Cooperative Cost Sharing via Incremental Mechanisms*

Janina Brenner† Guido Schäfer‡

July 29, 2009

Abstract

For many fundamental cooperative cost sharing games, especially when costs are supermodular, it is known that Moulin mechanisms inevitably suffer from poor budget balance factors. Mehta, Roughgarden, and Sundararajan recently introduced *acyclic mechanisms*, which achieve a slightly weaker notion of group-strategyproofness, but leave more flexibility to improve upon the approximation guarantees with respect to budget balance and social cost. In this paper, we provide a very simple but powerful method for turning any ρ-approximation algorithm for a combinatorial optimization problem into a ρ-budget balanced acyclic mechanism. Hence, we show that there is no gap between the best possible approximation guarantees of full-knowledge approximation algorithms and weakly group-strategyproof cost sharing mechanisms. The applicability of our method is demonstrated by deriving mechanisms for scheduling and network design problems which beat the best possible budget balance factors of Moulin mechanisms. By elaborating our framework, we provide means to construct weakly group-strategyproof mechanisms with approximate social cost. The mechanisms we develop for completion time scheduling problems perform surprisingly well by achieving the first constant budget balance and social cost factors.

Keywords: cooperative game theory, mechanism design, cost sharing, Moulin mechanisms, acyclic mechanisms, combinatorial optimization, network design problems, scheduling problems.

JEL Classification: C02 (Mathematical Methods), C61 (Optimization Techniques), C63 (Computational Techniques), C71 (Cooperative Games).

1 Introduction

One of the most fundamental problems in algorithmic mechanism design is to characterize the trade-off between truthfulness (or *incentive compatibility*), efficiency, and polynomial-time computability. 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 [45],

*A preliminary version of this paper appeared in [8]. This work was supported by the DFG Research Center MATHEON "Mathematics for key technologies".

†Full postal address: Technische Universität Berlin, Fakultät II: Mathematik und Naturwissenschaften, Institut für Mathematik (MA 5-1), Straße des 17. Juni 136, 10623 Berlin, Germany. Email: brenner@math.tu-berlin.de.

‡Corresponding author. Full postal address: Centrum Wiskunde & Informatica, Algorithms, Combinatorics and Optimization, Science Park 123, 1098 XG Amsterdam, The Netherlands. Email: g.schaefer@cwi.nl. Phone: +31 20 592 4257. Fax: +31 20 592 4199.

Clarke [12], and Groves [23] is 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 [36] for more details.

In general, the question is to which extent the additional restriction of polynomial-time computability influences the feasibility of game-theoretic objectives such as truthfulness, efficiency, etc. In this paper, we address this question in the context of *cooperative cost sharing*: We consider the problem of devising truthful (direct revelation) mechanisms for single-parameter cost sharing games.

In this setting, we are given a set of players that are interested in receiving a common service, e.g., connectivity to a network. The provision of the service incurs some cost that is specified by a (player-set dependent) cost function. Often, this cost function is given implicitly by the optimal solution cost of an underlying optimization problem. For example, the cost of connecting a set of players in a given network may be given by the cost of an optimal Steiner tree on these players. 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 personal 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 *cooperative* cost sharing games, i.e., 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 (formal definitions will be given in Section 2):

1. *Computational efficiency*: The mechanism runs in polynomial time.
2. *Truthfulness*: The selection and payment scheme implemented by the mechanism guarantee that it is in every player's own self-interest to reveal his private valuation.
3. *(Approximate) budget balance*: The sum of all payments charged to the players equals the cost to establish the service.
4. *(Approximate) social cost*: The selected set of players optimizes a socially desirable objective function.

We remark that for several natural cost sharing games, achieving the budget balance or social cost objectives exactly is tantamount to solving NP-hard optimization problems; additionally, there are various lower bounds on their approximability in the game theoretic context. We therefore relax these objectives and only require that they are met approximately.

In recent years, considerable progress has been made in devising truthful mechanisms for cooperative cost sharing games. Most notably, Moulin [34] proposed a general framework for designing so-called *Moulin mechanisms* that are truthful and (approximately) budget balanced. Moulin mechanisms realize one of the strongest notions of truthfulness, namely *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 every round, the mechanism asks every player whether he is willing to pay a certain cost share or not. It then removes all players who reject their offers from the game and continues with the next round. The mechanism halts when all remaining players accept their offer. Moulin showed that this mechanism is group-strategyproof if the cost shares proposed by the mechanism are *cross-monotonic*, i.e., the cost share of a player does not decrease when some of the other players leave the game.

