Generalized Incremental Mechanisms for Scheduling Games*

Janina Brenner† Guido Schäfer‡

Abstract

We study the problem of designing truthful mechanisms for cooperative cost sharing games that realize (approximate) budget balance and social cost. Recent negative results show that group-strategyproof mechanisms can only achieve very poor approximation guarantees for several fundamental cost sharing games. Driven by these limitations, we consider cost sharing mechanisms that realize the weaker notion of *weak group-strategyproofness*. Mehta et al. [*Games and Economic Behavior*, 67:125–155, 2009] recently introduced the broad class of weakly group-strategyproof *acyclic mechanisms* and show that several primal-dual approximation algorithms naturally give rise to such mechanisms with good approximation guarantees. In this paper, we provide a simple yet powerful approach that enables us to turn any ρ-approximation algorithm into a ρ-budget balanced acyclic mechanism. We demonstrate the applicability of our approach by deriving weakly group-strategyproof mechanisms for several fundamental scheduling problems that outperform the best possible approximation guarantees of Moulin mechanisms. The mechanisms that we develop for completion time scheduling problems are the first mechanisms that achieve constant budget balance and social cost approximation factors. Interestingly, our mechanisms belong to the class of *generalized incremental mechanisms* proposed by Moulin [*Social Choice and Welfare*, 16:279–320, 1999].

1 Introduction

Algorithmic Mechanism Design lies at the intersection of Operations Research and Mechanism Design in that it unites ideas and methodologies from both research areas. It is about the development of mechanisms (or algorithms) that realize game-theoretical objectives and at the same time are computationally efficient. One of the most fundamental questions in this context is to which extent the restriction of polynomial-time computability influences the feasibility of natural game-theoretic objectives such as truthfulness, efficiency, etc. As an example, consider a combinatorial auction in which m indivisible items are to be auctioned off to n players. It is well known that the VCG mechanism due to Vickrey [46], Clarke [12], and Groves [22] is

*A preliminary version of this paper appeared in the *Proceedings of the 1st International Symposium on Algorithmic Game Theory*, volume 4997 of Lecture Notes in Computer Science, pages 315–326, 2008. This work was supported by the DFG Research Center Matheon "Mathematics for key technologies".

†Technische Universität Berlin, Institut für Mathematik, Straße des 17. Juni 136, 10623 Berlin, Germany. Email: brenner@math.tu-berlin.de.

‡Center for Mathematics and Computer Science (CWI), Algorithms, Combinatorics and Optimization, Science Park 123, 1098 XG Amsterdam, The Netherlands. Email: g.schaefer@cwi.nl.

truthful and optimizes social welfare even in the most general multi-parameter setting, where players may value every possible subset of items differently. However, no truthful mechanism that is additionally required to run in polynomial time can approximate social welfare by a factor of less than $\sqrt{m}$ in general (unless NP $\subseteq$ ZPP). This result holds even in the very restricted single-minded setting, where every player is only interested in receiving a specific subset of items (we refer the reader to [34] for more details).

1.1 Cost Sharing Games

In this paper, we address the question mentioned above in the context of cooperative cost sharing games: We consider the problem of designing truthful mechanisms for cost sharing games arising from combinatorial optimization problems, with a particular focus on scheduling problems. In this setting, we are given a set of n players that are interested in a common service, e.g., being connected to a network infrastructure, or being processed on a supercomputer, etc. The provision of the service incurs some cost that is specified by a (player-set dependent) cost function. Every player announces a bid which represents the maximum price he is willing to pay for the service. Based on these bids, a cost sharing mechanism needs to decide which players receive the service and at what price. Each player's valuation for the service is private data only known to the player himself. We assume that every player acts strategically in that he solely aims for maximizing his own (quasi-linear) utility function. As a consequence, a player may declare a false valuation if this is advantageous to him. We consider the setting in which players can form coalitions in order to coordinate their bids and collectively attempt to manipulate the outcome of the mechanism.

We are primarily interested in mechanisms that meet the following objectives:

1. *Computational efficiency*: The mechanism runs in polynomial time.
2. *Truthfulness*: The selection and payment scheme implemented by the mechanism guarantee that it is a dominant strategy for every player to reveal his private valuation, even if players form coalitions.
3. *Budget balance*: The sum of all payments charged to the players is equal to the cost to establish the service.
4. *Social cost*: Assuming that every player bids truthfully, the servicing cost of the selected players plus the sum of the valuations of the excluded players is minimized.

Several natural cost sharing games exhibit cost functions that are given implicitly by the optimal solution cost of an underlying optimization problem. For most of the optimization problems that we consider here, achieving the budget balance or social cost objectives exactly is tantamount to solving NP-hard optimization problems. Since we insist on computational efficiency, we therefore relax these objectives and only require that they are met approximately (formal definitions will be given in Section 2). Our main concern is to derive mechanisms that guarantee good approximation factors for Objectives 3 and 4 and at the same time satisfy Objectives 1 and 2.

1.2 Scheduling Problems

Scheduling problems constitute an important class of optimization problems with various applications in practice. They have been the subject of intensive research in

Operations Research for decades. Many fundamental scheduling problems are rather well-understood with respect to their complexity and approximability.

As an example, consider the *minimum weighted completion time scheduling problem without preemption* $P||\sum_i w_i C_i$. (We assume that the reader is familiar with the three-field notation scheme by Graham et al. [19].) Here the task is to schedule n jobs non-preemptively on m parallel machines such that the (weighted) sum of the completion times of all jobs is minimized. Lenstra proved that this problem is NP-hard (see [9]). Afrati et al. [1] gave a polynomial-time approximation scheme (PTAS) for this problem. A simple scheduling rule, known as Smith's rule [44], schedules jobs by non-increasing weight per processing time ratios. Smith's rule achieves an approximation factor of $(1+\sqrt{2})/2 \approx 1.21$ for this problem. For unit processing times or equal weights, Smith's rule yields an optimal schedule.

The *minimum completion time scheduling problem with release dates and preemption* $P|r_i, pmtn|\sum_i C_i$ is another classical scheduling problem. The goal is to schedule n jobs on m parallel machines such that the total completion time is minimized. Each job becomes available at its release date r_i and jobs can be preempted, i.e., interrupted and resumed later. The problem is known to be NP-hard [36]. The single machine case is solved optimally by the *shortest remaining processing time* (SRPT) algorithm [42]. Sitters [43] very recently showed that SRPT achieves an approximation guarantee of 1.25 for the parallel machines case.

Only little attention has been given to studying these problem in the cost sharing context described above, despite the fact that it is natural to assume that jobs (corresponding to players) compete for machine resources and need to share the cost of the resulting schedule. The main focus of this paper is to derive cost sharing mechanisms with good budget balance and social cost approximation guarantees for fundamental scheduling problems with completion (or flow) time objectives.

1.3 Moulin Mechanisms

In recent years, considerable progress has been made in devising truthful mechanisms for cooperative cost sharing games. Most notably, Moulin [33] proposed a class of cost sharing mechanisms (widely known as *Moulin mechanisms*) that realize the strong notion of *group-strategyproofness*, which ensures that no coordinated bidding of a coalition of players can ever strictly increase the utility of some player without strictly decreasing the utility of another player in the coalition. Basically, a Moulin mechanism can be viewed as an iterative ascending auction: In each iteration, the mechanism asks every player to pay a certain cost share. If every player is willing to pay his cost share, the mechanism outputs the respective player set together with their cost shares and halts. Otherwise, it removes all players that are not willing to pay their cost shares from the game and continues with the next iteration. Moulin [33] showed that this mechanism is group-strategyproof if the proposed cost shares are *cross-monotonic*, i.e., the cost share of a player does not decrease when some of the other players are removed from the game. Moreover, he proved that for submodular cost functions, Moulin mechanisms are the only cost sharing mechanisms that are budget balanced and group-strategyproof. This result has very recently been complemented by Juarez [27] who shows that every group-strategyproof cost sharing mechanism is a Moulin mechanism and vice versa. A subtle point in this context is whether a player that is *indifferent*, i.e., whose valuation is equal to the requested payment, prefers to accept or reject the service (see [27] for a detailed discussion). Throughout this paper, we

assume that such players always prefer to accept the service.

1.4 Limitations of Moulin Mechanisms

Based on Moulin's original result [33], a lot of research in recent years has gone into designing cross-monotonic cost sharing methods for the cost sharing variants of many classical optimization problems. The cost functions induced by these optimization problems are often neither submodular nor supermodular. However, most optimization problems that have been considered in the literature bear the common characteristic that larger sets of served players have a larger potential for individual cost savings because the underlying cost functions are subadditive. Well-studied examples that fall into this class are fixed tree multicast [2, 16, 17], minimum spanning tree [26, 29], Steiner tree [26], price-collecting Steiner tree [23], facility location [35], connected facility location [24, 31, 35], Steiner forest [30], and vertex and set cover [25]. In general, scheduling problems do not have this characteristic because the objective functions are usually superadditive and thus adding additional players corresponds to a larger potential of individual cost increases. To the best of our knowledge, there are only a few articles that study scheduling problems in the cost sharing context introduced above [5, 6, 7].

Recent negative results showed that for several fundamental cost sharing games, Moulin mechanisms inevitably suffer from poor approximation factors with respect to budget balance [5, 7, 25, 30, 40] or social cost [7, 11, 40, 41]. For example, in [7] it is shown that no cross-monotonic cost sharing method can achieve a budget balance approximation factor of less than $(n+1)/2$ for the minimum weighted completion time scheduling game $P||\sum_i w_i C_i$ mentioned above, even in the single-machine case and if all jobs have unit processing times and weights. This is in stark contrast to the fact that an optimal schedule can be computed in polynomial time in this case by Smith's rule.

1.5 Acyclic Mechanisms

Driven by shortcomings inherent to Moulin mechanisms, Mehta, Roughgarden, and Sundararajan [32] recently introduced a general framework to derive so-called *acyclic mechanisms*. These mechanisms implement a slightly weaker notion of group-strategyproofness, called *weak group-strategyproofness* [13, 32], which ensures that no coordinated bidding of a coalition of players can ever strictly increase the utility of *every* player in the coalition. The authors show that primal-dual approximation algorithms for several combinatorial optimization problems naturally give rise to acyclic mechanisms that achieve good approximation guarantees both with respect to budget balance and social cost.

Alike Moulin mechanisms, acyclic mechanisms are driven by cost sharing methods that have to fulfill certain properties. However, these properties are less restrictive than the cross-monotonicity requirement for Moulin mechanisms and therefore leave more flexibility to define appropriate cost shares. Acyclic mechanisms therefore open new ground for improving the approximation guarantees with respect to budget balance and social cost. Nevertheless, finding such cost sharing methods is often still a highly non-trivial and problem-specific task.

Problem	Generalized incremental mechanisms	Lower bounds for Moulin mechanisms
$P\|\|\sum_i C_i$	(1, 2)	$\frac{n+1}{2}$ [7]
$P\|\|\sum_i w_i C_i$	(1.21, 2.42)	$\frac{n+1}{2}$ [7]
$1\|r_i, pmtn\|\sum_i C_i$	(1, 4)	$\frac{n+1}{2}$ [7]
$P\|r_i, pmtn\|\sum_i C_i$	(1.25, 5)	$\frac{n+1}{2}$ [7]
$1\|r_i, pmtn\|\sum_i F_i$	1	$\frac{n+1}{2}$ [7]

Table 1: Summary of the approximation guarantees obtained by generalized incremental mechanisms in this paper. (β, α) denotes the budget balance (β) and social cost (α) approximation factors; all other entries refer to budget balance factors. The respective lower bounds on the budget balance factors of Moulin mechanisms are given in the right column.

1.6 Our Contributions

In this paper, we present a general approach to derive weakly group-strategyproof mechanisms from approximation algorithms. We show how a ρ-approximation algorithm for the underlying optimization problem of a cost sharing game can be turned into a cost sharing mechanism that is ρ-budget balanced, and prove that this mechanism is weakly group-strategyproof. Our construction uses the approximation algorithm as a black-box. The basic idea is very simple: According to the order on the set of players that are remaining in the game, we charge every player his incremental cost with respect to the approximate cost function induced by the approximation algorithm. The key property of our approach is that the order on the remaining players may change during the course of the mechanism.

A difficulty that arises is that in general the resulting mechanism per se does not fulfill the *no positive transfer property* (which requires that all cost shares are non-negative). While there are different ways to overcome this problem, we identify a *consistency* property on the order in which players are considered and argue that the mechanism guarantees no positive transfers whenever its approximate cost is non-decreasing with respect to this consistent order. We provide a series of examples that illustrate that several approximation algorithms naturally induce a consistent order on the players such that its cost is non-decreasing.

