty \). If \( D = \mathbb{R}^n \) then \( i \notin Q(v) \) similarly implies \( i \notin Q(b) \) because \( b_i = b^- < 0 \). On the other hand, if \( D = \mathbb{R}^n_{\geq 0} \) then \( i \notin Q(v) \) implies \( v_i > 0 \) by definition of \( S' \) and \( b_i = b^- = 0 \). Consequently, \( i \notin Q(b) \) because otherwise \( u_i(b) = v_i > 0 = u_i(v) \). Here, the first equality is due to NPT and VP. A contradiction to [i].

iii) \( u_S(b) \geq u_S(v) \)

This holds because otherwise there is a player \( i \in S \) with \( u_i(b) < u_i(v) \). Note that this implies \( v_i > 0 \), so \( i \in K \). Then, due to [i] and [ii], the coalition \( K \) can help \( i \) at \( b \) by bidding as in \( v \). This contradicts WSGSP.
iv) $M_K(b) = M_K(v)$ and $u(b) = u(v)$

Since $K \subseteq S$, steps (i), (ii), and (iii) clearly imply $M_K(b) = M_K(v)$. Now if there is a player $j \in [n] \setminus S$ with $u_j(b) \neq u_j(v)$, then the coalition $K$ can help $j$ either at $v$ or at $b$ by bidding $b_K$ or $v_K$, respectively.

We remark that if $D = \mathbb{R}^n_{\geq 0}$ then it may happen that $Q(\hat{b}(Q(v))) \supseteq Q(v)$. This is not a problem because all players $j \in Q(\hat{b}(Q(v))) \setminus Q(v)$ satisfy $b_j = v_j = 0$, so $x_j(b) = 0 = x_j(v)$.

3.4 WGSP Does Not Imply Separability

We close this section by observing that WGSP, in contrast to WSGSP, does not imply separability, i.e., existence of a cost-sharing method. This observation gives some intuition why WSGSP is notably stronger than only WGSP.

As a simple corollary of the threshold property, the outcome of any SP mechanism could be computed as follows: For every player $i$, compute the threshold value $\theta_i$, together with a rule what to do in case of indifference, $\phi_i \in \{serve, reject\}$. Then, serve all players $i$ with $b_i > \theta_i$ or ($b_i = \theta_i$ and $\phi_i = serve$) for price $\theta_i$; reject all others. Next, we give a simple idea to transform this observation into a WGSP mechanism.

Definition 3.10. Suppose the outcome of a mechanism $M$ can be computed as follows: For each $j = 1, \ldots, n$, compute $\sigma_j \in [n]$, $\theta_{\sigma_j} \in \mathbb{R}_{\geq 0}$, and $\phi_{\sigma_j} \in \{serve, reject\}$ as functions of $b_{\sigma_1}, \ldots, b_{\sigma_{j-1}}$ so that $\sigma_1, \ldots, \sigma_n$ becomes a permutation of $[n]$. Then $M$ is called a sequential mechanism.

Note that, for every sequential mechanism, $\sigma_1$ is a constant.

Lemma 3.11. Every sequential mechanism is WGSP.

Proof. Let $v$ contain the true valuations, let $K$ be a non-empty coalition, and let $b$ be a $K$-variant of $v$. Consider the first iteration $i$ in which a player from $K$ is considered for input $b$, i.e., $i := \min\{j \in [n] \mid \sigma_j(b) \in K\}$. Obviously we have for all $j \in [i]$ that $\sigma_j(b) = \sigma_j(v)$ and $\theta_{\sigma_j}(b) = \theta_{\sigma_j}(v)$. Hence, $u_{\sigma_i}(b) \leq u_{\sigma_i}(v)$ because the threshold value for player $\sigma_i(b) \in K$ has not changed. Therefore, $K$ cannot be a WGSP-successful coalition.

Corollary 3.12. There is a WGSP and 1-BB mechanism that is not separable.

Proof. Let there be $n = 3$ players, and let the cost function be defined by $C(S) := 1$ if $S \neq \emptyset$ and $C(\emptyset) = 0$. Define a sequential mechanism $M$ as follows: If $b_1 \geq \frac{1}{2}$, let $\sigma := (1, 2, 3)$ and otherwise $\sigma = (1, 3, 2)$. Now find the first player (according to the order $\sigma$) who can pay for himself and also for all remaining players with a non-negative bid. Let $M$ serve this set.

Formally: The threshold value is $\theta_{\sigma_i} := 1$ if $b_{\sigma_j} < 1$ for all $j < i$, and $\theta_{\sigma_i} := 0$ otherwise. Moreover, the mechanism is upper-continuous, i.e., $\phi_i := serve$ for all $i$. Now let $b := (0, 1, 1)$ and $b' := (\frac{1}{2}, 1, 1)$. Then $Q(b) = Q(b') = \{2, 3\}$ but $x(b) = (0, 1, 0)$ and $x(b') = (0, 0, 1)$.

4 Egalitarian Mechanisms

Egalitarian mechanisms borrow an algorithmic idea proposed by Dutta and Ray (1989) for computing egalitarian solutions. Given a set of players $Q \subseteq [n]$, cost shares are computed by doing the following
iteratively: Find the most cost-efficient subset $S$ of the players that have not been assigned a cost share yet. That is, the quotient of the marginal cost for including $S$ divided by $|S|$ is minimal. Then, assign each player in $S$ this quotient as his cost share. If players remain who have not been assigned a cost share yet, start a new iteration.

Before discussing most cost-efficient subsets in Section 4.2, we generalize Dutta and Ray’s idea by making use of a more general set selection function $\sigma$ and price function $\rho$. Specifically, let $Q \subseteq [n]$ be the set of players to be served. For some fixed iteration, let $N \subset Q$ be the subset of players already assigned a cost share. Then, $\sigma(Q, N)$ selects the players $S \subseteq Q \setminus N$ who are assigned the cost share $\rho(Q, N)$. We require $\sigma$ and $\rho$ to be valid (see discussion below):

**Definition 4.1.** Set selection and price functions $\sigma$ and $\rho$ are valid if the following holds for all $N \subset Q' \subseteq Q \subseteq [n]$:

W1) $\emptyset \neq \sigma(Q, N) \subseteq Q \setminus N$,

W2) $\sigma(Q, N) \subseteq Q' \implies \sigma(Q, N) = \sigma(Q', N)$ and $\rho(Q, N) = \rho(Q', N)$,

W3) $\rho(Q, N) \leq \rho(Q', N)$,

W4) $0 \leq \rho(Q, N) \leq \rho(Q, N \cup \sigma(Q, N))$.

Now egalitarian mechanisms are defined by Algorithm 1.

---

**Input:** valid set selection and price functions $\sigma, \rho$; bid vector $b \in \mathbb{R}^n$

**Output:** set of served players $Q \in 2^{[n]}$; vector of cost shares $x \in \mathbb{R}^n_{\geq 0}$

1: $Q := [n]; N := \emptyset; x := 0$
2: while $N \neq Q$ do
3: $S := \sigma(Q, N), a := \rho(Q, N)$
4: $Q := Q \setminus \{i \in S | b_i < a\}$
5: if $S \subseteq Q$ then $x_i := a$ for all $i \in S$; $N := N \cup S$

**Algorithm 1:** Egalitarian mechanisms

We shortly comment on validity: Property [W1] implies that any player is assigned a cost share only once and that the algorithm terminates. Property [W2] is a consistency property. It will ensure that the outcome does not change if a rejected player unilaterally modifies his bid such that he is rejected in a different iteration (see the proof of Theorem 5.7). Finally, properties [W3] and [W4] imply that the assigned prices are non-decreasing throughout the iterations of the algorithm.

**Theorem 4.2.** Egalitarian mechanism are SGSP.

We defer the proof of Theorem 4.2 to Section 5.1 where it will be an immediate corollary of Theorems 5.6 and 5.7.

### 4.1 Efficiency of Egalitarian Mechanisms

In order to show economic-efficiency bounds, a further property of price functions is needed:
Algorithm 1 rejects a player where the last inequality holds because the fraction on the left-hand side is at least 1 and the same decreasing. Now let $|\epsilon|$, non-negative value is subtracted in both numerator and denominator. Now, consider an arbitrary after line 3 in that iteration $k$ and indicate them with a subscript $k$. Since player $i$ is dropped, $b_i < a_k = \rho(Q_k, N_k) \leq \beta \cdot \frac{C(A)}{|A|}$, where the last inequality holds because $A \subseteq Q_k \setminus N_k$ and $\beta$ is $\beta$-average. A contradiction.

\textbf{Theorem 4.5.} Let $\sigma$ and $\rho$ be valid set selection and price functions such that $\rho$ is $\beta$-average for non-decreasing costs $C$. Suppose the egalitarian mechanism $M = (Q, x)$ always recovers at least the actual cost $C'$. Then, $M$ is $(2\beta \cdot H_n)$-EFF.

\textbf{Proof.} Let $v$ contains the true valuations. Denote $Q := Q(v)$, $x := x(v)$, and let $\bar{v} := \max\{v_i, 0\}$. Moreover, let $P \subseteq [n]$ be a set that minimizes $C(P) + \sum_{i \notin P} \bar{v}_i$. We have

$$SC(Q) = C'(Q) + \sum_{i \in [n] \setminus Q} \bar{v}_i \leq \sum_{i \in Q \cap P} x_i + \sum_{i \in Q \setminus P} x_i + \sum_{i \in [n] \setminus Q} \bar{v}_i$$

due to cost recovery

$$\leq \sum_{i \in Q \cap P} x_i + \sum_{i \in P \setminus Q} \bar{v}_i + \sum_{i \in [n] \setminus P} \bar{v}_i$$

due to $x_i \leq v_i$ for $i \in Q$.