Most of the cost sharing mechanisms that are currently prevailing in literature are Moulin mechanisms. Designing cross-monotonic cost shares that are approximately budget balanced is often a highly non-trivial task, and a lot of research in recent years has gone into finding such cost shares for several different cost sharing games, including problems such as minimum spanning tree [28, 30] and Steiner tree [28], Steiner forest [11, 31], price-collecting Steiner forest [24], (connected) facility location [25, 32, 37], machine scheduling [5, 7], etc. However, recent negative results showed that for several fundamental cost sharing games, Moulin mechanisms can only achieve a very poor budget balance factor [5, 7, 27, 31, 41]; and this effect is even further amplified if approximate social cost is desired as additional objective [7, 11, 41, 42].

Motivated by these shortcomings, Mehta, Roughgarden, and Sundararajan [33] recently introduced a new class of cost sharing mechanisms called *acyclic mechanisms*. These mechanisms generalize Moulin mechanisms by slightly relaxing the notion of truthfulness from group-strategyproofness to *weak* group-strategyproofness. A mechanism is *weakly group-strategyproof* [13, 33] if no coordinated bidding of a coalition of players can ever strictly increase the utility of *every* player in the coalition. This relaxation opens new ground for improving budget balance and social cost approximation factors. The authors study their new framework mainly in relation to primal-dual algorithms.

In his seminal work [34], Moulin introduced fully budget balanced *(generalized) incremental mechanisms*. Roughly, an incremental mechanism works as follows: It proceeds in rounds; at the beginning of every round, it fixes an order on the set of all remaining players. According to this order, it then asks one player after the other whether he is willing to pay the offered cost share or not. The cost share offered to a player is simply its incremental cost, i.e., the increase in the cost caused by adding this player to the current set of served players. If a player accepts, he is added to the set of players that receive service and is never considered again; otherwise, the player is removed from the game and the mechanism continues with the next round. Moulin states that if the underlying cost function is supermodular, essentially only incremental mechanisms can be group-strategyproof and budget balanced.

1.1 Our Results

The main contributions of this paper are the following:

1. Framework to derive weakly group-strategyproof mechanisms

We provide a framework for deriving weakly group-strategyproof mechanisms for cooperative cost sharing games from approximation algorithms. More precisely, we show how a ρ-approximation algorithm for the underlying optimization problem of a cost sharing game can be turned into an *incremental* cost sharing mechanism that is ρ-budget balanced and prove that this mechanism is weakly group-strategyproof. The construction is very simple and uses the approximation algorithm as a black-box. While previously, most cost sharing mechanisms were developed in case-by-case studies, this is the first general framework for obtaining cost sharing mechanisms from existing approximation algorithms, thereby exploiting the full strength of the latter. As a consequence, we show that there is no gap between the best possible approximation guarantees obtainable by full-knowledge approximation algorithms and weakly group-strategyproof mechanisms.

We use our framework to derive weakly group-strategyproof and approximately budget balanced mechanisms for several scheduling and network design problems. Our examples include minimum makespan and completion time scheduling problems, the minimum spanning tree problem and the minimum Steiner tree problem. The results are summarized in Table 1. A direct

Problem	Incremental mechanism	Moulin mechanism	
$P\|\|C_{\max}$	$\frac{4}{3} - \frac{1}{3m}$	$2^\star$	[5, 7]
$P\|\|\sum C_i$	(1, 2)	$\Omega(n)$	[7]
$P\|\|\sum w_i C_i$	(1.21, 2.42)	$\Omega(n)$	[7]
$1\|r_i, pmtn\|\sum C_i$	(1, 4)	$\Omega(n)$	[7]
$1\|r_i, pmtn\|\sum F_i$	1	$\Omega(n)$	[7]
MST	1	$1^\star$	[28, 30]
Steiner tree	2	$2^\star$	[28, 31]
TSP	2	2	[28]

Table 1: Results of this paper. (β, α) states the budget balance (β) and social cost (α) approximation factors; all other entries refer to budget balance factors; tight corresponding lower bounds are indicated by a *.

consequence of some of the results presented in this paper is that several lower bounds on the budget balance factor of Moulin mechanisms can be overcome by incremental mechanisms. Given our framework, obtaining the respective incremental mechanisms is (almost) trivial, while this is in general not the case for Moulin mechanisms and acyclic mechanisms (cf., e.g., the Moulin mechanism by Jain and Vazirani [28] and the acyclic mechanism by Mehta et al. [33] for the minimum spanning tree problem and the Steiner tree problem).