Exploiting the consistency property, we also provide some general means that facilitate proving social cost approximation factors of our mechanisms. Essentially, we determine a *weak monotonicity* property that, if satisfied by the incremental approximate cost shares defined by the mechanism, enables us to simplify bounding its social cost approximation guarantee.

We demonstrate the applicability of our techniques by developing weakly group-strategyproof mechanisms for completion time (and flow time) scheduling problems with and without release dates and preemption. Our techniques turn out to be particularly effective in this scheduling context. The results are summarized in Table 1. Our mechanisms outperform the strong lower bounds of $(n+1)/2$ on the budget balance factor of Moulin mechanisms for all completion time related objectives [7]. We emphasize that these are the first cost sharing mechanisms that are weakly group-strategyproof and achieve constant approximation guarantees with respect to *both* budget balance *and* social cost. Moreover, for most of these problems our mechanisms achieve budget

balance factors that match the current best approximation guarantees known for these problems.

1.7 Consequences and Further Implications

While previously most cost sharing mechanisms were developed in case-by-case studies, this is the first characterization of sufficient conditions that allow to derive mechanisms from existing approximation algorithms, thereby exploiting their full strength. Our results can therefore be seen as a first step towards quantifying the trade-off between the best possible approximation guarantees achievable by approximation algorithms and weakly-groupstrategyproof mechanisms.

Using our approach, deriving a weakly group-strategyproof mechanism essentially boils down to identifying a consistent order on the players such that the cost of the (approximation) algorithm is non-decreasing, which is an easy task in many cases. Moreover, the resulting mechanism inherits the approximation guarantee as budget balance factor. Hence, all that is left is to bound its social cost approximation guarantee, for which we provide some general techniques. Our approach thus simplifies the process of developing weakly group-strategyproof cost sharing mechanisms.

Besides the game-theoretical insights that we gain here, our results also have some further implications in the scheduling context: Every mechanism that approximates the social cost objective is at the same time an approximation algorithm for the respective *scheduling problem with rejection* [4] (formal definitions are given in Section 2). As a by-product, we therefore obtain constant factor approximation algorithms for several machine scheduling problems with rejection. These results might be of independent interest.

1.8 Relation to Incremental and Acyclic Mechanisms

Another class of cost sharing mechanisms that have been introduced by Moulin [33] are *incremental mechanisms*. An incremental mechanism considers players sequentially according to an arbitrary but fixed order. The cost share offered to a player is equal to his *incremental cost*, i.e., the increase in cost caused by adding this player to the set of previously selected players. The player is added to the set of selected players if he accepts his cost share. Moulin claimed that for supermodular cost functions, incremental mechanisms are basically the only cost sharing mechanisms that are group-strategyproof and budget balanced.[1] Juarez [27] very recently showed that group-strategyproof cost sharing mechanisms correspond to so-called *sequential mechanisms* if indifferent players prefer to reject the service. Sequential mechanisms define a slightly more general class of cost sharing mechanisms than incremental mechanisms (see [27] for precise definitions). Moulin also extended the class of incremental mechanisms by allowing that the order on the set of players that have not been considered so far may change during the course of the mechanism. These mechanisms are called *generalized incremental mechanisms* in [33].

Incremental and generalized incremental mechanisms are much less prevailing in literature than Moulin mechanisms, especially in the context of deriving cost sharing mechanisms for optimization problems. Interestingly, the mechanisms proposed in this paper correspond to generalized incremental mechanisms as introduced by Moulin [33].

[1] However, the original characterization given in [33] is flawed (as indicated in [27]) and holds only under the assumption that players are never indifferent.

Our work therefore reveals that these mechanisms naturally arise as weakly group-strategyproof mechanisms with good budget balance and social cost approximation guarantees for several fundamental cost sharing games for which Moulin mechanisms inevitably fail.

Our generalized incremental mechanisms belong to the class of acyclic mechanisms. Indeed, we first encountered these mechanisms when studying the framework of acyclic mechanisms (see also the exposition in [8]). We explain how in this framework generalized incremental mechanisms can be viewed as being complementary to Moulin mechanisms regarding the degree of freedom that the mechanism has for ordering its cost share proposals to players.

1.9 Further Related Work

Dobzinski et al. [14] recently showed that for cost functions comprising the *public excludable good problem*, a logarithmic gap between the budget balance and social cost approximation factors is inevitable, even if only strategyproofness is required. Unfortunately, the situation is even worse for generalized incremental mechanisms. As we will show, our generalized incremental mechanisms cannot achieve a social cost approximation guarantee of less than n for the public excludable good problem. The public excludable good example heavily exploits the fact that its cost function is subadditive. With respect to social cost approximation, our mechanisms perform poorly for these types of cost functions. However, most objective functions induced by scheduling problems are superadditive and our mechanisms turn out to be particularly effective for these kinds of cost sharing games.

Independently of our work, Bleischwitz et al. [6] recently defined *egalitarian mechanisms*, which belong to the class of acyclic mechanisms. The authors show how to construct egalitarian mechanisms from approximation algorithms that fulfill a certain monotonicity property, requiring that the approximate solution cost does not increase when any player's *size* (e.g., its processing time) is reduced. They apply their results to makespan scheduling and bin packing problems. The authors also prove that acyclic mechanisms are *weakly group-strategyproof against collectors*, a notion that strengthens weak group-strategyproofness to the setting where indifferent players are assumed to strictly prefer receiving service at their valuation price over not receiving service.

Pountourakis and Vidali [37] very recently gave a complete characterization of group-strategyproof cost sharing mechanisms and thereby settled a major open problem. They show that group-strategyproof cost sharing mechanisms are completely characterized by a certain *fence monotonicity* property in combination with a stable allocation rule and a valid tie-breaking rule (see [37] for definitions). It remains open whether this characterization can be used algorithmically.

In scheduling problems *with rejection*, the algorithm may choose to schedule only a subset of the jobs and pay a specified penalty for each job that is omitted. This setting has been introduced by Bartal et al. [4] for an online minimum makespan scheduling problem. Engels et al. [15] study the offline version for completion time related problems. They give randomized algorithms for minimizing the weighted sum of completion times on related machines which achieve expected approximation guarantees of 2 with and 3/2 without release dates, respectively. For the single machine case, they were able to design optimal algorithms with pseudopolynomial running-time (unless either weights or processing times are all equal). Bunde [10] gives an optimal algorithm for the single machine case with release dates and unit processing times.

He also proves that the completion time scheduling problem with rejection is NP-hard even on a single machine if there are release dates. Bansal et al. [3] study the online preemptive single machine case where flow time or job idle time is concerned.

2 Preliminaries

2.1 Cost Sharing

A *binary demand cost sharing game* is defined as follows. We are given a universe U of players that are interested in a common service, and a cost function $C : 2^U \to \mathbb{R}^+$ that specifies the cost $C(S)$ to serve player set $S \subseteq U$. We require that $C(\emptyset) = 0$. In this paper, we assume that C is given implicitly by the cost of an optimal solution to an underlying cost-minimization problem $\mathcal{P}$. Every player $i \in U$ has a private, non-negative *valuation* v_i and a non-negative *bid* b_i for receiving the service.

A *(direct revelation) cost sharing mechanism* M takes the bid vector $b := (b_i)_{i \in U}$ as input and computes a binary allocation vector $x := (x_i)_{i \in U}$ and a payment vector $p := (p_i)_{i \in U}$. Let S^M be the subset of players associated with the allocation vector x, i.e., $i \in S^M$ iff $x_i = 1$. We say that S^M is the player set that receives service. We require that a cost sharing mechanism complies with the following three standard assumptions:

1. *Individual rationality*: A player is charged only if he receives service and his payment is at most his bid, i.e., $p_i = 0$ if $i \notin S^M$ and $p_i \leq b_i$ if $i \in S^M$.
2. *No positive transfer*: A player is not paid for receiving the service, i.e., $p_i \geq 0$ for all $i \in S^M$.
3. *Consumer sovereignty*: A player is guaranteed to receive service if he is willing to bid high enough, i.e., there exists a threshold value b_i^* for every player $i \in U$ such that $i \in S^M$ for all $b_i \geq b_i^*$.

In addition, the mechanism has to compute a (possibly suboptimal) feasible solution to the underlying optimization problem $\mathcal{P}$ on the player set S^M. We denote the cost of the computed solution by $\bar{C}(S^M)$. A mechanism M is *β-budget balanced* for some $\beta \geq 1$ if

$$\bar{C}(S^M) \leq \sum_{i \in S^M} p_i \leq \beta \cdot C(S^M).$$

We assume that players act strategically and every player's goal is to maximize his own utility. The *utility* u_i of player i is defined as $u_i(x, p) := v_i x_i - p_i$. Since the outcome (x, p) computed by the mechanism M solely depends on the bids b of the players (and not on their true valuations), a player may have an incentive to declare a bid b_i that differs from his valuation v_i. We say that M is *strategyproof* if bidding truthfully is a dominant strategy for every player. That is, for every player $i \in U$ and every two bid vectors b, b' with $b_i = v_i$ and $b_j = b'_j$ for all $j \neq i$, we have $u_i(x, p) \geq u_i(x', p')$, where (x, p) and (x', p') are the solutions output by M for bid vectors b and b', respectively.

We assume that players can form coalitions in order to coordinate their bids. A mechanism M is *group-strategyproof* if no coordinated bidding of a coalition $T \subseteq U$ can ever strictly increase the utility of some player in T without strictly decreasing the utility of another player in T. More formally, for every coalition $T \subseteq U$ and every

two bid vectors b, b' with $b_i = v_i$ for every $i \in T$ and $b_i = b'_i$ for every $i \notin T$,

$$\exists i \in T : \; u_i(x', p') > u_i(x, p) \qquad \Longrightarrow \qquad \exists j \in T : \; u_j(x', p') < u_j(x, p).$$

M is *weakly group-strategyproof* if no coordinated bidding can ever strictly increase the utility of *every* player in the coalition. That is, for every coalition $T \subseteq U$ and every two bid vectors b, b' with $b_i = v_i$ for every $i \in T$ and $b_i = b'_i$ for every $i \notin T$,

$$\exists i \in T : \; u_i(x', p') \leq u_i(x, p).$$

Intuitively, by requiring weak group-strategyproofness only, we assume that players adopt a slightly more conservative attitude with respect to their willingness of joining a coalition: While in the group-strategyproof setting a player only defects from a coalition if he is strictly worse off, he defects already if he is not strictly better off in the weakly group-strategyproof setting.

The standard notion that is used to assess the efficiency of a mechanism is its *social welfare*, which is defined as the sum of valuations of all served players minus the servicing cost. However, classical results in economics [21, 39] rule out the existence of strategyproof mechanisms that achieve budget balance and social welfare at the same time. Moreover, Feigenbaum et al. [16] showed that for the multicast cost sharing game these two objectives cannot even be approximated simultaneously, even if only strategyproofness is required.

Roughgarden and Sundararajan [40] proposed an alternative efficiency measure. The *social cost* [40] of a set $S \subseteq U$ is defined as

$$\Pi(S) := \bar{C}(S) + \sum_{i \notin S} v_i.$$

A mechanism M is said to be α-*approximate* for some $\alpha \geq 1$ if, assuming that every player $i \in U$ bids truthfully $b_i = v_i$, the social cost of the served set S^M output by the mechanism satisfies $\Pi(S^M) \leq \alpha \cdot \Pi^*$, where

$$\Pi^* := \min_{S \subseteq U} \left(C(S) + \sum_{i \notin S} v_i \right)$$

denotes the optimal social cost.

It is not hard to verify that the social welfare and social cost efficiency objectives coincide if they can be computed exactly; but they may differ with respect to their approximability. Using this alternative efficiency measure, it became possible to derive mechanisms with non-trivial approximation guarantees with respect to budget balance and social cost. Roughgarden and Sundararajan also revealed a relation between the social cost approximation factor of a Moulin mechanism and a *summability property* of the underlying cost sharing method (see [40] for more details).