Hence,

$$\frac{SC(Q)}{C(P) + \sum_{i \notin P} \bar{v}_i} \leq \frac{\sum_{i \in Q \cap P} x_i + \sum_{i \in P \setminus Q} \bar{v}_i + \sum_{i \in [n] \setminus P} \bar{v}_i}{C(P) + \sum_{i \in [n] \setminus P} \bar{v}_i} \leq \frac{\sum_{i \in Q \cap P} x_i + \sum_{i \in P \setminus Q} \bar{v}_i}{C(P)}.$$

The last inequality holds because the fraction on the left-hand side is at least 1 and the same non-negative value is subtracted in both numerator and denominator. Now, consider an arbitrary iteration $k$ when Algorithm 1 decides to accept a player $i \in Q \cap P$ in line 4. Fix all variables just after line 3 in that iteration $k$ and indicate them with a subscript $k$. We have

$$x_i = a_k = \rho(Q_k, N_k) \leq \beta \cdot \frac{C((Q \cap P) \setminus N_k)}{|(Q \cap P) \setminus N_k|} \leq \beta \cdot \frac{C(Q \cap P)}{|(Q \cap P) \setminus N_k|},$$

where the inequalities hold because $(Q \cap P) \setminus N_k \subseteq Q_k \setminus N_k$, $\beta$ is $\beta$-average for costs $C$, and $C$ is non-decreasing. Now let $i_1, \ldots, i_{|Q \cap P|}$ be the players in $Q \cap P$ ordered according to the iteration in which
they are accepted. Note that if a player \(i_j\) is accepted in iteration \(k\), then \(|(Q \cap P) \setminus N_k| \geq |Q \cap P| - j + 1\) because at most \(j - 1\) players from \(Q \cap P\) can be contained in \(N_k\). Consequently, we get

\[
x_{ij} \leq \beta \cdot \frac{C(Q \cap P)}{|Q \cap P| - j + 1}
\]

and thus \(\sum_{i \in Q \cap P} x_i \leq \beta \cdot H_{|Q \cap P|} \cdot C(Q \cap P)\).

On the other hand, in \(P \setminus Q\), there is at least one player \(i\) with \(v_i < \beta \cdot \frac{C(P \setminus Q)}{|P \setminus Q|}\). Otherwise, due to Lemma 4.4, we would have \((P \setminus Q) \cap Q \neq \emptyset\), a contradiction. Inductively and by the same lemma, for every \(j = 1, \ldots, |P \setminus Q| - 1\), there has to be a player \(i \in P \setminus Q\) with \(v_i < \beta \cdot \frac{C(P \setminus Q)}{|P \setminus Q| - j}\). Therefore, \(\sum_{i \in P \setminus Q} v_i \leq \beta \cdot H_{|P \setminus Q|} \cdot C(P \setminus Q)\).

Combining the previous bounds and exploiting that \(C\) is non-decreasing, we get

\[
\frac{SC(Q)}{C(P) + \sum_{i \notin P} v_i} \leq \frac{\beta \cdot H_{\max\{|Q \cap P|, |P \setminus Q|\}} \cdot (C(Q \cap P) + C(P \setminus Q))}{C(P)} \leq 2 \beta \cdot H_n.
\]

This completes the proof. \(\square\)

### 4.2 Most Cost-Efficient Set Selection

How can concrete set selection and price functions be defined so that they are valid and the previous findings apply? This is what we answer next.

**Definition 4.6.** A set selection function \(\sigma\) and its corresponding price function \(\rho\) are called most cost-efficient with regard to optimal costs \(C\) if they satisfy (W2) and

\[
\sigma(Q, N) \in \arg \min_{\emptyset \neq T \subseteq Q \setminus N} \left\{ \frac{C(N \cup T) - C(N)}{|T|} \right\},
\]

\[
\rho(Q, N) = \min_{\emptyset \neq T \subseteq Q \setminus N} \left\{ \frac{C(N \cup T) - C(N)}{|T|} \right\}.
\]

We remark that (W2) is not hard to achieve: A canonical way is, e.g., to always choose the lexicographic maximum of all sets contained in \(\arg \min_T \{(C(N \cup T) - C(N))/|T|\}\).

**Lemma 4.7.** Most cost-efficient set selection and price functions \(\sigma\) and \(\rho\) are valid. If the costs \(C\) are subadditive then \(\rho\) is also 1-average for \(C\).

**Proof.** It is a simple observation that \(\sigma\) and \(\rho\) fulfill properties (W1), (W3) of Definition 4.1. To see property (W4) let \(N \subseteq Q \subseteq |n|\). Define \(S := \sigma(Q, N), a := \rho(Q, N)\) and \(S' := \sigma(Q, N \cup S), a' := \rho(Q, N \cup S)\). Then,

\[
a \leq \frac{C(N \cup S \cup S') - C(N)}{|S| + |S'|} = \frac{C(N \cup S \cup S') - C(N \cup S) + |S| \cdot a}{|S| + |S'|},
\]

thereby implying that

\[
a \leq \frac{C(N \cup S \cup S') - C(N \cup S)}{|S'|} = a'.
\]
Now assume that $C$ is subadditive. Again, let $N \subseteq Q \subseteq [n]$ and $\emptyset \neq A \subseteq Q \setminus N$. Then,

$$\rho(Q, N) = \min_{\emptyset \neq T \subseteq Q \setminus N} \left\{ \frac{C(Q \cup T) - C(Q)}{|T|} \right\} \leq \frac{C(Q \cup A) - C(Q)}{|A|} \leq \frac{C(A)}{|A|}.$$ 

Hence, $\rho$ is 1-average for $C$ if $C$ is subadditive. \hfill \Box

As a corollary of Theorem 4.5 and Lemma 4.7 we get:

**Theorem 4.8.** For arbitrary costs $C$, any egalitarian mechanism $M$ induced by most cost-efficient set selection and prices is 1-BB. If $C$ is non-decreasing and subadditive, then $M$ is also $2H_n$-EFF.

Unfortunately, evaluating a most cost-efficient set selection function $\sigma$ can take exponentially many steps (in $n$). Furthermore, computing optimal costs $C$ is often NP-hard. In Section 6 we thus study how to pick “suitable” cost-efficient subsets in polynomial time. We conclude this subsection by showing that our bound on the social cost is tight up to a factor of 2.

**Lemma 4.9.** For costs $C$ defined by $C(S) := 1$ for all $\emptyset \neq S \subseteq [n]$, any egalitarian mechanism induced by most cost-efficient set selection and prices is no better than $H_n$-EFF.

**Proof.** Let $v := (1 - \epsilon)_{i=1}^{n}$ be the true valuation vector, where $\epsilon \in (0, \frac{1}{n})$. Then, $Q(v) = \emptyset$ because in Algorithm 1 line 4, one player after the other would be dropped. Now, $C([n]) = 1$ while $SC(Q(v)) = 4 + (z - 5) = 6$. \hfill \Box

**Lemma 4.10.** For any $\alpha > 1$, there is a non-decreasing cost function $C : 2^{[4]} \to \mathbb{R}_{\geq 0}$ so that no egalitarian mechanism induced by most cost-efficient set selection and prices is better than $\alpha$-EFF.

**Proof.** Let $z := 6\alpha + 1$. Define $C$ as follows: $C(\{i\}) = 1$ for all $i \in [4]$. Let $C(\{1, 2\}) := 2$ and $C(T) := 3$ for any other $T \subset [4]$ with $|T| = 2$. Let $C(\{1, 2, 3\}) := 4$ and $C(T) := 5$ for any other $T \subset [4]$ with $|T| = 3$. Furthermore, $C([4]) := z$.

Let $M = (Q, x)$ be an egalitarian mechanism induced by most cost-efficient set selection and prices, and suppose the true valuation vector is $v = (1, 1, 2, z - 5)$. Algorithm 1 first accepts $\{1, 2\}$, each for a price of 1. Subsequently, it gives the service to 3 for a price of 2 and in the next iteration, player 4 is rejected. Therefore, $Q(v) = \{1, 2, 3\}$ and $SC(Q(v)) = 4 + (z - 5) = 6\alpha$. However, $C(\{2, 3, 4\}) + v_1 = 5 + 1 = 6$. \hfill \Box

### 4.3 Submodular and Supermodular Costs

We remark that if a cost function $C$ is submodular, then the egalitarian mechanism induced by most cost-efficient set selection and prices is unique. Moreover, it is also a Moulin mechanism. This holds because by Definition 4.6 its cost-sharing method is exactly the egalitarian solution by Dutta and Ray (1989)—and for submodular costs, the egalitarian solution is known to produce cross-monotonic cost shares (Dutta, 1990).

On the other hand, when costs are supermodular, there is always a singleton set among the most cost-efficient subsets: Suppose the set of remaining players is $Q$, and the set of already accepted
players is \( N \). Consider now an arbitrary set \( T = \{ t_1, \ldots, t_{|T|} \} \subseteq Q \setminus N \). Denote \( N_i := N \cup \{ t_i \} \). Due to supermodularity, we get

\[
C(N \cup T) = C \left( \bigcup_{i=1}^{|T|} N_i \right) \geq C(N_1) + C \left( \bigcup_{i=2}^{|T|} N_i \right) - C(N) \\
\geq \cdots \geq \sum_{i=1}^{|T|} (C(N_i) - C(N)) + C(N).
\]

By an averaging argument, there is at least one \( i \in [|T|] \) so that

\[
\frac{C(N \cup T) - C(N)}{|T|} \geq \frac{\sum_{j=1}^{|T|} (C(N_j) - C(N))}{|T|} \geq C(N_i) - C(N),
\]

which proves the claim. Hence, when costs are supermodular, egalitarian mechanisms based on most cost-efficient set selection are sequential mechanisms (cf. Definition 3.10). They also coincide with the singleton mechanisms by Brenner and Schäfer (2008). In general, singleton mechanisms can be seen as egalitarian mechanism with set and price selection functions that satisfy only conditions (W1) and (W2) of Definition 4.1 but that additionally fulfill that \( \sigma(Q, N) \) is always a singleton set.