2. Method to bound approximate social cost and its application to completion time scheduling

We offer refined conditions and additional proof techniques for situations in which cooperative cost sharing mechanisms are not only required to attain attractive budget balance factors, but the concern is also about social cost minimization. We present conditions under which our incremental mechanisms fulfill the no positive transfer property without the need to artificially change negative prices to zero even when the cost function defined by the input algorithm is not increasing. We also provide a method to facilitate proving upper bounds on the social cost approximation factor of incremental mechanisms. Essentially, we identify an additional weak monotonicity property, which, if satisfied by the mechanism, allows to bound its social cost approximation factor.

We demonstrate the full power of this extended framework, also when social cost is concerned, by developing weakly group-strategyproof mechanisms for completion time scheduling problems with and without release dates and preemption. More specifically, using the three-field notation scheme by Graham et al. [20], we achieve 1-budget balance and 2-approximate social cost for $P||\sum C_i$, 1.21-budget balance and 2.42-approximate social cost for $P||\sum w_i C_i$, and 1-budget balance and 4-approximate social cost for $1|r_i, pmtn|\sum C_i$. Not only are these the first cost sharing mechanisms to achieve constant social cost approximation factors, but they also outperform the strong lower bound of $\Omega(n)$ on the budget balance factor of any Moulin mechanism for all completion time related objectives [7].

3. Implications for scheduling problems with rejection

Every mechanism which approximates social cost defines an approximation algorithm for the *price-collecting* variant of the underlying optimization problem; in the scheduling context, these problems are also called scheduling problems *with rejection* (formal definitions are given in Section 2). As a by-product of our game-theoretic results, we therefore obtain constant factor approximation algorithms for several machine scheduling problems with rejection.

Relation to acyclic mechanisms We show that 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, incremental mechanisms can be viewed as complementary to Moulin mechanisms regarding the degree of freedom that the mechanism has for ordering its price proposals to players.

1.2 Related Work

Cost Sharing Moulin [34] presented a framework that allows to obtain budget balanced and group-strategyproof mechanisms for cooperative cost sharing games from cross-monotonic cost sharing methods (cf. also the exposition given by Moulin and Shenker in [35]). Jain and Vazirani [28] observed that this framework can be adapted to achieve approximate budget balance, thereby opening the possibility to realize computational efficiency additionally. Immorlica et al. [27] were able to show that every group-strategyproof cost sharing mechanism that satisfies some additional conditions corresponds to a Moulin mechanism driven by a cross-monotonic cost sharing method.

Intrinsically, an additional objective in cost sharing is to let the mechanism choose an output set that maximizes the *social welfare*, defined as the sum of valuations of all served players minus the servicing cost. However, classical results in economics [22, 40] state that budget balance and social welfare cannot be achieved simultaneously. Feigenbaum et al. [17] showed that for the multicast cost sharing game, these two objectives cannot even be approximated simultaneously; even if only strategyproofness is required (i.e., in the non-cooperative case).

As a consequence, researchers focused on either approximating budget balance or social welfare. Moulin mechanisms with constant budget balance factors have been developed for the cost sharing variants of many classical optimization problems, including fixed tree multicast [2, 17, 18], submodular cost sharing [35], minimum spanning tree [28, 30], Steiner tree [28], price-collecting Steiner tree [24], facility location [37], connected facility location [25, 32, 37], Steiner forest [31], and machine scheduling [5, 7]. On the negative side, lower bounds on the budget balance factor achievable by Moulin mechanism were given in [5, 7, 27, 31, 41].

Recently, Roughgarden and Sundararajan [41] defined an alternative measure of social efficiency that circumvents the intractability results in [17, 22, 40] (at least partially). They define the *social cost* of an output set as the sum of valuations of excluded players plus the servicing cost. (Notice that a served set minimizes social cost if and only if it maximizes social welfare.) With this alternative social efficiency notion, it became possible to approximate both budget balance and social cost simultaneously. The authors also revealed a relation between the social cost approximation factor of a Moulin mechanism and a property of the underlying cost sharing method which they termed *α-summability*.

Following the work of Roughgarden and Sundararajan, the performance of Moulin mechanisms has also been studied with respect to social cost, e.g., for Steiner tree and forest, facility location, single-source rent-or-buy network design and machine scheduling (see [7, 11, 24, 41, 42]). However, for various problems, strong lower bounds on the budget balance and social cost approximation factors that are achievable by Moulin mechanisms exist [5, 7, 11, 27, 31, 41, 42].

Driven by the limitations inherent to cross-monotonic cost sharing methods, Mehta, Roughgarden, and Sundararajan [33] recently introduced a more general framework for designing truthful cost sharing mechanisms, termed *acyclic mechanisms*. These mechanisms implement a slightly weaker notion of truthfulness, called *weak group-strategyproofness*, but therefore leave more flexibility to improve upon the approximation guarantees with respect to budget balance and social cost. The authors demonstrate the applicability of their framework by showing that primal-dual approximation algorithms for several combinatorial optimization problems naturally give rise to acyclic mechanisms that achieve attractive approximation guarantees both with respect to budget balance and social cost.