2.2 Scheduling

2.2.1 Parallel Machine Scheduling

In a parallel machine scheduling problem, we are given a set U of n jobs that are to be scheduled on m identical machines. Every job $i \in U$ has a non-negative *release date* r_i, a positive *processing time* p_i, and a non-negative weight w_i. The release date specifies the time when job i becomes available for execution. The processing time describes the time needed to execute i on one of the machines. Every machine can execute at most one job at a time. In the *preemptive* setting, the execution of a job can be interrupted at any point of time and resumed later; in contrast, in the *non-preemptive* setting, job interruption is not permitted. In the cost sharing variant of a scheduling problem, each job is identified with a player who wants his job to be processed on one of the machines.

Depending on the respective scheduling applications, there are various meaningful objective functions for machine scheduling problems. Let $C_i(S)$ denote the *completion time* of job $i \in S$ in the schedule for the set $S \subseteq U$ computed by a given scheduling algorithm. Among the most common objectives are the minimization of the total weighted completion time, i.e., $\sum_i w_i C_i$, and the makespan, i.e., $\max_i C_i$, over all feasible schedules. The *flow time* F_i of a job is defined as the difference between its completion time and its release date, i.e., $F_i := C_i - r_i$. We will often use the three-field notation scheme by Graham et al. [19] to refer to specific scheduling settings.

2.2.2 Scheduling with Rejection

In *scheduling problems with rejection*, the algorithm may choose to schedule only a subset of the jobs and pay a specified penalty for each job that is omitted. This setting has been introduced by Bartal et al. [4]. Consider an arbitrary scheduling problem $\mathcal{P}$ with job set U and objective function C. A natural variant of this problem is the following: Every job $i \in U$ has a non-negative *penalty* z_i. For every job $i \in U$, we now have the option to either schedule i and incur its respective contribution to the objective function value, or not to schedule i and pay its penalty z_i. More formally, the problem is to compute a subset $S \subseteq U$ of jobs such that the overall cost $C(S) + \sum_{i \notin S} z_i$ is minimized.

3 Generalized Incremental Mechanisms

We describe our approach for converting approximation algorithms into weakly group-strategyproof cost sharing mechanisms. Our mechanisms belong to the class of *generalized incremental mechanisms* as introduced by Moulin in [33], and we therefore borrow this naming here.

3.1 Construction and Basic Properties

Suppose we are given a ρ-approximation algorithm ALG for the underlying optimization problem $\mathcal{P}$. Let $\bar{C}$ denote the cost function induced by ALG, i.e., $\bar{C}(S)$ is the cost of the solution computed by ALG for player set $S \subseteq U$. Without loss of generality, we assume that $\bar{C}(\emptyset) = 0$. Besides the approximation algorithm ALG, the main ingredient for our framework is an injective *order function* $\tau : U \times 2^U \to \mathbb{R}^+$ which defines a

Algorithm 1: Generalized incremental mechanism $M(\text{ALG}, \tau)$ induced by ALG and τ.

Input: Set of players U and bid vector $b = (b_i)_{i \in U}$
Output: Allocation vector $x = (x_i)_{i \in U}$ and payment vector $p = (p_i)_{i \in U}$

Initialize $A := \emptyset$, $R := U$.
while $A \neq R$ **do**
  Among all players $i \in R \setminus A$, let i^* be the one with minimum $\tau(i, R)$.
  Define $p_{i^*} := \bar{C}(A \cup \{i^*\}) - \bar{C}(A)$.
  if $b_{i^*} \geq p_{i^*}$ **then** set $A := A \cup \{i^*\}$;
  else set $R := R \setminus \{i^*\}$.
end

Output the characteristic vector x of A and payments p.

permutation for every subset $S \subseteq U$ by ordering the elements in S with respect to increasing τ-values.

The *generalized incremental mechanism* $M(\text{ALG}, \tau)$ induced by ALG and τ receives the bid vector b as input and proceeds as indicated in Algorithm 1. Throughout its execution, R refers to the set of players that currently remain in the game, and A denotes the set of players that have been accepted so far. The mechanism starts with the entire player set $R = U$ and initializes $A = \emptyset$. In every iteration, it picks the player i^* from $R \setminus A$ with the smallest $\tau(\cdot, R)$-value, and computes its incremental approximate cost share p_{i^*}, defined as the increase in the approximate cost $\bar{C}$ when player i^* is added to A. If player i^* accepts this cost share, he is added to the set A of accepted players; otherwise, he is removed from R and hence rejected from the game. The mechanism continues like this until eventually all remaining players have been accepted. It outputs the characteristic vector x of the accepted players A and the corresponding payments p (where we implicitly set $p_i = 0$ for all $i \notin A$).

We remark that the cost shares assigned to the served players depend on the cost function $\bar{C}$ induced by the approximation algorithm ALG and are not necessarily non-negative. Thus, our generalized incremental mechanism does not necessarily satisfy the *no positive transfer* property. We address this issue in Section 3.2 below.

It is straightforward to see that the generalized incremental mechanism inherits its budget balance factor from the input approximation algorithm:

Lemma 1. *The generalized incremental mechanism $M(\text{ALG}, \tau)$ is ρ-budget balanced.*

Proof. In every iteration of the mechanism, we have $\sum_{i \in A} p_i = \bar{C}(A)$, since every accepted player pays exactly the incremental approximate cost for adding him to the current set A. In particular, this is true for the output set S^M. Since ALG is a ρ-approximation algorithm, we obtain

$$\bar{C}(S^M) = \sum_{i \in S^M} p_i \leq \rho \cdot C(S^M),$$

which proves ρ-budget balance. □

We next prove that the generalized incremental mechanism is weakly-groupstrategyproof.

Lemma 2. *The generalized incremental mechanism $M(\text{ALG}, \tau)$ is weakly group-strategyproof.*

Proof. Fix a coalition $T \subseteq U$ and a bid vector b with $b_i = v_i$ for all $i \in T$. Assume for contradiction that all members of the coalition can increase their utilities by changing their bids to b' (while $b_i = b'_i$ for all $i \notin T$). The runs of the generalized incremental mechanism on b and b' are identical until the first member of T, say j, is offered a cost share. Since the cost share offered to him depends only on the set A of previously accepted players, which coincides in both runs, the utility of j is maximized when bidding v_j and cannot be influenced by other members of T. Hence j cannot increase his utility by joining the coalition. □

The following theorem follows from Lemmas 1 and 2.

Theorem 1. *Let τ be an arbitrary order function and let* ALG *be a ρ-approximate algorithm for an optimization problem $\mathcal{P}$. The generalized incremental mechanism $M(\text{ALG}, \tau)$ is a weakly group-strategyproof and ρ-budget balanced cost sharing mechanism for $\mathcal{P}$, which does not necessarily satisfy the no positive transfer property.*

The following example shows that generalized incremental mechanisms are not group-strategyproof in general.

Example 1. *We define an instance of a cost sharing game on $n = 2$ players with valuations $v_1 = 1$ and $v_2 = 2$. Let $\bar{C}(\{1\}) = \bar{C}(\{2\}) = 1$ and $\bar{C}(\{1, 2\}) = 3$. This cost function is, e.g., realized by an optimal algorithm for the completion time scheduling problem on one machine with two jobs of unit processing times. Let τ be the offer function which orders players by their index. The induced generalized incremental mechanism accepts both players and yields utilities $u_1(v) = u_2(v) = 0$. Consider the forming of a coalition with bids $b_1 = 0$ and $b_2 = 2$. In this case, player 1 rejects, and so $u_1(b) = 0$ as before, but $u_2(b) = 2 - 1 = 1$. Hence, this coalition breaks the property of group-strategyproofness.*

3.2 No Positive Transfer

As mentioned above, our generalized incremental mechanism does not necessarily guarantee the *no positive transfer* property. Clearly, this property is satisfied if the *approximate* cost function $\bar{C}$ induced by the approximation algorithm ALG is non-decreasing, i.e., $\bar{C}(S) \leq \bar{C}(T)$ for all $S \subseteq T \subseteq U$. However, this requirement is too restrictive in general as many approximation algorithms do not satisfy it.

The following standard adaptation can be used to realize the no positive transfer property if the *optimal* cost function C is non-decreasing, i.e., $C(S) \leq C(T)$ for all $S \subseteq T \subseteq U$: Simply charge a player zero if the incremental approximate cost of adding this player is negative. That is, we redefine the incremental approximate cost share in Line 4 of Algorithm 1 as

$$p_{i^*} := \max \left\{ 0,\ \bar{C}(A \cup \{i^*\}) - \sum_{i \in A} p_i \right\}.$$

Lemma 3. *The adapted generalized incremental mechanism $M(\text{ALG}, \tau)$ is ρ-budget balanced if the optimal cost function C is non-decreasing.*

Proof. The lemma is shown by induction on the cardinality of the set A of players that are accepted. Clearly, the claim holds for $|A| = 0$. Whenever a player i^* is added to the current set A, we have

$$\sum_{i \in A \cup \{i^*\}} p_i = p_{i^*} + \sum_{i \in A} p_i \geq \bar{C}(A \cup \{i^*\}).$$

If $p_{i^*} > 0$ the above inequality is an equality, and it holds that $\bar{C}(A \cup \{i^*\}) \leq \rho \cdot C(A \cup \{i^*\})$ since ALG is a ρ-approximation algorithm. Otherwise, $p_{i^*} = 0$ and we have $\sum_{i \in A} p_i \leq \rho \cdot C(A)$ by the induction hypothesis and $C(A) \leq C(A \cup \{i^*\})$ since C is non-decreasing. □

We propose a different approach here for the following two reasons:

1. The above adaptation works only if the optimal cost function is non-decreasing. This requirement might be too restrictive as the cost functions of several natural optimization problems do not fulfill this property. A simple example is the *minimum spanning tree game* given below.

2. The approach we propose here will enable us to derive a general technique to prove social cost approximation guarantees. We use this technique to obtain generalized incremental mechanisms that achieve constant approximation approximation factors with respect to both budget balance and social cost for completion time scheduling problems (see Section 5).

Throughout this section, we consider the *minimum spanning tree game* as an illustrative example. In the *minimum spanning tree game*, we are given an undirected graph $G = (V \cup \{r\}, E)$ with edge weights $w_e \geq 0$ for all edges $e \in E$ and a designated root vertex r. We assume that there is an edge between r and every vertex in V. Every player $i \in U$ is associated with a unique vertex in V and the goal of player i is to connect his node to the root r. A *minimum spanning tree* for a given subset $S \subseteq U$ of the players is a tree $\mathcal{T}$ in the subgraph induced by $S \cup \{r\}$ that spans all vertices in $S \cup \{r\}$ and minimizes the total weight $w(\mathcal{T}) := \sum_{e \in \mathcal{T}} w_e$. The cost $C(S)$ to connect a player set $S \subseteq U$ is defined as the weight of a minimum spanning tree for S.

The following example shows that the cost function defined by the minimum spanning tree game is not non-decreasing in general.

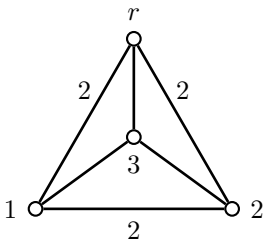

Figure 1:

Example 2. *Consider the instance of the minimum spanning tree game with player set $U = \{1, 2, 3\}$ depicted in Figure 1. All edges incident to vertex 3 have weight $(1+\varepsilon)$ with $0 < \varepsilon < \frac{1}{3}$, and all other edges have weight 2 as indicated. We have $C(\{1\}) = C(\{2\}) = 2$, $C(\{3\}) = 1 + \varepsilon$, $C(\{1, 2\}) = 4$, $C(\{1, 3\}) = C(\{2, 3\}) = 2 + 2\varepsilon$ and $C(U) = 3(1 + \varepsilon)$. Note that C is not non-decreasing because $C(\{1, 2\}) = 4 > 3(1 + \varepsilon) = C(\{1, 2, 3\})$ by our choice of ε.*

Recall that Prim's algorithm (PRIM) [38] solves the minimum spanning tree problem optimally. It starts with the root vertex r as the initial connected component and then iteratively picks a minimum weight edge that connects a new vertex to the current component until all vertices are connected. (If there are several minimum weight edges that might be chosen, we assume that an arbitrary but consistent tie breaking rule is used.)

Suppose we want to use Prim's algorithm to derive a generalized incremental mechanism $M(\text{PRIM}, \tau)$ for the minimum spanning tree game that is weakly group-strategyproof and budget balanced. How should we define the order function τ? As the above example indicates, if τ orders players by increasing index then the resulting mechanism does not satisfy the no positive transfer property because the cost share of player 3 is $3(1+\varepsilon) - 4 < 0$ (assuming that players 1 and 2 have been accepted). Clearly, the mechanism satisfies the no positive transfer property if τ orders players by decreasing index. However, a fixed global ordering of the players does not work in general as the following example shows.