5 Acyclic Mechanisms and SGSP

Acyclic mechanisms have been introduced by Mehta et al. (2009) as a generalization (from an algorithmic point of view) of Moulin mechanism. Their outcome is likewise computed by simulating iterative ascending auctions. However, for any set of remaining players \( S \), there is a specific order in which prices are offered to the players. This order is specified by an offer function \( \tau : 2^{[n]} \rightarrow \mathbb{R}_{\geq 0} \). Now, whenever a player cannot afford an offer, a new iteration is started prematurely. Roughly speaking, acyclic mechanisms "conceal" the lack of cross-monotonicity from the players and thus preserve truthfulness. Mehta et al. (2009) proved that acyclic mechanisms are WGSP if they are driven by valid cost-sharing methods and offer functions.

**Definition 5.1 (Mehta et al. (2009)).** Let \( \xi \) be a cost-sharing method and \( \tau \) be an offer function. For all \( i \in S \subseteq [n] \) let

\[
E_i(S) := \{ j \in S \mid \tau_j(S) = \tau_i(S) \}, \\
L_i(S) := \{ j \in S \mid \tau_j(S) < \tau_i(S) \}, \\
G_i(S) := \{ j \in S \mid \tau_j(S) > \tau_i(S) \}
\]

be the subsets of players in \( S \) with equal, lesser, and greater offer time compared to \( i \). Then \( \tau \) is called valid for \( \xi \) if for all \( i \in S \subseteq [n] \):

i) \( \xi_i(S \setminus T) = \xi_i(S) \) for all \( T \subseteq G_i(S) \) and

ii) \( \xi_i(S \setminus T) \geq \xi_i(S) \) for all \( T \subseteq G_i(S) \cup (E_i(S) \setminus \{i\}) \).
Input: cost-sharing method $\xi$; valid offer function $\tau$; bid vector $b$
Output: set of players $Q$, vector of cost shares $x$

1: $Q := [n]$
2: while $\exists i \in Q : b_i < \xi_i(Q)$ do
3: Choose an arbitrary non-empty set $T \subseteq \arg\min_i \{\tau_i(Q)\}$
4: $Q := Q \setminus T$
5: $x := \xi(Q)$

Algorithm 2: Acyclic mechanisms

Now, acyclic mechanisms are defined by Algorithm 2.

We remark that Algorithm 2 is more general than the original algorithm given by Mehta et al. (2009). They proposed a special case of Algorithm 2 where $T$ in line 3 is always a singleton set, chosen deterministically according to some arbitrary tie breaking scheme. For instance, such a deterministic tie breaking scheme could be to always pick the singleton set $T$ consisting only of the player with the smallest number; i.e., formally $T := \min(\arg\min_i \{\tau_i(Q)\})$.

5.1 Egalitarian Mechanisms Are Acyclic

In the following, we show that acyclic mechanisms are well-defined also by Algorithm 2. As a welcome by-product, we immediately get that acyclic mechanisms are in fact SGSP and thus notably stronger than only WGSP. Moreover, it is then a straightforward result that egalitarian mechanisms are indeed acyclic.

We start with a simple proposition, which was already shown by Mehta et al. (2009). Even though their proof was only for the special case of Algorithm 2 as described above, it can be reused word by word. Like Mehta et al. (2009), we say that a player $j$ is offered the price $p$ in iteration $i$ if the following conditions hold immediately before line 3 of iteration $i$: First, $j \in Q$. Second, if $k$ is a player who will be dropped in line 4, then $\tau_j(Q) \leq \tau_k(Q)$. Third, $p = \xi_j(Q)$.

Proposition 5.2 (Mehta et al. (2009)). Suppose Algorithm 2 offers the price $p$ to player $j$ in iteration $i$ and the price $p'$ in a subsequent iteration. Then $p \leq p'$. Moreover, suppose $Q^*$ is the set of players returned by Algorithm 2. Then, at the beginning of iteration $i$ (i.e., immediately after line 2), it holds that $L_j(Q) \subseteq Q^*$ and for all $k \in L_j(Q)$ it holds that $\xi_k(Q^*) = \xi_k(Q)$.

The next two technical lemmata contain the main technique for showing that the order in which players are dropped is irrelevant.

Lemma 5.3. Let $S$ be an output set of Algorithm 2 for the bid vector $b$. Suppose $A$ is a strict superset of $S$, i.e., $S \subset A \subseteq [n]$. Then the following holds:

i) There is a player $k \in A \setminus S$ with $b_k < \xi_k(A)$.

ii) Suppose there is a player $j \in S$ with $\xi_j(S) < \xi_j(A)$. Then there is a player $k \in L_j(A) \setminus S$ with $b_k < \xi_k(A)$.

Proof. i) Consider the first iteration in which some $k \in A \setminus S$ is dropped. Immediately before line 4 it holds that $b_k < \xi_k(Q) \leq \xi_k(A)$. Here, the second inequality holds because Proposition 5.2 implies $A = Q \setminus B$ for some $B \subseteq G_k(Q) \cup (E_k(Q) \setminus \{k\})$. 

23
ii) Due to Definition 5.1, $L_j(A) \setminus S$ is non-empty, and for all $\ell \in L_j(A) \setminus S$ it holds that $\xi_{\ell}(A) = \xi_{\ell}(S \cup L_j(A))$ because $S \cup L_j(A) = A \setminus B$ for some set $B \subseteq G_\ell(S)$. Define now $A' := S \cup L_j(A)$. Since $S \subseteq A' \subseteq A$, it follows by \[\] that there is a player $k \in A' \setminus S = L_j(A) \setminus S$ with $b_k < \xi_k(A') = \xi_k(S \cup L_j(A)) = \xi_k(A)$. \[\]

**Lemma 5.4.** Let $K \subseteq [n]$ be a set of players, and let $v, b$ be $K$-variants. Consider two distinct executions of Algorithm 2, the first for input $v$ and the second for $b$. Let the output sets be $R$ and $S$, respectively. Suppose that $K \cap S \supseteq K \cap R$ and $\xi_K(S) \leq v_K$ (if $K = \emptyset$ then this is a vacuous truth). Then, $S \subseteq R$.

**Proof.** Let $r$ denote the number of iterations (i.e., repetitions of the body of the while-loop) needed for the first execution (with input $v$). Define $Q_0 := [n]$, and for $i \in [r]$, define $Q_i$ and $T_i$ as the values of $Q$ and $T$ at the end of iteration $i$ (i.e., immediately after line 4). Clearly, it holds for all $i \in [r]$ that $Q_i = [n] \setminus (T_1 \cup \cdots \cup T_i)$. Moreover, $Q_r = R$.

Now let $q \in \mathbb{N}_0$ be maximal so that the first $q$ iterations are identical for both executions; i.e., in each iteration $i = 1, \ldots, q$ of the second execution for input $b$ the set $T_i$ is chosen, too. Consequently, $S \subseteq Q_q$. In the following, we show by induction over $i \in \{q+1, \ldots, r\}$ that $S \subseteq Q_i$.

We first verify the base case $i = q + 1$. Consider an arbitrary player $j \in T_{q+1}$. Also during the second execution, he is offered price $\xi_j(Q_q) > v_j$ in iteration $q + 1$. Due to Proposition 5.2, any price offered to him in a subsequent iteration cannot be smaller. Consequently, $j \notin S$.

Finally, we verify the induction step $i \rightarrow (i+1)$. Due to the induction hypothesis, $S \subseteq Q_i$. Consider again an arbitrary player $j \in T_{i+1}$. By way of contradiction, assume $j \in S$. Then, $\xi_j(Q_i) > v_j \geq \xi_j(S)$ where the first inequality is due to $j \in T_{i+1}$. Consequently, due to Lemma 5.3, it holds that there is a player $k \in L_j(Q_i) \setminus S$ with $b_k < \xi_k(Q_i)$. However, by Proposition 5.2, it holds for all players $k \in L_j(Q_i)$ that $k \in R$ and $k \notin T_{i+1}$, so $v_k \geq \xi_k(Q_i) > b_k$. Then, $k \in K$ but $k \in R \setminus S$. This is a contradiction. \[\]

**Theorem 5.5.** The output of Algorithm 2 is independent of the way $T$ in chosen in line 3.

**Proof.** This follows immediately from Lemma 5.4 for the special case $v = b$, i.e., $K = \emptyset$. \[\]

**Theorem 5.6.** Acyclic mechanisms are SGSP.

**Proof.** Let $M = (Q, x)$ be an acyclic mechanism, let $v$ contain the true valuations, let $K$ be a non-empty coalition, and let $b$ be a $K$-variant. Define $R := Q(v)$ and $S := Q(b)$. By way of contradiction, assume that $K$ is SGSP-successful, i.e., $(u_K(b) > u_K(v)$ and $K \cap S \supseteq K \cap R$) or $(u_K(b) \geq u_K(v)$ and $K \cap S \supseteq K \cap R$).

By Lemma 5.4, it follows that $S \subseteq R$. Hence, there is a player $j \in K \cap S$ with $\xi_j(S) < \xi_j(R)$, i.e., even $S \subseteq R$. Due to Lemma 5.3, there is then a player $k \in L_j(R) \setminus S$ with $b_k < \xi_k(R) \leq v_k$, so $k \in K$ but $k \in R \setminus S$. This is a contradiction. \[\]

We now show that for valid set selection and price functions $\sigma$ and $\rho$, Algorithm 1 gives the same result as running Algorithm 2 with cost-sharing method $\xi$ and offer function $\tau$ as defined by Algorithm 3. Hence, every egalitarian mechanism is acyclic.

**Theorem 5.7.** Egalitarian mechanisms are acyclic mechanisms.
We close this section by noting that SGSP and 1-BB alone is in fact not hard to achieve. The valid set selection and price functions and thus SGSP.

Lemma 5.8. Sequential stand-alone mechanisms are 1-BB. Moreover, they are acyclic mechanisms and thus SGSP.