Example 3. *Consider the instance of the minimum spanning tree game depicted in Figure 2. All edges incident to vertex* 1 *have weight* 1*, all edges incident to vertex* 2 *have weight* $(1+\varepsilon)$ *for some* $\varepsilon > 0$ *and all other edges (not depicted) have weight equal to the shortest path distance between their endpoints. Let* L *refer to the vertices in the bottom layer.*

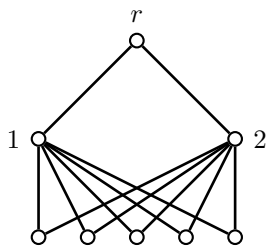

Figure 2:

The mechanism starts with the entire player set $R = U$*. A natural order* $\tau(\cdot, U)$ *of the players in* U *that satisfies the no positive transfer property is to first consider player* 1*, then the players in* L *and finally player* 2*. Note that if player* 1 *is rejected then the order* $\tau(\cdot, R)$ *of the remaining players* $R = U \setminus \{1\}$ *has to change such that first player* 2 *is considered and then the players in* L*. To see this, suppose that* $\tau(\cdot, R)$ *orders the players such that first the players in* $S \subseteq L$*,* $S \neq \emptyset$*, are considered, then player* 2 *and finally all remaining players of* R*. The cost share of player* 2 *is then* $(1+\varepsilon)(|S|+1) - 2|S|$*, which is negative for a sufficiently small* ε*.*

The above example illustrates some finer points of our approach. Essentially, we will show that several approximation algorithms exhibit a natural order function that ensures that the incremental approximate cost shares are non-negative. Hence, a much weaker restriction than requiring monotonicity of the approximate cost function $\bar{C}$ or the optimal cost function C suffices if we define the order function τ appropriately. As we will see, for some applications (see, e.g., Section 5.2) it will be crucial to allow that the order of the remaining players may change as the generalized incremental mechanism progresses.

3.3 Consistency

Consider a set $S \subseteq U$ and order the players in S according to increasing $\tau(\cdot, S)$ values, i.e.,

$$S = \{i_1, \ldots, i_p\} \quad \text{such that} \quad \tau(i_k, S) < \tau(i_l, S) \quad \forall 1 \leq k < l \leq p.$$

We also say S *is ordered by* τ. We denote by $S_k := \{i_1, \ldots, i_k\} \subseteq S$ the set of the first $1 \leq k \leq p$ elements in S ordered by τ and define $S_0 := \emptyset$.

Definition 1. An order function $\tau : U \times 2^U \to \mathbb{R}^+$ is *consistent* if for all subsets $S \subseteq T \subseteq U$, ordered by τ as $S = \{i_1, i_2, \ldots, i_p\}$ and $T = \{j_1, j_2, \ldots, j_q\}$, the following holds: If k is minimal with $j_k \in T \setminus S$, then $i_l = j_l$ for all $l < k$.

Figure 3 illustrates the restriction imposed by the consistency property.

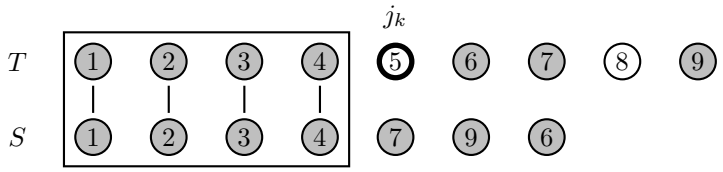

Figure 3: Illustration of the consistency property for two sets $S \subseteq T := \{1, \ldots, 9\}$; elements belonging to S are depicted in gray. Both sets are ordered according to τ. Note that consistency requires that S has to be ordered like T up to the first element j_k of T that is missing in S (indicated in bold).

Reconsider the minimum spanning tree game introduced above. We claim that Prim's algorithm gives rise to a consistent order function. For a subset $S \subseteq U$ of vertices, let $\tau(\cdot, S)$ be the order in which PRIM adds the vertices to the connected component in the run on S. It is not hard to see that τ is a consistent order function: Let $S \subseteq T \subseteq U$ be two subsets ordered by τ as $S = \{i_1, i_2, \ldots, i_p\}$ and $T = \{j_1, j_2, \ldots, j_q\}$. Let k be minimal with $j_k \in T \setminus S$. We need to argue that the order in which PRIM picks the first $k-1$ vertices in the run on S is the same as in the run on T, i.e., $i_l = j_l$ for all $l < k$. The proof is by induction on the first $1 \leq l < k$ vertices added by PRIM in the run on T. The claim clearly holds for $l = 1$, since PRIM starts with the same vertex in both runs. Suppose the claim is true for the first $l-1$ vertices and consider the iteration in the run on T in which vertex j_l, $1 < l < k$, is added. Then j_l is a vertex that is closest to the current connected component in the run on T. By the choice of k, j_l is also contained in S. Moreover, by the induction hypothesis, the $l-1$ vertices that have previously been chosen in the run on T and in the run on S are the same. Note that j_l must also be closest to the current connected component in the run on S because $S \subseteq T$. Thus $i_l = j_l$ (assuming a consistent tie-breaking rule). However, note that the order in which the vertices in $S \setminus \{j_1, \ldots, j_{k-1}\}$ are considered might differ from the respective order in T due to the absence of j_k. Our consistency property reflects exactly this.

We next show that a consistent order function τ in combination with a certain monotonicity property of the algorithm ALG guarantees that the resulting generalized incremental algorithm $M(\text{ALG}, \tau)$ (as defined in Algorithm 1) satisfies the no positive transfer property. Consider the execution of the generalized incremental mechanism $M(\text{ALG}, \tau)$ induced by ALG and a consistent order function τ. Recall that R refers to the set of players that are currently remaining in the game. Note that the order in which the players in $R = \{i_1, \ldots, i_{|R|}\}$ are considered remains the same until the first player, say i_k, is dropped. The consistency of τ now ensures that the ordered sets $R' = R \setminus \{i_k\}$ and R agree on the first $k-1$ players. Said differently, only the order of the players succeeding i_k in R can change in R'. Hence, the first $k-1$ players correspond to the set A of currently accepted players. We prove this formally in the next lemma.

Lemma 4. *At the beginning of every iteration of $M(\text{ALG}, \tau)$, we have $R_{|A|} = A$.*

Proof. We prove the lemma by induction on the number of iterations. In the first iteration, $R_{|A|} = R_0 = \emptyset = A$. For the induction step, assume that $R_{|A|} = A$ at the beginning of some iteration. Let i^* be the player that is picked in this iteration, i.e., by the induction hypothesis, i^* is the $(|A|+1)$st player in the order on R. Let R' and A' denote the updated sets at the end of this iteration. There are two cases:

(i) If i^* accepts, then $R' = R$ and $A' = A \cup \{i^*\}$. Hence, we can conclude that $R'_{|A'|} = R_{|A|+1} = A \cup \{i^*\} = A'$. (ii) On the other hand, if i^* rejects, then $A' = A$ and $R' = R \setminus \{i^*\}$. Note that i^* is the first element in the order on R which is not in R', and so by consistency of τ, we have $R'_{|A'|} = R_{|A|} = A = A'$. □

We can use this lemma to prove that the order in which players are added to the set A during the course of the generalized incremental mechanism $M(\text{ALG}, \tau)$ coincides with the order induced by τ on the final output set S^M.

Lemma 5. *Let S^M be the set of players output by $M(\text{ALG}, \tau)$. During the course of $M(\text{ALG}, \tau)$, players are added to A by increasing $\tau(\cdot, S^M)$-values.*

Proof. Suppose S^M is ordered by τ as $S^M = \{i_1, \ldots, i_p\}$. Let i be the kth player that is added to A in the execution of $M(\text{ALG}, \tau)$. We need to show that $i = i_k$.

Consider the iteration in which player i is added to A. Let R and A be the sets of remaining and accepted players at the beginning of the iteration, respectively, and let R' and A' be the respective sets at the end of the iteration. We have $R' = R$, $A' = A \cup \{i\}$ and $|A'| = k$. By Lemma 4, $R_{k-1} = A$ and $R'_k = R_k = A'$.

Note that all players in $R_{k-1} = A$ are contained in S^M, so by consistency of τ, at least the first $k-1$ elements of S^M coincide with those of R, i.e., $R_{k-1} = S^M_{k-1}$. By the same argument, we have $R'_k = R_k = S^M_k$. We conclude that $S^M_k \setminus S^M_{k-1} = R_k \setminus R_{k-1}$ and thus $i = i_k$. □

Lemma 5 has an important consequence: If the cost function $\bar{C}$ of the approximation algorithm ALG is non-decreasing as players are added to S^M one by one (in the order of τ), then the final payments charged by the generalized incremental mechanism $M(\text{ALG}, \tau)$ to the players in S^M are non-negative. Since potentially every subset $S \subseteq U$ might be chosen as the final output set, we require that the approximation algorithm satisfies this property for every subset of players:

Definition 2. Let ALG be a ρ-approximate algorithm for the underlying optimization problem $\mathcal{P}$. We say that ALG is *τ-increasing* if for every $S \subseteq U$ and every $1 \leq k \leq |S|$, we have $\bar{C}(S_k) \geq \bar{C}(S_{k-1})$.

The following theorem now follows directly from Lemma 5 and Definition 2.

Theorem 2. *Let τ be a consistent order function and let ALG be a τ-increasing ρ-approximate algorithm for an optimization problem $\mathcal{P}$. The generalized incremental mechanism $M(\text{ALG}, \tau)$ is a weakly group-strategyproof and ρ-budget balanced cost sharing mechanism for $\mathcal{P}$, which satisfies the no positive transfer property.*

Consider the minimum spanning tree game. We argued above that Prim's algorithms induces a consistent order function. It is not hard to verify that PRIM is τ-increasing: Given an arbitrary subset $S = \{i_1, \ldots, i_p\}$ ordered by τ, PRIM adds the vertices $i_1, \ldots, i_p$ one by one to the current component. The cost $C(S_k) - C(S_{k-1})$ of adding a vertex i_k, $1 \leq k \leq p$, is equal to the weight of the edge that is added to connect i_k to the tree. Since edge weights are non-negative, it follows that PRIM is τ-increasing. Theorem 2 yields the following result.

Corollary 1. *The generalized incremental mechanism M^{PRIM} induced by Prim's algorithm and τ is weakly group-strategyproof and 1-budget balanced for the minimum spanning tree problem.*

In Section 4 we provide some additional examples of approximation algorithms that are τ-increasing with respect to natural order functions that are consistent.

3.4 Social Cost Approximation

In this section, we derive some general techniques that will be helpful in proving social cost approximation guarantees of our generalized incremental mechanisms. These techniques seem particularly applicable if the underlying approximate cost function is superadditive. The following example shows that generalized incremental mechanisms may have poor social cost approximation guarantees if the underlying cost function is subadditive.

In the *public excludable good problem*, the serving cost is $C(S) = 1$ for every non-empty subset of players $\emptyset \neq S \subseteq U$ and $C(\emptyset) = 0$. Dobzinski et al. [14] showed that for this problem no constant budget balanced and truthful mechanisms can approximate social cost by less than a logarithmic factor, even if only strategyproofness is required. The situation is worse for generalized incremental mechanisms.

Example 4. *Consider an instance of the public excludable good problem with n players. Set the valuation of each player $i \in U$ to $v_i = 1 - \epsilon$ for an arbitrarily small constant $\epsilon > 0$. Assuming truthful bidding, any generalized incremental mechanism serves the empty set incurring a social cost of $\Pi(\emptyset) = (1 - \epsilon)n$. On the other hand, serving the whole set induces a social cost of $\Pi(U) = 1$. We therefore obtain a lower bound on the social cost approximation factor of essentially n.*

We define an *incremental approximate cost share function* $\xi : U \times 2^U \to \mathbb{R}$ induced by ALG and τ as follows. Let $S = \{i_1, \ldots, i_p\} \subseteq U$ be an arbitrary subset of players ordered by τ. The incremental approximate cost share $\xi_i(S)$ of player $i \in S$ is defined as the increase in the approximate cost function $\bar{C}$ caused by player i when the players in S are added one by one according to the order τ. More formally, we define for every player $i = i_k$, $1 \leq k \leq p$, $\xi_i(S) := \bar{C}(S_k) - \bar{C}(S_{k-1})$; for every player $i \notin S$, let $\xi_i(S) := 0$. Note that Definition 2 is equivalent to stating that for every subset $S \subseteq U$ and every player $i \in S$, $\xi_i(S)$ is non-negative.

We next introduce a weak monotonicity property (which is reminiscent of the inverse of the core property).

Definition 3. Let ξ be the incremental approximate cost share function induced by an approximation algorithm ALG and an order function τ. We call ξ *weakly monotone* if for all subsets $S \subseteq T \subseteq U$, $\sum_{i \in S} \xi_i(T) \geq \bar{C}(S)$.

Intuitively, the weak monotonicity property in Definition 3 is easier to satisfy if the (approximate) cost function is superadditive.

Let S^* refer to a player-set that minimizes the social cost objective function, i.e.,

$$S^* := \arg\min_{S \subseteq U} \left(C(S) + \sum_{i \notin S} v_i \right).$$

We obtain the following theorem for generalized incremental mechanisms that implement weakly monotone cost share functions.

Theorem 3. *Let τ be a consistent order function and let ALG be a τ-increasing algorithm. Suppose that the incremental approximate cost share function ξ induced by ALG and τ is weakly monotone. Then, the generalized incremental mechanism $M(\text{ALG}, \tau)$ approximates social cost by a factor of α if*

$$\frac{\bar{C}(S^M \cup S^*)}{C(S^*) + C(S^M \setminus S^*)} \leq \alpha.$$

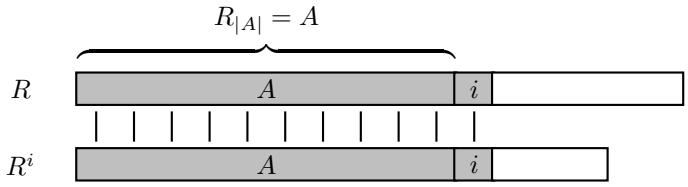

Figure 4: Illustration of the consistency property as used in the proof of Theorem 3.

Proof. We can bound the social cost approximation factor by

$$\frac{\Pi(S^M)}{\Pi^*} = \frac{\bar{C}(S^M) + \sum_{i \in S^* \setminus S^M} v_i + \sum_{i \notin S^M \cup S^*} v_i}{C(S^*) + \sum_{i \in S^M \setminus S^*} v_i + \sum_{i \notin S^M \cup S^*} v_i} \leq \frac{\bar{C}(S^M) + \sum_{i \in S^* \setminus S^M} v_i}{C(S^*) + \sum_{i \in S^M \setminus S^*} v_i}$$
$$\leq \frac{\bar{C}(S^M) + \sum_{i \in S^* \setminus S^M} v_i}{C(S^*) + \sum_{i \in S^M \setminus S^*} \xi_i(S^M)} \leq \frac{\bar{C}(S^M) + \sum_{i \in S^* \setminus S^M} v_i}{C(S^*) + C(S^M \setminus S^*)}.$$

Here, the first inequality follows from the fact that $\frac{a}{b} \leq \frac{a-c}{b-c}$ for arbitrary real numbers $a \geq b > c \geq 0$. The second inequality holds because $v_i \geq \xi_i(S^M)$ for every player $i \in S^M$, since i accepted and we assume truthful bidding. The last inequality follows from weak monotonicity of ξ and the fact that $\bar{C}(S) \geq C(S)$ for every set $S \subseteq U$.

We conclude the proof by showing that

$$\sum_{i \in S^* \setminus S^M} v_i \leq \bar{C}(S^M \cup S^*) - \bar{C}(S^M).$$

Without loss of generality, number the players in $S^* \setminus S^M$ in the order in which they were rejected by M, i.e., $S^* \setminus S^M = \{1, \ldots, \ell\}$. Fix a player $i \in S^* \setminus S^M$ and consider the iteration in which player i was removed. Let R and A be the sets of remaining and accepted players at the beginning of this iteration, respectively. Define R^i as the subset of players in $S^* \cup S^M$ that were still remaining in the game when i was picked, i.e., $R^i := S^M \cup \{i, i+1, \ldots, \ell\}$. Let $k := |A|$. By Lemma 4, we have $R_k = A$. Moreover, since i is chosen, we have $R_{k+1} = A \cup \{i\}$. Note that $A \cup \{i\}$ is a subset of R^i. By the consistency of τ, the first $k+1$ elements of R^i and R must coincide and we thus have $A \cup \{i\} = R_{k+1} = R^i_{k+1}$. The same argument also yields that $A = R_k = R^i_k$; see Figure 4 for an illustration. Therefore,

$$p_i = \bar{C}(A \cup \{i\}) - \bar{C}(A) = \bar{C}(R^i_{k+1}) - \bar{C}(R^i_k) = \xi_i(R^i).$$

Since i rejected, we have $v_i < p_i = \xi_i(R^i)$. Note that $R^i = R^{i+1} \cup \{i\}$. Exploiting that ξ is weakly monotone, we obtain that

$$\bar{C}(R^i) = \sum_{j \in R^i} \xi_j(R^i) = \xi_i(R^i) + \sum_{j \in R^{i+1}} \xi_j(R^i) \geq \xi_i(R^i) + \bar{C}(R^{i+1}).$$

Summing over all $i \in \{1, \ldots, \ell\}$ yields

$$\sum_{i \in S^* \setminus S^M} v_i < \sum_{i=1}^{\ell} \xi_i(R^i) \leq \sum_{i=1}^{\ell} \left(\bar{C}(R^i) - \bar{C}(R^{i+1})\right) = \bar{C}(S^M \cup S^*) - \bar{C}(S^M).$$

□

We use Theorem 3 in Section 5 to derive constant social cost approximation guarantees for our generalized incremental mechanisms for scheduling problems.

4 Applications

We demonstrate the applicability of our approach by deriving generalized incremental mechanisms for a series of fundamental cost sharing games. For some of the examples given here, better cost sharing mechanisms (with respect to social cost approximation) exist in the literature. However, the main purpose of this section is to show that our mechanisms can easily be derived from existing approximation algorithms.

4.1 Spanning Tree, Steiner Tree and Traveling Salesman

We already argued that Prim's algorithms gives rise to a generalized incremental mechanism M^{PRIM} that is budget balanced and weakly group-strategyproof for the minimum spanning tree problem (see Corollary 1). We can use Prim's algorithm to obtain 2-budget balanced cost sharing methods for the Steiner tree problem and the traveling salesman problem. The *Steiner tree problem* asks for a minimum weight tree that spans a subset of prespecified terminal vertices, possibly including some non-terminal vertices. In the *traveling salesman problem*, the goal is to determine a minimum weight tour through all vertices such that every vertex is visited exactly once. Both problems admit a simple approximation algorithm that constructs a 2-approximate solution from a minimum spanning tree (see, e.g., [45]). Using standard arguments, we can therefore use the cost sharing mechanism M^{PRIM} to obtain 2-budget balanced generalized incremental mechanisms for these problems.

Corollary 2. *Prim's algorithm yields a 2-budget balanced and weakly group-strategyproof generalized incremental mechanism for the Steiner tree problem and for the traveling salesman problem.*

Our generalized mechanisms for the above problems match the budget balance factors of previously known cost sharing mechanisms [26, 29, 32], but are inferior in terms of social cost approximation. While the known mechanisms achieve polylogarithmic social cost approximation factors (see [41]), the situation is much worse for our generalized incremental mechanisms: It is not hard to see that the minimum spanning tree game contains the public excludable good problem as a special case and thus inherits the social cost inapproximability of n (see Example 4).

4.2 Makespan Scheduling

In the minimum makespan scheduling problem $P||C_{\max}$, a set of jobs U is to be scheduled on a set of identical parallel machines to minimize the latest completion time of a job, also called the *makespan*. The problem is NP-complete (see [18]). Graham's *largest processing time* (LPT) algorithm [20] is a 4/3-approximation. LPT is a *list scheduling* algorithm: It first orders the jobs by non-increasing processing times and then adds jobs one by one (according to this order) to the current schedule. Every new job is assigned to the machine which currently has the least amount of processing time assigned to it. We use Graham's LPT algorithm to obtain a generalized incremental

mechanism which beats the corresponding lower bound of essentially 2 for Moulin mechanisms [5]. Let $M^{\text{LPT}} := M(\text{LPT}, \tau)$ be the generalized incremental mechanism induced by LPT and the order function τ which sorts the jobs in LPT's list scheduling order. Clearly, M^{LPT} satisfies the properties of Theorem 2.

Corollary 3. *The generalized incremental mechanism M^{LPT} induced by Graham's* LPT *algorithm and τ is weakly group-strategyproof and 4/3-budget balanced for the makespan scheduling problem $P||C_{\max}$.*

The makespan scheduling problem contains the public excludable good problem as a special case and thus M^{LPT} suffers from the same inefficiency as outlined in Example 4. Bleischwitz et al. [6] give a 4/3-budget balanced and $O(\log n)$-approximate weakly group-strategyproof mechanism for the makespan problem, which outperforms our mechanism in terms of social cost approximation.

4.3 Weighted Completion Time Scheduling without Preemption

The weighted completion time scheduling problem without preemption $P||\sum_i w_i C_i$ asks to schedule a set U of n jobs with non-negative weights w_i on m parallel machines such that the total weighted completion time is minimized. Smith's list scheduling algorithm (SM) [44] orders the jobs by non-increasing weight per processing time ratios w_i/p_i and iteratively assigns each job to a machine with smallest total load. It is optimal on a single machine and $(1+\sqrt{2})/2 \approx 1.21$-approximate in the general case [28]. In the unweighted setting, i.e., when $w_i = 1$ for all $i \in U$, it reduces to the *shortest processing time* policy and also delivers an optimal schedule. Even in the unweighted case, no Moulin mechanism can achieve a budget balance factor better than $(n+1)/2$ [7]. Our generalized incremental mechanisms significantly improve upon this. We define the generalized incremental mechanism $M^{\text{SM}} := M(\text{SM}, \tau)$ induced by Smith's rule and the order function τ which orders all jobs in the list scheduling order. M^{SM} satisfies the conditions of Theorem 2.

Corollary 4. *The generalized incremental mechanism M^{SM} induced by Smith's algorithm and τ is weakly group-strategyproof and budget balanced for $P||\sum_i C_i$ and $1||\sum_i w_i C_i$, and 1.21-budget balanced for $P||\sum_i w_i C_i$.*

We will show in Section 5 that M^{SM} achieves constant social cost approximation guarantees for these problems.

4.4 Completion and Flow Time Scheduling with Release Dates and Preemption

In the completion time scheduling problem with release dates and preemption $P|r_i, pmtn|\sum_i C_i$, the goal is to schedule a set U of n jobs on m parallel machines such that the total completion time is minimized. Each job $i \in U$ becomes available at its release date r_i and jobs can be preempted. The *shortest remaining processing time* (SRPT) policy is an effective approximation algorithm for this problem. At any point of time, SRPT executes the m available jobs with the smallest remaining processing times. SRPT computes an optimal schedule in the single-machine case for the total completion time and flow time objective [42]. Sitters [43] very recently showed that SRPT achieves an approximation factor of 1.25 for $P|r_i, pmtn|\sum_i C_i$. Moulin mechanisms cannot achieve a budget balance factor better than $\Omega(n)$ for these problems [7].

We obtain the following superior results. For a given subset of jobs $S \subseteq U$, let $\tau(\cdot, S)$ be the order induced by increasing completion times in the SRPT schedule for S. (If two jobs are completed at the same time, we assume an arbitrary but consistent tie breaking rule.) Let $M^{\text{SRPT}} := M(\text{SRPT}, \tau)$ be the generalized incremental mechanism induced by the SRPT algorithm and τ. We prove in Section 5 that τ is consistent and that SRPT is τ-increasing. Using Theorem 2, we obtain the following theorem.

Corollary 5. *The generalized incremental mechanism M^{SRPT} induced by the* SRPT *algorithm and τ is weakly group-strategyproof and budget balanced for $1|r_i, pmtn|\sum_i F_i$ and $1|r_i, pmtn|\sum_i C_i$, and 1.25-budget balanced for $P|r_i, pmtn|\sum_i C_i$.*

We will prove in Section 5 that this mechanism also achieves constant social cost approximation guarantees for these problems.

5 Completion Time Scheduling: Social Cost Approximation

In this section, we exploit Theorem 3 to prove social cost approximation factors for our generalized incremental mechanisms for scheduling problems with completion time objectives.