5.2 Sequential Stand-Alone Mechanisms

We close this section by noting that SGSP and 1-BB alone is in fact not hard to achieve. The following (sequential) mechanisms are called sequential stand-alone mechanisms by Moulin (1999) and work as follows: Start with the empty player set $Q$ and do the following iteratively, for every player $i = 1, \ldots, n$: If $i$ can afford his marginal cost $C(Q \cup \{i\}) - C(Q)$, then charge him this price and add him to $Q$. Otherwise, he will not be served. A formal definition is given in Algorithm 4.

Algorithm 3: Cost-sharing method and offer function of egalitarian mechanisms

Proof. Let $\sigma, \rho$ be valid set and price selection functions, and let $\xi$ and $\tau$ be the cost-sharing method and offer functions defined by Algorithm 3. We first show that $\tau$ is valid for $\xi$. Denote the values of all variables immediately after line 3 of iteration $k$ with a subscript $k$ and with the input player set in parentheses. Let $Q \subseteq [n]$ and $i \in Q$ be arbitrary. Fix now $k$ as the (unique) iteration where $i \in S_k(Q)$.

Let $T \subseteq G_i(Q)$ be arbitrary. We show by induction over $m \in [k]$ that $N_m(Q) = N_m(Q \setminus T)$, $S_m(Q) = S_m(Q \setminus T)$, and $a_m(Q) = a_m(Q \setminus T)$. Then, $\xi_i(Q) = a_k(Q \setminus T) \leq \xi_i(Q \setminus T)$. For the base case $m = 1$, we have $N_1(Q) = N_1(Q \setminus T) = \emptyset$. For the induction step $m - 1 \rightarrow m$, the induction hypothesis implies $N_m(Q) = N_m(Q \setminus T)$.

Now, both for the base case and for the induction step, we have $S_m(Q) \subseteq Q \setminus T$ because for all $j \in S_m(Q)$: $\tau_i(Q) = m$ whereas for all $j \in T$: $\tau_i(T) > k$. Hence, (W2) implies $S_m(Q) = S_m(Q \setminus T)$ and $a_m(Q) = a_m(Q \setminus T)$.

Let $T \subseteq G_i(Q) \cup (E_i(Q) \setminus \{i\})$ be arbitrary. With exactly the same inductive argument as for (i), we get for all $m \in [k - 1]$ that $N_m(Q) = N_m(Q \setminus T)$, $S_m(Q) = S_m(Q \setminus T)$, and $a_m(Q) = a_m(Q \setminus T)$. Moreover, also $N_k(Q) = N_k(Q \setminus T)$. Now, due to property (W3) and $Q \setminus T \subseteq Q$, we have $a_k(Q) \leq a_k(Q \setminus T)$. Furthermore, $a_k(Q \setminus T) \leq \xi_i(Q \setminus T)$ since $a$ is non-decreasing in Algorithm 3 due to property (W4). Thus, $\xi_i(Q) = a_k(Q) \leq a_k(Q \setminus T) \leq \xi_i(Q \setminus T)$.

Finally, we show that the egalitarian mechanism induced by $\sigma, \rho$ yields the same outcome as the acyclic mechanism induced by $\xi, \tau$. Whenever Algorithm 1 accepts a set $S := \sigma(Q, N)$ this means that the players in $S$ have the minimum offering time of those in $Q \setminus N$ and that $b_i \geq a := \rho(Q, N)$ for all $i \in S$. Consequently, also the acyclic mechanism serves these players for the same price. On the other hand, when Algorithm 1 rejects players from $S$, the same players are also rejected by the acyclic mechanism, due to Theorem 5.5.

□
**Input:** non-decreasing cost function $C : 2^{[n]} \to \mathbb{R}_{\geq 0}$; bid vector $b \in \mathbb{R}^n$

**Output:** set of served players $Q \in 2^{[n]}$; vector of cost shares $x \in \mathbb{R}_{\geq 0}^n$

1: $Q := \emptyset$; $x := 0$
2: for $i := 1, \ldots, n$ do
3: \quad if $b_i \geq C(Q \cup \{i\}) - C(Q)$ then
4: \quad \quad $x_i := C(Q \cup \{i\}) - C(Q)$; $Q := Q \cup \{i\}$

**Algorithm 4:** Sequential stand-alone mechanisms

**Proof.** Define the cost-sharing method $\xi$ and the offer function $\tau$ as follows. For $S \subseteq [n]$ and $i \in [n]$, let $\xi_i(S) := C(S \cap [i]) - C(S \cap [i - 1])$. Furthermore, $\tau_i(S) := i$. Note that $\tau$ is indeed valid for $\xi$. It is now a simple observation that Algorithm 2 with input $\xi$, $\tau$, and $b$ yields the same output as Algorithm 4 with input $C$ and $b$. \hfill \square

However, for many natural cost-sharing problems with subadditive costs, the economic efficiency of sequential stand-alone mechanisms is poor. There is hence good reason to use more advanced mechanisms like our egalitarian ones:

**Lemma 5.9.** For the cost function $C : 2^{[n]} \to \mathbb{R}_{\geq 0}$ with $C(S) = 1$ for all $\emptyset \neq S \subseteq [n]$, sequential stand-alone mechanisms are no better than $n$-EFF.

**Proof.** Let $M = (Q, x)$ be the sequential stand-alone mechanism for $C$. Let $v := (1 - \varepsilon)^i_{i=1\ldots n}$. Then, $Q(v) = \emptyset$ and thus $SC(Q(v)) = (1 - \varepsilon) \cdot n$. However, $SC([n]) = 1$. \hfill \square

Interestingly, sequential stand-alone mechanisms are useful for cost-sharing problems with supermodular costs. Recall that supermodular costs imply that serving two disjoint sets of players separately is never more costly than serving both groups at once. They can best be seen as a result of congestion effects that occur in the underlying optimization problem. This includes, for instance, traffic networks where the objective is the total (or average) latency or min-sum scheduling problems (see, e.g., [Schulz and Uhan (2007)]).

In order to show bounds on the economic efficiency, we slightly extend the notion of subadditivity: We say a cost function $C$ is $\alpha$-subadditive ($\alpha \geq 1$) if for all $A, B \subseteq [n]$ it holds that $C(A \cup B) \leq \alpha \cdot (C(A) + C(B))$.

**Theorem 5.10.** For any costs $C$ the sequential stand-alone mechanism $M$ is $1$-BB. If $C$ is supermodular and $\alpha$-subadditive, i.e., it always holds that $C(A) + C(B) - C(A \cap B) \leq C(A \cup B) \leq \alpha \cdot (C(A) + C(B))$, then $M$ is also $\alpha$-EFF.

In order to prove Theorem 5.10 we use a result by [Brenner and Schäfer (2008)]. They gave a bound on the economic efficiency of singleton mechanisms, which are clearly a superclass of sequential stand-alone mechanisms. The proof bears some resemblance to Theorem 4.5 and is not repeated here. Note that we state the result only in terms of sequential stand-alone mechanisms.

**Definition 5.11** [Brenner and Schäfer (2008)]. A cost-sharing method $\xi$ is said to be weakly monotone with respect to costs $C$ if for all sets of players $A \subseteq B \subseteq [n]$ it holds that $\sum_{i \in A} \xi_i(B) \geq C(A)$.  

26
Proposition 5.12 (Brenner and Schäfer (2008)). Let \( M \) be a sequential stand-alone mechanism so that its induced cost-sharing method \( \xi \) is weakly monotone with respect to costs \( C \). Suppose that for all \( A, B \subseteq [n] \) it holds that \( C(A \cup B) \leq \alpha \cdot (C(A) + C(B)) \). Then, \( M \) is \( \alpha \)-EFF.

Proof (of Theorem 5.10). Let \( \xi \) denote the cost-sharing method induced by \( M \). It is straightforward to see that \( \xi \) is "cross-monotonic with reversed signs" (formally, \(-\xi \) is cross-monotonic): Let \( A \subseteq B \subseteq [n] \). Then it holds for all \( i \in [n] \) that \( \xi_i(B) = C(B \cap [i]) - C(B \cap [i-1]) \geq C(A \cap [i]) - C(A \cap [i-1]) = \xi_i(A) \). Consequently, \( \xi \) is weakly monotone with respect to \( C \) because

\[
\sum_{i \in A} \xi_i(B) \geq \sum_{i \in A} \xi_i(A) = C(A).
\]

Now the proof follows by Proposition 5.12.

\[\square\]

6 A Framework for Polynomial-Time Computation

In this section, we show how to solve cost-sharing problems in polynomial time by using egalitarian mechanisms with a set selection function that picks the most cost-efficient set with regard to costs of approximate solutions.

Formally, an optimization problem with the objective to minimize cost is a triple \( \Pi = (D, S = (S_I)_{I \in D}, f = (f_I)_{I \in D}) \), where \( D \) is the set of problem instances (domain) such that for any instance \( I \in D \), \( S_I \) is the set of feasible solutions, and \( f_I : S_I \to \mathbb{R}_{\geq 0} \) is a function mapping any solution to its cost.

We henceforth write a cost-sharing problem as a pair \( \Phi = (\Pi, \text{INST}) \), where \( \Pi \) is the underlying optimization problem and \( \text{INST} : 2^{[n]} \to D \) denotes the function mapping any subset of the \( n \) players to the respective instance of \( \Pi \). In particular, \( \Phi \) implicitly defines the optimal cost \( C : 2^{[n]} \to \mathbb{R}_{\geq 0} \) by \( C(T) := \min_{Z \in S_{\text{INST}(T)}} \{f(Z)\} \). Moreover, for any algorithm \( \text{ALG} \) that computes feasible solutions for \( \Pi \), we define \( C_{\text{ALG}} : 2^{[n]} \to \mathbb{R}_{\geq 0} \) by \( C_{\text{ALG}}(T) := f(\text{ALG}(\text{INST}(T))) \).