5.1 Weighted Completion Time without Preemption

We reconsider the non-preemptive weighted completion time scheduling problems introduced in Section 4. We already showed (Corollary 4) that the generalized incremental mechanism $M^{\text{SM}} := M(\text{SM}, \tau)$ induced by Smith's rule and the offer function τ defined by non-increasing weight per processing time is weakly group-strategyproof and achieves ρ^{SM}-budget balance, where ρ^{SM} is the approximation guarantee of Smith's rule. In this section, we show that M^{SM} approximates social cost by $2\rho^{\text{SM}}$.

Theorem 4. *The generalized incremental mechanism M^{SM} induced by Smith's algorithm and τ is weakly group-strategyproof, ρ^{SM}-budget balanced, and $2\rho^{\text{SM}}$-approximate for the respective weighted completion time scheduling problem without preemption.*

We first prove the following lemma.

Lemma 6. *Let* ALG *be an algorithm for $P||\sum_i w_i C_i$ with cost function $\bar{C}$. Let X and Y be two disjoint sets of jobs. Then, the cost of an optimal schedule for $X \cup Y$ can be bounded by $C(X \cup Y) \leq 2(\bar{C}(X) + \bar{C}(Y))$.*

Proof. We prove the inequality individually for each machine $\hat{M}$. Consider the jobs $\hat{X} \subseteq X$ and $\hat{Y} \subseteq Y$ scheduled on $\hat{M}$ in the runs of ALG on X and Y, respectively. We denote by c_i the completion time of job i in his respective schedule, i.e., $c_i := \bar{C}_i(X)$ for all $i \in \hat{X}$ and $c_i := \bar{C}_i(Y)$ for all $i \in \hat{Y}$.

Consider the schedule which processes all jobs in $\hat{X} \cup \hat{Y}$ on $\hat{M}$ according to non-decreasing c_i. The completion time of a job $i \in \hat{X}$ in this schedule is $c_i + c_{i^*}$, where i^* denotes the last job in $\hat{Y}$ that is processed before i. Since i^* is processed before i, we have $c_i + c_{i^*} \leq 2c_i$. By exchanging the roles of X and Y, we can show the same for the completion time of every job $i \in \hat{Y}$.

Since the cost of an optimal schedule for $X \cup Y$ is at most that of the schedule produced by repeating the above procedure for each machine, we have

$$C(X \cup Y) \leq \sum_{i \in X \cup Y} w_i \cdot 2c_i = 2\Big(\sum_{i \in X} w_i c_i + \sum_{i \in Y} w_i c_i\Big) = 2\big(\bar{C}(X) + \bar{C}(Y)\big).$$

□

We can now prove Theorem 4.

Proof. It follows from Corollary 4 that M^{SM} is weakly group-strategyproof and ρ^{SM}-budget balanced. In order to obtain the social cost approximation guarantee, we show that the induced cost sharing method ξ is weakly monotone. Consider an arbitrary subset $S \subseteq U$ of jobs. Note that the incremental approximate cost share $\xi_i(S)$ of a player $i \in S$ with respect to S equals his completion time in the schedule output by Smith's rule for S. It is not hard to see that $C_i(T) \geq C_i(S)$ for every $i \in S \subseteq T$. Hence, $\sum_{i \in S} \xi_i(T) \geq \sum_{i \in S} \xi_i(S) = \bar{C}(S)$. The social cost approximation factor now follows from Lemma 6 (letting ALG refer to an optimal algorithm) and Theorem 3. □

The following example shows that our social cost analysis is tight, even in the unweighted single machine case.

Example 5. *Consider an instance of the single-machine completion time scheduling problem $1|p_i = 1|\sum_i C_i$ on an even number of n jobs with valuations $v_i = i$ for all $i \in [n] := \{1, \ldots, n\}$. Assume that M^{SM} orders the jobs according to increasing valuations (note that we can enforce this by slightly perturbing the processing times) and thus accepts all jobs. We have $\Pi(S^M) = \bar{C}([n]) = n(n+1)/2$. However, if we exclude the first $n/2$ jobs from the scheduled set, we obtain a social cost of*

$$C\Big(\Big[\frac{n}{2}\Big]\Big) + \sum_{i=1}^{n/2} v_i = 2 \cdot \Big(\frac{n}{4}\Big(\frac{n}{2} + 1\Big)\Big) = \frac{1}{4} n(n+2) \geq \Pi^*.$$

We thus obtain a social cost approximation ratio that approaches 2.

5.2 Completion Time with Release Dates and Preemption

We reconsider the preemptive completion (and flow) time scheduling problems introduced in Section 4. We prove the following result:

Theorem 5. *The generalized incremental mechanism M^{SRPT} induced by the* SRPT *algorithm and τ is weakly group-strategyproof, ρ^{SRPT}-budget balanced, and $4\rho^{\text{SRPT}}$-approximate for the respective completion time scheduling problem with release dates and preemption.*

For the sake of clarity, we first consider the single machine case and comment on the extension of the results given below to the parallel machine case at the end of this section.

5.2.1 Single Machine Case

Consider the problem $1|r_i, pmtn|\sum_i C_i$ of scheduling a set of jobs U on a single machine to minimize the total completion time. The *shortest remaining processing time* (SRPT) policy solves this problem to optimality [42]. Throughout this section, we denote by

$C_i(S)$ the completion time of job $i \in S$ in the SRPT schedule for $S \subseteq U$. Note that by optimality of SRPT, we have $\bar{C}(S) = C(S) = \sum_{i \in S} C_i(S)$. As in Section 4, we define $\tau(\cdot, S)$ to be the order induced by increasing completion times in the SRPT schedule, i.e., $\tau(i, S) := C_i(S)$ for all $i \in S$, and let $M^{\text{SRPT}} := M(\text{SRPT}, \tau)$ be the generalized incremental mechanism induced by SRPT and τ.

The proof of Theorem 5 relies on Lemmas 7 and 8 below. The most work goes into showing that the order function τ is consistent and that SRPT is τ-increasing. However, we defer this part of the proof to the end of this section. Lemma 8 is used to prove the social cost approximation factor.

Lemma 7. *The order function τ is consistent. Moreover,* SRPT *is τ-increasing.*

Lemma 8. *Let* ALG *be an algorithm for $P|r_i, pmtn| \sum_i C_i$ with cost function $\bar{C}$. Let X and Y be two disjoint sets of jobs. Then, the cost of an optimal schedule for $X \cup Y$ can be bounded by $C(X \cup Y) \leq 4(\bar{C}(X) + \bar{C}(Y))$.*

Proof. Phillips et al. [36] prove that any preemptive schedule for $P|r_i, pmtn| \sum_i C_i$ can be turned into a non-preemptive schedule NP with at most twice the cost. With Lemma 6, we obtain $C(X \cup Y) \leq 2(C^{\text{NP}}(X \cup Y)) \leq 4(\bar{C}(X) + \bar{C}(Y))$. □

Assuming that Lemma 7 holds true, we can now prove Theorem 5.

Proof. Lemma 7 together with Theorem 1 imply that M^{SRPT} is weakly group-strategyproof and budget balanced. To prove that M^{SRPT} approximates social cost, we first show that ξ is weakly monotone. Fix some set T and let $S \subseteq T$. Consider the SRPT schedule for T. By removing all jobs in $T \setminus S$ from this schedule, we obtain a feasible schedule for S of cost at most $\sum_{i \in S} C_i(T)$, hence $\sum_{i \in S} C_i(T) \geq C(S)$. Subsequently, it will become clear that the incremental cost share $\xi_i(T)$ of a job $i \in T$ with respect to T is equal to its completion time $C_i(T)$. We conclude that ξ is weakly monotone. Now, the bound on the social cost approximation factor follows from Lemma 8 (letting ALG refer to an optimal algorithm) and Theorem 3. □

It remains to show that the order function τ induced by increasing completion times in the SRPT schedule is consistent and that SRPT is τ-increasing. To this end, we study the effect of removing a single job from the SRPT schedule. We claim the following:

Lemma 9. *Let $T \subseteq U$. Suppose we remove an arbitrary job j from T. Define $S := T \setminus \{j\}$ as the set of remaining jobs. Let $C_i(S)$ and $C_i(T)$ denote the completion times of job $i \in S$ in the* SRPT *schedules for S and T, respectively. Then*

1. *$C_i(S) = C_i(T)$ for every job $i \in S$ with $C_i(T) < C_j(T)$; and*
2. *$C_i(S) \geq C_j(T)$ for every job $i \in S$ with $C_i(T) > C_j(T)$.*

Suppose this lemma holds true. We can then prove that τ is consistent and that SRPT is τ-increasing:

Lemma 7. We first prove consistency. Let $S \subseteq T \subseteq U$ be two subsets ordered by τ as $S = \{i_1, i_2, \ldots, i_p\}$ and $T = \{j_1, j_2, \ldots, j_q\}$. Let k be minimal with $j_k \in T \setminus S$. Define $j := j_k$ to simplify notation. By definition of τ, for every job $i = j_l$ with $1 \leq l < k$, we have $C_i(T) < C_j(T)$. Also, for every job $i = j_r$ with $k < r \leq q$, we have $C_i(T) > C_j(T)$. Thus, by removing job j from T we obtain a new set $T' = T \setminus \{j\}$ such that $C_i(T') = C_i(T)$ for all $i = j_l$ with $1 \leq l < k$ and $C_i(T') \geq C_j(T)$ for all

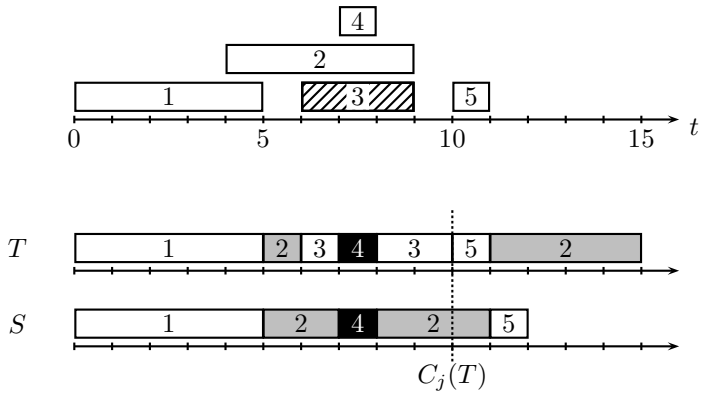

Figure 5: The effect of removing a single job $j = 3$ from the SRPT schedule on $T = \{1, \ldots, 5\}$. The upper part represents the input instance for T; jobs are numbered by increasing release times. The lower part shows the two SRPT schedules for T and $S := T \setminus \{j\}$. The winning and losing jobs are indicated in black and gray, respectively.

$i = j_r$ with $k < r \leq q$. Repeating the above procedure (with T' instead of T), we eventually remove all jobs in $T \setminus S$ from T and conclude that $i_l = j_l$ for all $1 \leq l < k$.

It remains to prove that SRPT is τ-increasing. Consider an arbitrary subset $S \subseteq U$ of jobs and suppose S is ordered by τ as $S = \{i_1, \ldots, i_p\}$. We need to argue that $\bar{C}(S_k) \geq \bar{C}(S_{k-1})$ for every $1 \leq k \leq p$. The proof is by induction on k. For $k = p$ the claim follows since we remove a job $j = i_p$ with $C_j(S) > C_i(S)$ for all $i \in S \setminus \{j\}$ and by Lemma 9, the completion times of all remaining jobs remain the same. Thus $\bar{C}(S_p) - \bar{C}(S_{p-1}) = C_j(S_p) = C_j(S) \geq 0$. Suppose the claim holds true for all $k + 1 \geq \ell$ for some $1 < \ell \leq p$. We show that it remains true for k. Let $j = i_k$. We have $C_j(S) > C_i(S)$ for all $i \in S_{k-1}$. The consistency of τ implies that $C_j(S_k) > C_i(S_k)$ for all $i \in S_{k-1}$. Thus, by Lemma 9, the completion times of all jobs $i \in S_{k-1}$ remain the same if we remove job j from the SRPT schedule for S_k. We conclude that the incremental cost share of player j is exactly its completion time, i.e., $\bar{C}(S_k) - \bar{C}(S_{k-1}) = C_j(S_k) \geq 0$. □

Intuitively, it is relatively easy to verify that Lemma 9 holds true: During the lifetime (i.e., between release and completion time) of job j in the SRPT schedule for T, job j prevents some jobs, call them *losing jobs*, to be executed (because they have a larger remaining processing time) while some other jobs, call them *winning jobs*, prevent j from being executed (because they have a smaller remaining processing time). Clearly, every losing job has a larger completion time than j, while every winning job has a smaller completion time than j. Now suppose we remove job j from the input set and consider the resulting SRPT schedule. There are two crucial insights: (i) nothing changes for the winning jobs, and (ii) whenever j was processed in the SRPT schedule for T, a losing job might now be processed in the SRPT schedule for S; however, this losing job will not be completed before time $C_j(T)$. See Figure 5 for an illustration.

In order to turn this intuition into a formal proof, we first introduce some more notation. Let $e_i(t)$ be the amount of time that has been spent on processing job i up to time t. The *remaining processing* time $x_i(t)$ of job i at time t is $x_i(t) := p_i - e_i(t)$. We call a job i *active* at time t if it has been released but not yet completed at this

time, i.e., $r_i \leq t < C_i$. Let $A(t)$ be the set of jobs that are active at time t. SRPT works as follows: At any time $t \geq 0$, SRPT schedules an active job $i \in A(t)$ with minimum remaining processing time, i.e., $x_i(t) \leq x_k(t)$ for all $k \in A(t)$. We assume that SRPT uses a consistent tie breaking rule, e.g., if $x_i(t) = x_k(t)$ for two different jobs i and k, then schedule the one with smaller index.

Consider the SRPT schedule for a set $T \subseteq U$. Let $i, j \in A(t)$ be two jobs that are active at time t. We define $i \prec_t j$ iff either $x_i(t) < x_j(t)$ or $x_i(t) = x_j(t)$ and $i \leq j$. Note that at any point of time t, SRPT schedules the job $i \in A(t)$ with $i \prec_t j$ for all $j \in A(t)$. Thus, if $i \prec_t j$ for some t, then $i \prec_{t'} j$ for all $t' \in [t, C_i)$. We therefore simply write $i \prec j$ iff there exists a time t with $i \prec_t j$. Let $\sigma(t)$ denote the job that is executed at time t in the SRPT schedule for T; we define $\sigma(t) = \emptyset$ if $A(t) = \emptyset$.

Let $j \in T$ be an arbitrary job and consider the time interval $[r_j, C_j)$. We define the set $\mathcal{C}_j$ of jobs that are *competing* with j as $\mathcal{C}_j := \{i \in T \setminus \{j\} : [r_i, C_i) \cap [r_j, C_j) \neq \emptyset\}$. Note that $j \notin \mathcal{C}_j$. We partition the jobs in $\mathcal{C}_j$ into a set $\mathcal{W}_j$ of *winning jobs* and a set $\mathcal{L}_j$ of *losing jobs* with respect to j: $\mathcal{W}_j := \{i \in \mathcal{C}_j : i \prec j\}$ and $\mathcal{L}_j := \mathcal{C}_j \setminus \mathcal{W}_j$. Intuitively, suppose i and j are both active at some time t. If i is a winning job, then i prevents j from being executed by SRPT. On the other hand, if i is a losing job, then j prevents i from being executed.

We next investigate the effect of removing a job j from T. We use the superscript S if we refer to the SRPT schedule for $S := T \setminus \{j\}$.

Lemma 10. *Consider the two* SRPT *schedules on job sets T and $S := T \setminus \{j\}$. For every job $i \in \mathcal{C}_j$ that is active at time $t \in [r_j, C_j)$,*

$$x_i^S(t) = x_i(t) \text{ if } i \in \mathcal{W}_j \quad \text{and} \quad x_i^S(t) \geq x_j(t) \text{ if } i \in \mathcal{L}_j.$$

Proof. We partition the time interval $[r_j, C_j)$ into a sequence of maximal subintervals $I_1, I_2, \ldots, I_f$ such that the set of active jobs remains the same within every subinterval $I_\ell := [s_\ell, e_\ell)$. We prove by induction over ℓ that the claim holds for every $t \in [r_j, e_\ell)$.

Note that both schedules are identical up to time $r_j = s_1$. If $\sigma(s_1) \neq j$, then both schedules process the same job during I_1 and the claim follows. Suppose $\sigma(s_1) = j$. This implies that $A(s_1) \cap \mathcal{W}_j = \emptyset$ and thus all jobs in $A(s_1) \setminus \{j\} = A^S(s_1)$ are losing jobs. If $A^S(s_1) = \emptyset$, the claim follows. Otherwise, let $k := \sigma^S(s_1)$ be the job that is processed in the schedule for S. Since k is a losing job, we have $x_k^S(s_1) = x_k(s_1) \geq x_j(s_1)$. Since k and j receive the same processing time during I_1 in their respective schedules, the claim holds for all $t \in [r_j, e_1)$.

Now, assume that the claim is true for every $t \in [r_j, e_{\ell-1})$ for some $\ell > 1$. We show that it remains true during the time interval I_ℓ. By the induction hypothesis, $x_i^S(t) = x_i(t)$ for every job $i \in \mathcal{W}_j$ that is active at time $t \in [r_j, e_{\ell-1})$. This implies that a job $j \in \mathcal{W}_i$ is executed at time $t \in [r_j, e_{\ell-1})$ in the schedule for T iff it is executed at time t in the schedule for S. We thus have $A^S(s_\ell) \cap \mathcal{W}_j = A(s_\ell) \cap \mathcal{W}_j$. Moreover, $x_i^S(t) \geq x_j(t)$ for every job $i \in \mathcal{L}_j$ that is active at time $t \in [r_j, e_{\ell-1})$. Since $x_j(t) > 0$ for every $t \in [r_j, C_j)$, every job $i \in \mathcal{L}_j$ that is active at time $t \in [r_j, e_{\ell-1})$ in the schedule for T must also be active at time t in the schedule for S. Thus, $A^S(s_\ell) \cap \mathcal{L}_j = A(s_\ell) \cap \mathcal{L}_j$. We now distinguish two cases:

(i) First, assume $\sigma(s_\ell) =: k \in \mathcal{W}_j$. Job k then has smallest remaining processing time, i.e., $x_k(s_\ell) \leq x_i(s_\ell)$ for all $i \in A(s_\ell)$. We conclude that

$$x_k^S(s_\ell) = x_k(s_\ell) \leq x_i(s_\ell) = x_i^S(s_\ell) \quad \forall i \in A(s_\ell) \cap \mathcal{W}_j = A^S(s_\ell) \cap \mathcal{W}_j$$
$$x_k^S(s_\ell) = x_k(s_\ell) \leq x_j(s_\ell) \leq x_i^S(s_\ell) \quad \forall i \in A(s_\ell) \cap \mathcal{L}_j = A^S(s_\ell) \cap \mathcal{L}_j.$$

Since we assume that SRPT uses a consistent tie breaking rule, this implies that $\sigma^S(s_\ell) = k$ and the claim follows.

(ii) Now, suppose $\sigma(s_\ell) = j$. (Note that $\sigma(s_\ell) \in \mathcal{L}_j$ is impossible.) Then $x_j(s_\ell) \leq x_i(s_\ell)$ for every $i \in A(s_\ell)$ and $A(s_\ell) \cap \mathcal{W}_j = \emptyset$. But then we also have $A^S(s_\ell) \cap \mathcal{W}_j = \emptyset$ and thus $A^S(s_\ell) \subseteq \mathcal{L}_j$. If $A^S(s_\ell) = \emptyset$, the claim follows. Otherwise, let $k := \sigma^S(s_\ell) \in \mathcal{L}_j$ be the job that is executed at time s_ℓ in the schedule for S. Since $x_k^S(s_\ell) \geq x_j(s_\ell)$ and the remaining processing times of k and j in their respective schedules reduce by the same amount during I_ℓ, the claim follows. □

Using Lemma 10, we can now easily prove Lemma 9.

Proof. Let $i \in S$ be a job with $C_i(T) < C_j(T)$. If i is not competing with j, then $r_j \geq C_i$ and thus removing j from the schedule does not change the completion time of i, i.e., $C_i(S) = C_i(T)$. Otherwise, i is competing with j, but since $C_j(T) > C_i(T)$, i is a winning job with respect to j. By Lemma 10, job i is completed at the same time in the SRPT schedules for S and for T and thus $C_i(S) = C_i(T)$.

Next, consider a job $i \in S$ with $C_i(T) > C_j(T)$. The claim clearly holds if $r_i \geq C_j(T)$ since $C_i(S) \geq r_i$. Assume $r_i < C_j(T)$. Then i is competing with j and i is a losing job with respect to j. By Lemma 10, job i cannot be completed before time $C_j(T)$ in the SRPT schedule for S. Thus $C_i(S) \geq C_j(T)$. □

5.2.2 Parallel Machine Case

The crucial insight in the single machine case is Lemma 9. The same property holds in the parallel machine case if we assume a consistent tie breaking rule between jobs with equal remaining processing times. Showing that the computed output set is $4\rho^{\text{SRPT}}$-approximate proceeds exactly along the same lines as in Theorem 5 (in fact, Lemma 8 is formulated for the multiple machine case). The only difference is that SRPT produces a schedule whose total completion time is at most 1.25 times the optimum [43].

6 Connections to Other Frameworks

6.1 Acyclic Mechanisms

Mehta, Roughgarden, and Sundararajan [32] introduced the general framework of acyclic mechanisms, which we briefly review here. Our generalized incremental mechanisms are a subclass of acyclic mechanisms and can be viewed as being complementary to Moulin mechanisms in this framework.

An acyclic mechanism is defined in terms of a *cost sharing method* $\xi : U \times 2^U \to \mathbb{R}$ and an *offer function* τ, which defines for every subset $S \subseteq U$ and every player $i \in S$ a non-negative *offer time* $\tau(i, S)$. The *acyclic mechanism* $A(\xi, \tau)$ induced by ξ and τ receives the bid vector b as input and proceeds as described in Algorithm 2.

For a given subset $S \subseteq U$ and a player $i \in S$, define the following partition of the player set S into three subsets with respect to the offer time of i. Let $L(i, S)$, $E(i, S)$ and $G(i, S)$ be the sets of players with offer times $\tau(\cdot, S)$ strictly less than, equal to, or strictly greater than $\tau(i, S)$, respectively. The following definition is crucial to achieve weak group-strategyproofness.

Definition 4. Let ξ and τ be a cost sharing method and an offer function on U. The offer function τ is *valid* for ξ if the following two properties hold for every subset $S \subseteq U$ and player $i \in S$:

Algorithm 2: Acyclic mechanism $A(\xi, \tau)$ induced by ξ and τ.

Input: Set of players U and bid vector $b = (b_i)_{i \in U}$
Output: Allocation vector $x = (x_i)_{i \in U}$ and payment vector $p = (p_i)_{i \in U}$

1 Initialize $S := U$.
2 **if** $\xi_i(S) \leq b_i$ *for every player* $i \in S$ **then** halt and output the characteristic vector x of S and payments $p := (\xi_i(S))_{i \in U}$.
3 Among all players in S with $\xi_i(S) > b_i$, let i^* be one with minimum $\tau(i, S)$ (breaking ties arbitrarily).
4 Set $S := S \setminus \{i^*\}$ and return to Step 2.

(P1) $\xi_i(S \setminus T) = \xi_i(S)$ for every subset $T \subseteq G(i, S)$;

(P2) $\xi_i(S \setminus T) \geq \xi_i(S)$ for every subset $T \subseteq G(i, S) \cup (E(i, S) \setminus \{i\})$.

A cost sharing method ξ is called *β-budget balanced* if for every subset $S \subseteq U$ we have $\bar{C}(S) \leq \sum_{i \in S} \xi_i(S) \leq \beta \cdot C(S)$. We summarize the main result of Mehta, Roughgarden, and Sundararajan [32] in the following theorem:

Theorem 6 ([32])**.** *Let ξ be a β-budget balanced cost sharing method on U and let τ be an offer function on U that is valid for ξ. Then, the induced acyclic mechanism $A(\xi, \tau)$ is β-budget balanced and weakly group-strategyproof.*

Our interest in generalized incremental mechanisms was initiated by the following simple observations. Consider the offer function τ of an acyclic mechanism. For a given set of players $S \subseteq U$, τ divides S into subsets of players with equal offer times $\tau(\cdot, S)$. We like to think about acyclic mechanisms in terms of such maximal player sets with equal offer times, and call them *clusters*. Depending on the size of these clusters, we can illustrate the landscape of acyclic mechanisms as follows: Towards one end, assume that every set S constitutes its own cluster containing all players in S. Then, Definition 4 reduces to (P2), which is equivalent to the definition of cross-monotonicity (cf. [33]). Hence, acyclic mechanisms with maximum cluster size are Moulin mechanisms. Towards the other end, consider an acyclic mechanism for which all clusters are singletons, i.e., in every set S, every player has a unique offer time. In this case, Definition 4 reduces to (P1) and once a cost share is announced to a player, it can never be changed again.