Resorting to approximate solutions does, of course, not yet remedy the need to iterate through all available subsets in order to pick the most cost-efficient one. The basic idea therefore consists of using an (approximation) algorithm \( \text{ALG} \) that is monotonic (see, e.g., Murgolo (1988)): Seemingly favorable changes to the input must not worsen the algorithm’s performance. In the problems considered here, every player is endowed with a size (e.g., processing requirement in the case of scheduling), and reducing a player’s size must not increase the cost of the algorithm’s solution. Provided that this property holds we can then simply number the players in the order of their size such that \( C_{\text{ALG}}(\text{MIN}_{|U|} T) \leq C_{\text{ALG}}(U) \) for all \( U \subseteq T \subseteq [n] \). Finding the most cost-efficient set then only requires iterating through all possible cardinalities.

We generalize this basic idea such that only a (polynomial-time computable) monotonic bound \( C_{\text{mono}} \) on \( C_{\text{ALG}} \) is needed whereas \( \text{ALG} \) itself does not need to be monotonic any more.

Definition 6.1. Let \( \Phi = (\Pi, \text{INST}) \) be a cost-sharing problem. Suppose \( \text{ALG} \) is an approximation algorithm for \( \Pi \), and \( C_{\text{mono}} : 2^{[n]} \to \mathbb{R}_{\geq 0} \) is a cost function that satisfies the following:

- For all \( T \subseteq [n] \): \( C_{\text{ALG}}(T) \leq C_{\text{mono}}(T) \leq \beta \cdot C(T) \).
- For all \( U \subseteq T \subseteq [n] \): \( C_{\text{mono}}(\text{MIN}_{|U|} T) \leq C_{\text{mono}}(U) \).

Then, the pair \( (\text{ALG}, C_{\text{mono}}) \) is called a \( \beta \)-relaxation for \( \Phi \).
Lemma 6.2. Let 

\[ \sigma \text{ only } \]

Note that this recursion is well-defined. Computing moreover, let \( \text{players is recursively as follows. Suppose the set of remaining players is } Q \text{ and the set of already accepted players is } N. \) Let \( \xi \) be the vector of cost shares computed by Algorithm 3 for input \( \sigma, \rho, \) and \( N. \)

Moreover, let

\[ k := \max\left\{ \arg \min_{i \in [Q \cup N]} \left\{ \frac{C_{\text{mono}}(N \cup \text{MIN}_i(Q \setminus N)) - \sum_{i \in N} \xi_i}{C_{\text{mono}}(\text{MIN}_i(Q \setminus N))} \right\} \right\}, \]

and \( S := \text{MIN}_k(Q \setminus N). \) Then, define

\[ \sigma_R(Q, N) := S \quad \text{and} \quad \rho_R(Q, N) := \min \left\{ \frac{C_{\text{mono}}(N \cup S) - \sum_{i \in N} \xi_i}{k}, \frac{C_{\text{mono}}(S)}{k} \right\}. \]

Note that this recursion is well-defined. Computing \( \sigma_R(Q, N) \) and \( \rho_R(Q, N) \) requires \( \xi \) for which only \( \sigma_R(N, \cdot) \) and \( \rho_R(N, \cdot) \) are needed (unless \( N = \emptyset \)). Yet, \( N \not\subseteq Q \) by assumption.

**Lemma 6.2.** Let \( R = (\text{alg}, C_{\text{mono}}) \) be a \( \beta \)-relaxation for some cost-sharing problem \( \Phi. \) Then the following holds:

i) \( \sigma_R \) and \( \rho_R \) are valid.

ii) \( \rho_R \) is \( \beta \)-average for \( C. \)

**Proof.** i) Let \( \sigma := \sigma_R, \rho := \rho_R. \) Let \( \xi \) be the cost-sharing method induced by \( \sigma \) and \( \rho. \) We show that Definition 4.1 holds. Clearly, properties [W1] and [W2] are fulfilled. To see [W3], let \( N \not\subseteq Q' \subseteq Q \subseteq [n]. \) Define \( \Sigma(N) := \sum_{i \in N} \xi_i(N) \) and \( S := \sigma(Q, N), k := |S| \) and \( S' := \sigma(Q', N), k' := |S'|. \) Since \( 1 \leq k' \leq |Q' \setminus N| \leq |Q \setminus N|, \)

\[ \rho(Q, N) \leq \frac{C_{\text{mono}}(\text{MIN}_{k'}(Q \setminus N))}{k'} \leq \frac{C_{\text{mono}}(\text{MIN}_{k'}(Q' \setminus N))}{k'} = \frac{C_{\text{mono}}(S')}{{k'}}. \]

Furthermore,

\[ \rho(Q, N) \leq \frac{C_{\text{mono}}(N \cup \text{MIN}_{k'}(Q \setminus N)) - \Sigma(N)}{k'} \leq \frac{C_{\text{mono}}(N \cup \text{MIN}_{k'}(Q' \setminus N)) - \Sigma(N)}{k'} = \frac{C_{\text{mono}}(N \cup S') - \Sigma(N)}{k'}. \]

Since \( \rho(Q', N) \) is equal to one of these upper bounds, we have \( \rho(Q, N) \leq \rho(Q', N). \)

Finally, to see property [W4] let \( N \not\subseteq Q \subseteq [n] \) and define \( S := \sigma(Q, N), k := |S| \) and \( N' := N \cup S, S' := \sigma(Q', N'), k' := |S'|. \) Then,

\[ \rho(Q, N) \leq \frac{C_{\text{mono}}(\text{MIN}_{k'}(Q \setminus N))}{k'} \leq \frac{C_{\text{mono}}(\text{MIN}_{k'}(Q' \setminus N'))}{k'} = \frac{C_{\text{mono}}(S')}{k'}. \]
Moreover, we have $\text{MIN}_{k+k'}(Q \setminus N) = S \cup S'$. Also, it is easy to see that $\Sigma(N') = \Sigma(N) + k \cdot \rho(Q, N)$ by making use of property [W2] similarly as in first part of the proof of Theorem 5.7.

Consequently,

$$\rho(Q, N) \leq \frac{C_{\text{mono}}(N \cup S \cup S') - \Sigma(N)}{k + k'} = \frac{C_{\text{mono}}(N' \cup S') - \Sigma(N') + k \cdot \rho(Q, N)}{k + k'},$$

implying that

$$\rho(Q, N) \leq \frac{C_{\text{mono}}(N' \cup S') - \Sigma(N')}{k'}.$$

Again, $\rho(Q, N')$ is the minimum of the upper bounds, and therefore $\rho(Q, N) \leq \rho(Q, N \cup \sigma(Q, N'))$.

ii) Let $N \subseteq Q \subseteq [n]$ and $A \subseteq Q \setminus N$. Then,

$$\rho_R(Q, N) \leq \frac{C_{\text{mono}}(\text{MIN}_A((Q \setminus N)))}{|A|} \leq \frac{C_{\text{mono}}(A)}{|A|} \leq \frac{\beta \cdot C(A)}{|A|}. \quad \square$$

To also compute a feasible solution for the instance of the optimization problem that corresponds to the players served by an egalitarian mechanism, we need:

**Definition 6.3.** Let $\Phi = (\Pi, \text{Inst})$ be a cost-sharing problem where $\Pi = (D, S, f)$. Then, $\Phi$ is called mergable if for all disjoint $T, U \subseteq [n]$ and for all $X \in S_{\text{Inst}(T)}$ and $Y \in S_{\text{Inst}(U)}$, there is a $Z \in S_{\text{Inst}(T \cup U)}$ with $f(Z) \leq f(X) + f(Y)$. We denote this operation by $Z = X \oplus Y$.

Based on $\sigma_R$ and $\rho_R$, **Algorithm 5** completely solves the cost-sharing problem, including computing a feasible solution for the underlying optimization problem. In the following, we verify correctness.

**Algorithm 5:** Egalitarian mechanisms with $\beta$-relaxations
Lemma 6.4. Let $R = (\text{ALG}, C_{\text{mono}})$ be a $\beta$-relaxation for a mergable cost-sharing problem $\Phi$. The following holds:

i) At the end of each iteration of Algorithm 5, it holds that $x = \xi(N)$ where $\xi$ is the cost-sharing method defined by Algorithm 5 for input $\sigma_R$ and $\rho_R$.

ii) In every iteration, line 3 of Algorithm 5 needs at most $2n$ evaluations of $C_{\text{mono}}$.

iii) The mechanism defined by Algorithm 5 is $\beta$-BB.

Proof. Consider the execution of Algorithm 5 for input $R$ and $b$. Let $m \in \mathbb{N}$ be the number of iterations needed. For all $k \in [m]_0$, indicate the value of all variables at the end of the $k$-th iteration (i.e., immediately after line 10 if $k > 0$, and immediately before line 2 if $k = 0$) with a superscript $k$. Moreover, let $p(k)$ be the number of times line 5 has evaluated to true before the end of iteration $k$.

i) We show that $\xi(N^k)$ is computed in exactly the same way as $x^k$ was. Consider therefore the execution of Algorithm 3 for input $\sigma_R$, $\rho_R$, and $N^k$. Indicate the value of all variables at the end of the $j$-th iteration with a tilde and a superscript $j$. Again, superscript 0 refers to the variable values immediately before the while-loop.

We prove by induction over $k \in [m]_0$ that $\tilde{N}^p(k) = N^k$ and $\tilde{\xi}^p(k) = x^k$. Clearly, the base case $k = 0$ is fulfilled because $\tilde{N}^0 = N^0 = 0$ and $\tilde{\xi}^0 = x^0 = 0$. In the following, we verify the induction step $(k - 1) \rightarrow k$. If line 5 of Algorithm 5 evaluated to false then $p(k) = p(k - 1)$, $N^k = N^{k-1}$, and $x^k = x^{k-1}$, which proves the induction step for this case.