Following these observations, our *order functions* correspond to offer functions that produce only singleton clusters, i.e., offer functions $\tau(i, S)$ in which each $i \in S$ receives a unique offer time with respect to S. We call this subclass of acyclic mechanisms *singleton mechanisms* (see also [8]). Here we study singleton mechanisms in which every player is charged the *incremental approximate cost share* of adding him to the current solution. It can easily be verified that consistent order functions are valid (according to Definition 4) for the induced incremental approximate cost share functions defined in this paper. Intuitively, the reason is that the cost share of a player only depends on the set of players that precede him in the order of τ. As a consequence, generalized incremental mechanisms fulfill all conditions of Theorem 6 and thus belong to the class of acyclic mechanisms. Bleischwitz et al. [6] showed that acyclic mechanisms are *weakly group-strategyproof against collectors*. As a consequence, our generalized incremental mechanisms also satisfy this slightly stronger truthfulness notion.

6.2 Scheduling with Rejection

It is easy to verify that every cost sharing mechanism that approximates social cost by a factor of α defines an α-approximate algorithm for the underlying scheduling problem with rejection. Let $\mathcal{P}$ be an arbitrary scheduling problem. For every job $i \in U$, let z_i be the rejection penalty for the price-collecting variant of $\mathcal{P}$. We define a cost sharing game on $\mathcal{P}$ by identifying every player's valuation with the penalty of his job, i.e., $v_i := z_i$ for all $i \in U$. An α-approximate mechanism for this cost sharing game outputs a served set of players S^M and a feasible solution of cost $\bar{C}(S^M)$ for this set, with social cost

$$\bar{C}(S^M) + \sum_{i \notin S^M} v_i \leq \alpha \cdot \min_{S \subseteq U} \left(C(S) + \sum_{i \notin S} v_i \right).$$

Now, it is easy to see that the algorithm that schedules S^M and rejects all other jobs outputs an α-approximate solution to the scheduling problem with rejection.

Theorem 7. *Let M be a mechanism that approximates social cost by a factor α for a scheduling problem $\mathcal{P}$. Then M is an α-approximation algorithm for the respective scheduling problem $\mathcal{P}$ with rejection.*

See Table 1 for the approximation ratios that we obtain for the respective scheduling problems with rejection in this paper.

7 Conclusion

We presented a general approach to derive weakly group-strategyproof mechanisms from approximation algorithms. The approach is applicable whenever the approximation algorithm exhibits an order function that is consistent and with respect to which the approximate cost is monotonically increasing. We provided a series of examples showing that many approximation algorithms naturally give rise to such order functions. It turned out that our mechanisms are particularly efficient for completion time scheduling problems. We are confident that our approach can be applied to various other combinatorial optimization problems not considered in this paper. It would be interesting to see more examples for which good social cost approximation guarantees can be proven. The most promising problems in this context seem to be ones with superadditive cost functions.

Our generalized incremental mechanisms belong to the class of *singleton mechanisms* that constitutes a subclass of acyclic mechanisms which can be seen as being complementary to Moulin mechanisms. Although different cost share definitions are conceivable, we concentrated on singleton mechanisms with incremental approximate cost shares in this paper. These type of mechanisms are sufficient to exploit the full strength of existing approximation algorithms for completion time scheduling problems and allow to derive mechanisms with constant budget balance and social cost approximation guarantees. It would be interesting to understand the limitations of singleton mechanisms in general. Moreover, stepping back to the full generality of acyclic mechanisms, some of the most intriguing open problems are to find a general way to construct acyclic mechanisms from approximation algorithms and to find a general property for proving approximate social cost, alike the summability property for Moulin mechanisms (see [40]).

References

[1] F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko, *Approximation schemes for minimizing average weighted completion time with release dates*, Proc. of the 40th Sympos. on the Foundations of Computer Sci., 1999, pp. 32–43.

[2] A. Archer, J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker, *Approximation and collusion in multicast cost sharing*, Games Econ. Behav. **47** (2004), no. 1, 36–71.

[3] N. Bansal, A. Blum, S. Chawla, and K. Dhamdhere, *Scheduling for flow-time with admission control*, In Proc. of the 11th Annual Europ. Sympos. on Algorithms, Lecture Notes in Computer Sci., vol. 2832, Springer, 2003, pp. 43–54.

[4] Y. Bartal, S. Leonardi, A. Marchetti-Spaccamela, and L. Stougie, *Multiprocessor scheduling with rejection*, Proc. of the 7th ACM-SIAM Sympos. on Discrete Algorithms, 1996, pp. 95–103.

[5] Y. Bleischwitz and B. Monien, *Fair cost-sharing methods for scheduling jobs on parallel machines*, Proc. of the 6th Int. Conf. on Algorithms and Complexity, Lecture Notes in Computer Sci., vol. 3998, 2006, pp. 175–186.

[6] Y. Bleischwitz, B. Monien, and F. Schoppmann, *To be or not to be (served)*, Proc. of the 3rd Int. Workshop on Internet and Network Economics, Lecture Notes in Computer Sci., vol. 4858, 2007, pp. 515–528.

[7] J. Brenner and G. Schäfer, *Cost sharing methods for makespan and completion time scheduling*, Proc. of the 24th Int. Sympos. on Theoretical Aspects of Computer Sci., Lecture Notes in Computer Sci., vol. 4393, 2007, pp. 670–681.

[8] ———, *Singleton acyclic mechanisms and their applications to scheduling problems*, Proc. of the 1st Int. Sympos. on Algorithmic Game Theory, Lecture Notes in Computer Sci., vol. 4997, 2008, pp. 315–326.

[9] P. Brucker, *Scheduling algorithms*, Springer, New York, USA, 1998.

[10] D. Bunde, *Scheduling on a single machine to minimize total flow time with job rejections*, Proc. of the 2nd Multidisciplinary Int. Conference on Scheduling: Theory & Applications, 2005, pp. 562–572.

[11] S. Chawla, T. Roughgarden, and M. Sundararajan, *Optimal cost-sharing mechanisms for Steiner forest problems*, Proc. of the 2nd Int. Workshop on Internet and Network Economics, 2006, pp. 112–123.

[12] E. H. Clarke, *Multipart pricing of public goods*, Public Choice **11** (1971), 17–33.

[13] N. Devanur, M. Mihail, and V. Vazirani, *Strategyproof cost-sharing mechanisms for set cover and facility location games*, Proc. of the ACM Conference on Electronic Commerce, 2003.

[14] S. Dobzinski, A. Mehta, T. Roughgarden, and M. Sundararajan, *Is shapley cost sharing optimal?*, Proc. of the 1st Int. Sympos. on Algorithmic Game Theory, Lecture Notes in Computer Sci., vol. 4997, 2008, pp. 327–336.

[15] D. Engels, D. Karger, S. Kolliopoulos, S. Sengupta, R. Uma, and J. Wein, *Techniques for scheduling with rejection*, J. Algorithms **49** (2003), 175–191.

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

[17] J. Feigenbaum, C. Papadimitriou, and S. Shenker, *Sharing the cost of multicast transmissions*, J. Comput. System Sci. **63** (2001), no. 1, 21–41, Special issue on internet algorithms.

[18] M. R. Garey and D. S. Johnson, *Strong NP-completeness results: Motivation, examples and implications*, J. ACM **25** (1978), no. 3, 499–508.

[19] R. Graham, E. Lawler, J. Lenstra, and A. Rinnooy Kan, *Optimization and approximation in deterministic sequencing and scheduling: a survey*, Annals of Discrete Math. **5** (1979), 287–326.

[20] R. L. Graham, *Bounds on multiprocessing timing anomalies*, SIAM J. Appl. Math. **17** (1969), no. 2, 416–429.

[21] J. Green, E. Kohlberg, and J. J. Laffont, *Partial equilibrium approach to the free rider problem*, J. Public Econ. **6** (1976), 375–394.

[22] T. Groves, *Incentives in teams*, Econometrica **41** (1973), 617–631.

[23] A. Gupta, J. Könemann, S. Leonardi, R. Ravi, and G. Schäfer, *An efficient cost-sharing mechanism for the prize-collecting Steiner forest problem*, Proc. of the 18th ACM-SIAM Sympos. on Discrete Algorithms, 2007, pp. 1153–1162.

[24] A. Gupta, A. Srinivasan, and É. Tardos, *Cost-sharing mechanisms for network design*, Proc. of the 7th Int. Workshop on Approximation Algorithms for Combinatorial Optimization Problems, 2004.

[25] N. Immorlica, M. Mahdian, and V. S. Mirrokni, *Limitations of cross-monotonic cost sharing schemes*, Proc. of the 16th ACM-SIAM Sympos. on Discrete Algorithms, 2005, pp. 602–611.

[26] K. Jain and V. Vazirani, *Applications of approximation algorithms to cooperative games*, Proc. of the 33rd ACM Sympos. on Theory of Computing, 2001, pp. 364–372.

[27] R. Juarez, *Group strategyproof cost sharing*, Unpublished manuscript, 2008.

[28] T. Kawaguchi and S. Kyan, *Worst case bound of an LRF schedule for the mean weighted flow time problem*, SIAM J. Computing **15** (1986), no. 4, 1119–1129.

[29] K. Kent and D. Skorin-Kapov, *Population monotonic cost allocations on MSTs*, Proc. of the 6th Int. Conf. on Operational Res., Croatian Oper. Res. Soc., Zagreb, 1996, pp. 43–48.

[30] J. Könemann, S. Leonardi, G. Schäfer, and S. van Zwam, *From primal-dual to cost shares and back: a stronger LP relaxation for the Steiner forest problem*, Automata, Languages and Programming, Lecture Notes in Computer Sci., vol. 3580, Springer, 2005, pp. 930–942.

[31] S. Leonardi and G. Schäfer, *Cross-monotonic cost sharing methods for connected facility location games.*, Theor. Comput. Sci. **326** (2004), no. 1-3, 431–442.

[32] A. Mehta, T. Roughgarden, and M. Sundararajan, *Beyond Moulin mechanisms*, Proc. of the ACM Conference on Electronic Commerce, 2007.

[33] H. Moulin, *Incremental cost sharing: Characterization by coalition strategy-proofness*, Soc. Choice Welfare **16** (1999), 279–320.

[34] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani (eds.), *Algorithmic game theory*, Cambridge University Press, 2007.

[35] M. Pál and É. Tardos, *Group strategyproof mechanisms via primal-dual algorithms*, Proc. of the 44th Sympos. on the Foundations of Computer Sci., 2003, pp. 584–593.

[36] C. Phillips, C. Stein, and J. Wein, *Minimizing average completion time in the presence of release dates*, Math. Programming **82** (1998), 199–223.

[37] Emmanouil Pountourakis and Angelina Vidali, *A complete characterization of group-strategyproof mechanisms of cost-sharing*, Proc. of the 18th Annual Europ. Sympos. on Algorithms, Lecture Notes in Computer Sci., vol. 6346, Springer, 2010, pp. 146–157.

[38] R. Prim, *Shortest connection networks and some generalizations*, Bell System Technical J. **36** (1957), 1389–1401.

[39] K. Roberts, *The characterization of implementable choice rules*, Aggregation and Revelation of Preferences (J. J. Laffont, ed.), North-Holland, 1979.

[40] T. Roughgarden and M. Sundararajan, *New trade-offs in cost-sharing mechanisms*, Proc. of the 38th ACM Sympos. on Theory of Computing, 2006, pp. 79–88.

[41] ———, *Optimal efficiency guarantees for network design mechanisms*, Proc. of the 12th Int. Conf. on Integer Programming and Combinatorial Optimization, 2007, pp. 469–483.

[42] L. Schrage, *A proof of the optimality of the shortest remaining processing time discipline*, Operations Res. **16** (1968), 687–690.

[43] R. Sitters, *Efficient algorithms for average completion time scheduling*, Proc. of the 14th Int. Conf. on Integer Programming and Combinatorial Optimization, 2010, p. to appear.

[44] W. Smith, *Various optimizers for single-stage production*, Naval Res. Logistics Quarterly, vol. 3, 1956, pp. 59–66.

[45] Vijay V. Vazirani, *Approximation algorithms*, Springer, 2004.

[46] W. Vickrey, *Counterspeculations, auctions, and competitive sealed tenders*, J. Finance **16** (1961), no. 1, 8–37.