Consider therefore the case that line 5 evaluated to true. Then, $p(k) - 1 = p(k - 1)$. We have that $\tilde{S}^p(k) = \sigma(\tilde{Q}^p(k), \tilde{N}^{p(k)-1}) = \sigma(N^k, N^{k-1})$ where the last inequality is due to the induction hypothesis. Moreover, we have $S^k = \sigma(Q^{k-1}, N^{k-1}) = \sigma(N^k, N^{k-1})$ where the last equality is due to (W2) and $S^k \subseteq N^k$. Hence, $\tilde{S}^p(k) = S^k$ and likewise $\tilde{a}^p(k) = a^k$. The induction step follows.

ii) This follows directly from the definition of $\sigma_R$ and $\rho_R$.

iii) Define $\Sigma(N^k) := \sum_{i \in N^k} x^k_i$. We show by induction over $k \in [m]_0$ that $f(Z^k) \leq \Sigma(N^k) \leq C_{\text{mono}}(N^k)$.

The base case $k = 0$ holds trivially. For the induction step $(k - 1) \rightarrow k$, we only need to consider the case that line 5 evaluated to true in iteration $k$. Otherwise, $Z^k = Z^{k-1}$ and $N^k = N^{k-1}$, so we would be done. Now, if $C_{\text{mono}}(N^{k-1} \cup S^k) - \Sigma(N^{k-1}) \leq C_{\text{mono}}(S^k)$ then $f(Z^k) = C_{\text{ALG}}(N^k) \leq C_{\text{mono}}(N^k) = \epsilon(N^k)$. Otherwise, $f(Z^k) = f(Z^{k-1} \oplus \text{ALG(INST}(S^k)))$ 

\[ \leq f(Z^{k-1}) + C_{\text{ALG}}(S^k) \leq \Sigma(N^{k-1}) + C_{\text{mono}}(S^k), \]

where the last inequality is due to the induction hypothesis and because $C_{\text{ALG}}$ is a lower bound for $C_{\text{mono}}$. Now, $\Sigma(N^k) = \Sigma(N^{k-1}) + C_{\text{mono}}(S^k)$. Since line 5 evaluated to false, we moreover have $\Sigma(N^{k-1}) + C_{\text{mono}}(S^k) < C_{\text{mono}}(N^{k-1} \cup S^k) = C_{\text{mono}}(N^k)$. Hence, the induction step follows.

Clearly, the output of Algorithm 5 is $Q^m = N^m$, $x^m$, and $Z^m$. We have shown that $C'(Q^m) = f(Z^m) \leq \Sigma(Q^m) \leq C_{\text{mono}}(Q^m) \leq \beta \cdot C(Q^m)$. This completes the proof. $\square$
As a corollary of Lemmata 6.2 and 6.4, we obtain:

**Theorem 6.5.** Let $\Phi$ be a mergable cost-sharing problem, and let $(\text{ALG}, C_{\text{mono}})$ be a $\beta$-relaxation for $\Phi$. Then the mechanism defined by Algorithm 5 is SGSP, $\beta$-BB, and $(2\beta \cdot H_n)$-EFF. Moreover, Algorithm 5 evaluates $C_{\text{mono}}$ for no more than $2n^2$ subsets of $[n]$, makes no more than $n$ (direct) calls to ALG, and the number of merge operations is no more than $n$.

7 Applications

We use three approaches for obtaining $\beta$-relaxations that are polynomial-time computable in the size of the succinct representation of the cost-sharing problem plus the bid vector: Monotonic approximation algorithms (Section 7.2), a non-monotonic approximation algorithm with a polynomial-time computable monotonic bound $C_{\text{mono}}$ (Section 7.3), and optimal costs that are monotonic and polynomial-time computable (Section 7.4). Subsequently, we also give some remarks about applying sequential stand-alone mechanisms to cost-sharing problems with supermodular optimal costs (Section 7.5).

The combinatorial optimization problems that we consider are bin packing, where the task is to pack $n$ items into the least possible number of unit-sized bins, and machine scheduling, where the task is to schedule $n$ jobs on $m$ parallel machines. In the following, we assume at least a nodding acquaintance with these problems (see, e.g., Brucker (2007); Hochbaum (1997)).

In order to distinguish between the various variants of scheduling problems, we make use of the three-field notation $|\alpha|\beta|\gamma$ introduced by Graham et al. (1979): The field $\alpha$ represents the machine environment: E.g., “1” denotes a single machine, “$P$” identical parallel machines, “$Q$” related parallel machines, and an optional $m \in \mathbb{N}_{\geq 2}$ after $P$ or $Q$ denotes that the number of machines is $m$ and thus a constant. The field $\beta$ defines job characteristics: $p_i$ means that all processing requirements are identical, $r_i$ means that the jobs may have release times (earliest starting time), and “pmtn” indicates that preemption is allowed. Finally, the field $\gamma$ refers to the objective function. E.g, $C_{\text{max}}$ is the makespan, $\sum C_i$ is the sum of completion times, and $\sum w_i C_i$ is the weighted sum of completion times.

7.1 Approximations for Makespan and Bin Packing Cost-Sharing Problems

A makespan cost-sharing problem $Q||C_{\text{max}}$ is succinctly represented by a pair $(p, s)$ where $p \in \mathbb{N}^n$ contains the processing requirements of the $n$ jobs, and $s \in \mathbb{N}^m$ contains the speeds of the $m$ machines. Each player owns exactly one job and for any set of players $S \subseteq [n]$, $C(S)$ is the value of a minimum-makespan schedule for the jobs from $S$.

A bin packing cost-sharing problem is succinctly represented by a vector of item sizes $\varsigma \in (0, 1]^n$. Each player owns exactly one item, and for any set of players $S \subseteq [n]$, $C(S)$ is the minimum number of bins with capacity 1 that are needed for $S$.

In order to keep our notation clean when designing $\beta$-relaxations that fulfill Definition 6.1, we assume in this section that players’ indices are always sorted in ascending order of their processing requirements (in the case of scheduling) or item sizes (in the case of bin packing). This is without loss of generality: Otherwise, players could be sorted (in a deterministic way) before Algorithm 5 is called, which adds only $O(\log n)$ to the running time and is thus always negligible.

**Lemma 7.1.** Any bin packing or makespan cost-sharing problem $\Phi = (\Pi, \text{INST})$ is mergable in time $O(n)$. Moreover, INST is computable in linear time (in the size of the succinct representation of $\Phi$).
Lemma 7.3. Let $f$ be an approximation algorithm for the makespan optimization problem on identical machines. Consequently, for any instance $(p, m)$, we have $f(lpt(p, m)) \leq f(lpt(p', m))$.

Proof. For bin packing with disjoint item/player sets $T$ and $U$, we obtain a bin packing for $T \cup U$ by taking both the bins with items from $T$ and the bins with items from $U$. The costs (number of bins) simply add up. For scheduling disjoint job/player sets $T$ and $U$, we obtain a schedule for $T \cup U$ by assigning each job to the machine assigned before. The resulting makespan doesn’t exceed the sum of the two makespans. □

7.2 Monotonic Approximation Algorithms

Makespan Costs on Identical Machines We start by considering identical-machine makespan cost-sharing problems $(P||C_{max})$. Their succinct representation is $(p, m)$ where $p \in \mathbb{N}^n$ and $m \in \mathbb{N}$. The LPT (longest processing time first) algorithm (Graham 1969) is known to be a $\frac{4m-1}{3m}$-approximation algorithm for this problem. It processes the jobs in decreasing order and assigns each job to the machine on which its completion time will be smallest. Its running time is $O(n \cdot \log n)$ for the sorting phase and $O(n \cdot \log m)$ for the job assignment phase. For identical machines, we show that LPT is monotonic with regard to processing requirements. In order to do so, we first need a technical lemma. Let $\text{SORT}$ denote a function that sorts the components of a vector in ascending order.

Lemma 7.2. Let $a, b \in \mathbb{R}^n$ be vectors whose components are sorted in ascending order. Moreover, let $c, d \in \mathbb{R}$ and define $a' := \text{SORT}(a_{-1}, a_1 + c)$ and $b' := \text{SORT}(b_{-1}, b_1 + d)$. Suppose that $a \leq b$ and $c \leq d$; then it holds that $a' \leq b'$.

Proof. Let $j, k \in [n]$ be arbitrary with $a'_j = a_1 + c$ and $b'_k = b_1 + d$. Note that a value may occur several times in the vector. By definition,

$$a' = (a_2, a_3, \ldots, a_j, a_1 + c, a_{j+1}, \ldots, a_n) \quad \text{and} \quad b' = (b_2, b_3, \ldots, b_k, b_1 + d, b_{k+1}, \ldots, b_n).$$

Now let $i \in [n]$ be arbitrary. We verify that $a'_i \leq b'_i$. Note that

$$a'_i = \begin{cases} 
    a_{i+1} & \text{if } i < j \\
    a_1 + c \in [a_i, a_{i+1}] & \text{if } i = j \\
    a_i & \text{if } i > j
  \end{cases} \quad \text{and} \quad b'_i = \begin{cases} 
    b_{i+1} & \text{if } i < k \\
    b_1 + d \in [b_i, b_{i+1}] & \text{if } i = k \\
    b_i & \text{if } i > k.
  \end{cases}$$

Consequently, $b'_i \geq b_i$ and, if $i < n$, then $a_{i+1} \geq a'_i$. By case analysis, we get:

- Case $i \geq k$ and $i \leq j$: Then $b'_i \geq b'_k = b_1 + d \geq a_1 + c = a'_j \geq a'_i$.

- Case $i \geq k$ and $i > j$: Then $b'_i \geq b_i \geq a_i = a'_i$.

- Case $i < k$: Then $b'_i = b_{i+1} \geq a_{i+1} \geq a'_i$. □

Lemma 7.3. Let $n, m \in \mathbb{N}$, $i \in [n]$, and $p, p' \in \mathbb{N}^n$ be i-variants with $p_i < p'_i$. Then $f(lpt(p, m)) \leq f(lpt(p', m))$.
Algorithm 5 runs in time \( \Sigma \) which is known to produce an optimal packing for this modified instance. The overall running-time when given as input to Algorithm 5 is \( O(n \cdot \log m) \).

Lemma 7.5. For any bin packing cost-sharing problem with succinct representation \( \varsigma \), where \( \varsigma_1 \leq \cdots \leq \varsigma_n \), there is a 2-relaxation for \( C \) and Algorithm 5 runs in time \( O(n^3 \cdot \log n) \).

Proof. Since RFFD is a monotonic 2-approximation algorithm, it holds that \( (\text{RFFD}, C_{\text{RFFD}}) \) is a 2-relaxation. The overall running-time when given as input to Algorithm 5 is \( O(n^3 \cdot \log n) \). \( \square \)
Remark. The tight bound of $\text{ffd}$ is $\frac{11}{9} \cdot \text{opt} + \frac{6}{9}$, (Dósa 2007). However, $\text{ffd}$ is not monotonic: Let $\varsigma := (\frac{9}{17}, \frac{9}{17}, \frac{5}{17}, \frac{5}{17}, \frac{4}{17}, \frac{4}{17}, \frac{4}{17}, \frac{3}{17}, \frac{3}{17})$ and $\varsigma' := (\frac{5}{17}, \frac{5}{17})$. Then, the corresponding costs are $f(\text{ffd}(\varsigma)) = 3$ and $f(\text{ffd}(\varsigma')) = 4$.

Remark. It is known that the NFD (next fit decreasing) algorithm is monotonic (Murgolo 1988) and a 2-approximation algorithm for the bin packing problem. Hence, also $(\text{NFD}, C_{\text{NFD}})$ is a 2-relaxation.

Theorem 7.6. For any related-machine makespan cost-sharing problem $(Q||C_{\text{max}})$ with succinct representation $(p, s)$, where $p_1 \leq \cdots \leq p_n$, there is a 2-relaxation. Algorithm 5 runs in time $O(n^3 \cdot \log m \cdot \log \sum_{i \in [n]} p_i)$.

Proof. Consider the decision variant of the following modified bin packing problem: Given $n \in \mathbb{N}$ items with sizes $\varsigma \in \mathbb{Q}_{>0}^n$ and $m \in \mathbb{N}$ bins with capacities $c \in \mathbb{Q}_{>0}^m$, decide whether all $n$ items fit into the $m$ bins. Let $\text{ffd}^*$ be the following algorithm: Run $\text{ffd}$. That is, in descending order of item sizes, put every item into the first bin where it fits. Note here that no changes are necessary to account for the variable bin capacities. If, at some point an item does not fit any more, return “false”. Otherwise, return “true”.

We show that $\text{ffd}^*$ is optimal when item sizes are divisible, meaning that every item size is exactly divided by any smaller item size (cf. also Coffman, Jr. et al. (1987)). Let $n, m \in \mathbb{N}$, $\varsigma \in \mathbb{Q}_{>0}^n$, be a vector of divisible item sizes, and $c \in \mathbb{Q}_{>0}^m$ be the capacity vector. W.l.o.g., assume $\varsigma_1 \geq \cdots \geq \varsigma_n$ here. Suppose item $j$ is the first item that does not fit any more into one of the $m$ bins, i.e., $\text{ffd}^*(\varsigma, c) = \text{false}$. This means that the remaining space in any bin is less than $\varsigma_j$.

Now note that, for every packing of the first $(j-1)$ items, the filling level in each bin is always a multiple of $\varsigma_j$. Consequently, if in some other bin packing there was still space for item $j$, there would also be a bin for which the filling level exceeds the bin capacity. This is a contradiction and proves optimality of $\text{ffd}^*$ for divisible item sizes.

Now let $\text{rffd}^*$ denote the algorithm that first rounds each item size up to the next power of 2 and then calls $\text{ffd}^*$. Due to the observation that $\text{ffd}^*$ is optimal for divisible item sizes, we know that $\text{rffd}^*$ is monotonic in the item sizes. In order to obtain a 2-relaxation for makespan minimization, we can employ $\text{rffd}^*$ together with binary search (compare also Algorithm 7): Within trivial upper and lower bounds, search for the minimum makespan $d$ so that $\text{rffd}^*(\mathbb{P}, \mathbb{s}) = \text{true}$.

7.3 Non-Monotonic Approximation Algorithms with a Polynomial-Time Computable Monotonic Bound

Besides the previous result, we also show how to adapt the PTAS for identical machines $(P||C_{\text{max}})$ by Hochbaum and Shmoys (1987). Although the running time of the PTAS is prohibitive for any
small $\epsilon$, the result is theoretically interesting: First, any fixed budget balance greater than 1 can be achieved in polynomial time. Second, the approach here is different to before: Not the PTAS itself is monotonic but only a bound computed inside the algorithm.

The basic idea of the PTAS is a reduction to bin packing (see Algorithm 6): Given processing requirements $p \in \mathbb{N}^n$, binary search between trivial upper and lower bounds is employed in order to find a makespan $d$ such that the bin packing instance $\frac{p}{d}$ does not need more than $m$ bins of capacity $(1 + \epsilon)$, whereas the bin packing instance $\frac{p}{d-\epsilon}$ does need more than $m$ bins. Specifically, the PTAS makes use of $\text{bpDual}_\epsilon$, which is an $\epsilon$-dual approximation algorithm for the bin packing problem [Hochbaum and Shmoys 1987, pp.149–151]. For completeness, it is shown in Algorithm 6 $\text{bpDual}_\epsilon$ outputs solutions that are dual feasible; this means that $\text{bpDual}_\epsilon$ uses bins of capacity $(1 + \epsilon)$ but never needs more bins than the feasible optimal solution (with capacity 1).

**Input:** approximation $\epsilon \in (0, 1)$; item size vector $\varsigma \in (0, 1]^n$

**Output:** allocation $a \in \mathbb{N}^n$

1. Partition the interval $[\epsilon, 1]$ of large sizes into $s := \lceil \frac{1}{\epsilon} \rceil$ equal-length subintervals $(l_i, l_{i+1}]$. Use $l_i$ as rounded size for all original sizes in this interval.
2. Determine all feasible configurations $(x_1, \ldots, x_s) \in \mathbb{N}_0^s$ defined by $\sum_{i=1}^s x_i \cdot l_i \leq 1$ (where $x_i$ is number of items with size in the interval $(l_i, l_{i+1}]$)
3. Use dynamic programming to find an allocation of the large items (using rounded sizes; excluding original sizes $\leq \epsilon$), based on following the recurrence:

$$
\text{Bins}(y_1, \ldots, y_s) := 1 + \min_{\left( \begin{array}{c} x_1, \ldots, x_s \\ \text{is feasible configuration} \end{array} \right)} \{ \text{Bins}(y_1 - x_1, \ldots, y_s - x_s) \}
$$

$\text{Bins}(y_1, \ldots, y_s)$ is the minimum number of bins needed when there are $y_i$ pieces of size $l_i$.
4. Enlarge bins to $1 + \epsilon$ and go back to original sizes
5. Pack small items with original size $\leq \epsilon$ into an arbitrary bin containing $\leq 1$. If no such bin exists, open a new bin. Let $a$ denote the final allocation of items to bins.

**Algorithm 6:** $\epsilon$-dual approximation algorithm for bin packing

Now, for any bin packing instance $\varsigma \in (0, 1]^n$, let $S^*_\varsigma \supseteq \varsigma$ be the set of all dual-feasible solutions and $f^*_\varsigma : S^*_\varsigma \rightarrow \mathbb{N}$ be a function mapping each dual-feasible solution to its cost, i.e., to the number of used bins. We define $g^*_\varsigma : S^*_\varsigma \rightarrow \mathbb{N}$ by $g^*_\varsigma(Z) := \max\{f^*_\varsigma(Z), \lceil \sum_{i \in [n]} S_i \rceil \}$. Hence, the crucial property of $g^*_\varsigma$ is to guarantee that $g^*_\varsigma$ is never less than the total size of all items. We show that $g^*$ is monotonic.

**Lemma 7.7.** Let $\varsigma, \varsigma' \in \mathbb{Q}^n_{\geq 0}$ be two vectors of item sizes, $i \in [n]$, $\varsigma_i > \varsigma'_i$, and $\varsigma_{-i} = \varsigma'_{-i}$. Then $b := g^*_\varsigma(\text{bpDual}_\epsilon(\varsigma)) \geq g^*_{\varsigma'}(\text{bpDual}_\epsilon(\varsigma')) =: b'$.

**Proof.** By way of contradiction, assume $b < b'$. Now consider an execution of $\text{bpDual}_\epsilon$ for input $\varsigma'$ (see Algorithm 6). All items of size $< \epsilon$ are called “small”. Other items of size $\geq \epsilon$ are called “large”. In the first phase, round each of the sizes of the large items to one of constantly many sizes and solve this rounded instance optimally without the small items. Afterwards in the second phase, go back to original sizes and pack small items one after the other into an arbitrary bin containing $\leq 1$. If no such bin exists, open a new bin.

35
Since the first phase computes optimal solutions of rounded instances, it is monotonic. Hence, \( b < b' \) implies that only in the last phase where the small items are packed, the \((b+1)\)-th bin is opened. More precisely, according to line 5 of Algorithm 6, there must be a point where \( b \) bins are used but all bins contain more than 1. However, this means that \( \sum_{i \in [n]} \epsilon' \geq b \), i.e., the total size of all items of instance \( \epsilon' \) is more than \( b \). Specifically, \( b = g^*_\epsilon(b_{\text{Dual}}(\epsilon)) \geq \lceil \sum_{i \in [n]} s_i \rceil \geq \sum_{i \in [n]} s_i > \sum_{i \in [n]} \epsilon_i > b \). A contradiction.

Algorithm 7 contains the PTAS, together with a crucial extension in line 6. Note that this line is not necessary for the approximation guarantee, but only needed for monotonicity. For \( n, m \in \mathbb{N} \) and \( p \in \mathbb{N}^n \), define \( \text{SIZE}(p, m) := \max \left\{ \frac{1}{m} \cdot \sum_{i \in [n]} p_i, p_1, p_2, \ldots, p_n \right\} \).

\begin{algorithm}
Input: approximation \( \epsilon \in (0, 1) \);
vector \( p \in \mathbb{N}^n \) of processing requirements; number of machines \( m \in \mathbb{N} \)
Output: allocation \( a \in [m]^n \), lower bound on optimum makespan \( \text{lower} \)
1: \( \text{upper} := 2 \cdot \text{SIZE}(p, m) \)
2: \( \text{lower} := \text{SIZE}(p, m) \)
3: while \( \text{upper} \neq \text{lower} \) do
4: \( d := \lfloor (\text{upper} + \text{lower})/2 \rfloor \)
5: \( a := b_{\text{Dual}}(\frac{p}{2}) \); set \( b \) to number of bins used in \( a \)
6: \( b := \max \{ b, \lceil \sum_{i \in [n]} \frac{p_i}{d} \rceil \} \) \hspace{1cm} \triangleright \text{Crucial extension for monotonicity}
7: if \( b > m \) then
8: \( \text{lower} := d + 1 \) \hspace{1cm} \triangleright \text{ Afterwards, still lower} \leq d \leq \text{upper} \)
9: else
10: \( \text{upper} := d \) \hspace{1cm} \triangleright \text{ Afterwards, still lower} \leq d \leq \text{upper} \)
11: \( a := b_{\text{Dual}}(\frac{p}{\text{lower}}) \) \hspace{1cm} \triangleright \text{Not necessary if} \ b \leq m \)

\end{algorithm}

Algorithm 7: Modified PTAS for the minimum makespan problem

Note that \( \text{lower} \) in Algorithm 7 is always a lower bound on the optimal makespan: Since \( b_{\text{Dual}}(\epsilon) \) is an \( \epsilon \)-dual approximation algorithm, this holds at the beginning and also whenever \( \text{lower} \) is updated in line 8. On the other hand, \( \text{upper} \cdot (1 + \epsilon) \) is always an upper bound both on the optimal makespan as well as on the makespan of the schedule \( a \). Our crucial extension of the PTAS is as follows: Letting \( \epsilon := \frac{p}{2} \), we use the check \( g^*_\epsilon(\text{BP}_{\text{Dual}}(\epsilon)) \leq m \) in the binary search (instead of testing \( f^*_\epsilon \) as in the original PTAS).

Let \( \text{lower}_\epsilon(p, m) \) denote the final value of \( \text{lower} \) returned by Algorithm 7 for input \( \epsilon, p, \) and \( m \). That is, \( \text{lower}_\epsilon(p, m) \) is the minimum \( d \) for which the check \( g^*_\epsilon(\text{BP}_{\text{Dual}}(\epsilon)) \leq m \) evaluates to true. Moreover, let \( \text{HS}_\epsilon \) denote the adapted PTAS. Now, \( \text{lower}_\epsilon(p, m) \) is a lower bound on the optimal makespan and \( (1 + \epsilon) \cdot \text{lower}_\epsilon(p, m) \) is an upper bound on the makespan of the schedule found by \( \text{HS}_\epsilon \). Moreover, \( \text{lower}_\epsilon(p) \) is computed within \( \text{HS}_\epsilon \) in polynomial time because monotonicity of \( g^* \) ensures that indeed the minimum \( d \) is found by the binary search. As a corollary of Lemma 7.7 we get:

**Theorem 7.8.** Let \( \Phi \) be an identical-machine makespan cost-sharing problem \( (P||C_{\text{max}}) \) with succinct representation \( (p, m) \), where \( p_1 \leq \cdots \leq p_n \). Define the monotonic cost function \( C_{\text{mono}}(A) := (1 + \epsilon) \cdot \text{lower}_\epsilon(\text{INST}(A)) \). Then, \( (\text{HS}_\epsilon, C_{\text{mono}}) \) is a \((1 + \epsilon)\)-relaxation for \( \Phi \), and Algorithm 3 runs in time \( O(n^{2 + \frac{1}{\epsilon^2}} \cdot \log \sum_{i \in [n]} p_i) \).
7.4 Makespan Problems with Monotonic Optimal Costs

There are several mergable makespan problems for which optimal costs are monotonic and computable in polynomial time. For instance, for the problem of scheduling identical jobs on identical parallel machines \((P|p_i = p|C_{\text{max}})\), it holds that \((\text{LPT}, C_{\text{LPT}})\) is a 1-relaxation and Algorithm 5 runs in time \(O(n^3 \cdot \log m)\). In the following, we give a selection of further such problems (see, e.g., Brucker (2007)):

- Symmetric costs:
  \(- Q|p_i = p|C_{\text{max}}\)

- Variable release dates:
  \(- Q|p_i = p, r_i|C_{\text{max}}\)
  \(- Q|\text{pmtn}, p_i = p, r_i|C_{\text{max}}\)

- Variable processing requirements:
  \(- Q|\text{pmtn}|C_{\text{max}}\)

It is straightforward to see that all of the induced (optimal) cost functions are subadditive and the problems are mergable. This holds as well for the preemptive case. Moreover, the optimal costs are always monotonic in the variable property (release dates or processing requirements) so that determining the most cost-efficient set can always be done in polynomial time by only checking a single set for each cardinality (see Section 6). If jobs are ordered by increasing value of the variable property, the first \(k\) jobs minimize the cost over all sets of cardinality \(k\). Consequently, we get that 1-relaxations exist for all of the above problems.

**Theorem 7.9.** For sharing the (optimal) cost induced by any of the above makespan problems, there is a 1-BB and \(2H_n\)-EFF egalitarian mechanism. Its outcome can be computed in polynomial time.

7.5 Scheduling Problems with Supermodular Costs

We find it interesting to note that Brenner and Schäfer’s singleton mechanisms (Brenner and Schäfer, 2008) for \(P||\sum C_i\) and \(1||\sum w_iC_i\) are in fact egalitarian mechanisms based on most cost-efficient set selection: For these problems, the induced (optimal) cost functions are supermodular (see, e.g., Schulz and Uhan (2007)). Moreover, for any set of remaining players \(Q\) and any set of already accepted players \(N\), a player \(i \in Q \setminus N\) with minimal \(w_i/p_i\) constitutes a most cost-efficient set—and indeed, singleton mechanisms always choose one of these singleton sets. In particular, the assigned cost shares of each job are equal to the completion time under Smith’s rule (Smith, 1956), an algorithm which assigns the jobs in the order of increasing ratios \(w_i/p_i\) and which is known to deliver optimal schedules for the above problems in polynomial time.

For \(P||\sum w_iC_i\), computing optimal costs is NP-hard, but Smith’s rule guarantees an approximation ratio of \((1 + \sqrt{2})/2 \approx 1.21\), (Kawaguchi and Kyan, 1986). It is easy to verify that the costs induced by Smith’s rule are supermodular. Somewhat unsurprisingly now, the egalitarian mechanisms induced by always choosing the most cost-efficient set with respect to this approximation cost are again equivalent to the singleton mechanisms by Brenner and Schäfer (2008) for this problem.
Since the order in which players are offered prices is constant, the above mechanisms are in fact even sequential stand-alone mechanisms. Thus, all of the previously mentioned subclasses of the acyclic-mechanism framework coincide here in a natural way.

For the objective to minimize the completion time on identical parallel machines, Brenner and Schäfer (2008) showed that optimal costs are 2-subadditive. Yet, it is a simple observation that the very same proof holds also for related parallel machines:

**Proposition 7.10 (Brenner and Schäfer (2008)).** Let $C$ be the optimal cost function induced by $Q||\sum w_iC_i$. Then, $C$ is 2-subadditive.

Now recall Theorem 5.10 stating that when $\beta$-approximate costs are supermodular and $\alpha$-subadditive, simple sequential stand-alone mechanisms guarantee $\beta$-BB and $(\alpha \cdot \beta)$-EFF (regardless of the order of the players). This implies, of course, that the above mechanisms for $P||\sum C_i$ and $1||\sum w_iC_i$ are 1-BB and 2-EFF and the mechanisms for $P||\sum w_iC_i$ are 1.21-BB and 2.42-EFF, as previously shown by Brenner and Schäfer (2008). Another polynomial-time solvable problem with supermodular optimal costs is $Q|p_i=1|\sum w_iC_i$, (Schulz and Uhan, 2007). We therefore obtain the following new result:

**Theorem 7.11.** For sharing the (optimal) cost induced by $Q|p_i=1|\sum w_iC_i$, every sequential stand-alone cost-sharing mechanism is 1-BB, 2-EFF, and computable in polynomial time.

As a last remark, the recovered costs of a cross-monotonic cost-sharing methods are always subadditive. Consequently, it is not surprising that Moulin mechanisms suffer bad budget balance if the underlying optimization problem severely violates subadditivity. E.g., for the problem $1||\sum_i C_i$, no cross-monotonic cost-sharing method can be better than $\frac{n+1}{2}$-BB (Brenner and Schäfer, 2007). Obviously, the SGSP mechanisms discussed above tremendously improve on this.

8 Conclusion and Future Work

The pivotal point of this work was to study cost-sharing scenarios where the case that a player feels indifferent about being served is negligible. We believe that SGSP (or one of the other collusion-resistance properties without indifferences) is often a viable replacement for the often too limiting GSP requirement. We consider the main asset of our work to be fourfold: (a) Characterizing the relationship between the new collusion-resistance properties. (b) Egalitarian mechanisms; showing existence of SGSP, 1-BB, and $2H_n$-EFF mechanisms for any non-decreasing subadditive costs. (c) Our framework for polynomial-time computability that reduces constructing SGSP, $O(1)$-BB, and $O(\log n)$-EFF mechanisms to finding monotonic approximation algorithms. (d) Showing that acyclic mechanisms are robust against the scheme underbidders are removed; as a consequence, they comprise egalitarian mechanisms and are SGSP—i.e., in a precise sense, they are remarkably stronger than was known before.

An immediate issue left open by our work is, of course, to find more applications of our polynomial-time framework. For instance, it is easy to see that (rooted) Steiner tree cost-sharing problems are mergable and their costs non-decreasing and subadditive; but do they allow for a $\beta$-relaxation?
References