The basic idea of acyclic mechanisms is to ask players according to an order chosen by the mechanism designer whether they accept an offered cost share or not. Similar to Moulin mechanisms, the crucial condition for truthfulness is that the cost shares offered to a player during the run of the mechanism are non-decreasing. However, since not every player is offered a cost share in every round, acyclic mechanisms provide a lot more flexibility for defining such cost shares. Nevertheless, in order to be non-decreasing, the cost shares must satisfy certain properties which are tied to the order in which players are considered (more details will be given in Section 6.1), and finding such cost shares may still be a highly non-trivial and problem-specific task.

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

Moulin [34] introduced (generalized) incremental mechanisms with full budget balance. He states that if the underlying cost function is supermodular, essentially only incremental mechanisms can be group-strategyproof and budget balanced. On the other hand, if the cost function is submodular, all cross-monotonic cost sharing methods for binary demand games yield group-strategyproof mechanisms. We build on and add to Moulin's work by consolidating the strong theory on approximation algorithms with incremental mechanisms and proving weak group-strategyproofness for the whole generality of incremental mechanisms. We define incremental mechanisms slightly differently than Moulin in that they accept requests in the borderline cases in which a player's bid equals the offered price.

Scheduling The problem of scheduling independent jobs on parallel machines is well-studied for various objective functions. We assume that the reader is familiar with the three-field notation scheme by Graham et al. [20]. The minimum makespan version $P||C_{\max}$ is shown to be NP-complete by Garey and Johnson [19]. Hochbaum and Shmoys [26] gave a polynomial-time approximation scheme (PTAS) for this problem. Graham's *largest processing time* (LPT) algorithm [21] is a $4/3$-approximation. Lenstra proves that the minimum weighted completion time scheduling problem $P||\sum_i w_i C_i$ is NP-complete (see [9]). A PTAS for this problem has been given in [1]. Smith's rule [44] schedules jobs by non-increasing weight per processing time ratios and approximates the problem by a factor of $\frac{1}{2} \cdot (1+\sqrt{2}) \approx 1.21$. For unit processing times or equal weights, Smith's rule delivers an optimal solution. With release dates and preemption, minimizing the sum of (unweighted) completion times $P|r_i, pmtn|\sum_i C_i$ becomes NP-hard [38]. Only the single machine case is solved optimally by the *shortest remaining processing time* (SRPT) algorithm [43]. SRPT is a 2-approximation algorithm for the parallel machine case [15].

In scheduling problems with rejection, the algorithm may choose to schedule only part of the input job set at a certain penalty per omitted job. This setting has been introduced by Bartal et al. [4] for an online minimum makespan scheduling problem. Engels et al. study the offline version for completion time related problems [16]. 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; however, their running-time is only pseudopolynomial 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-complete 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.

1.3 Organization of Paper

We introduce the formal definitions and notations used in this paper in Section 2. In Section 3, we describe our general framework for constructing incremental mechanisms and give a few straightforward examples from the areas of scheduling and network design. We provide more elaborate characterizations and a method for proving approximate social cost for incremental mechanisms in Section 4. This method is used in Section 5 to prove constant budget balance and social cost approximation factors of incremental mechanisms for fundamental min-sum scheduling problems. In Section 6, we draw the connection between incremental mechanisms and acyclic mechanisms and sketch the implications of our results in the area of scheduling with rejection. Finally, we give some concluding remarks in Section 7.

2 Preliminaries

2.1 Cost Sharing

A *binary demand, single parameter 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 $\mathscr{P}$. 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 $\boldsymbol{b} := (b_i)_{i\in U}$ as input and computes a binary allocation vector $\boldsymbol{x} := (x_i)_{i\in U}$ and a payment vector $\boldsymbol{p} := (p_i)_{i\in U}$. Let S^M be the subset of players associated with the allocation vector $\boldsymbol{x}$, 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 two 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 $p_i \le b_i$ if $i \in S^M$.

2. *No positive transfer*: A player is not paid for receiving the service, i.e., $p_i \ge 0$ for all $i \in S^M$.

In addition, the mechanism has to compute a (possibly suboptimal) feasible solution to the underlying optimization problem $\mathscr{P}$ 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 $\beta \ge 1$ if

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

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 $u_i(\boldsymbol{x},\boldsymbol{p}) := v_i x_i - p_i$. Since the outcome $(\boldsymbol{x},\boldsymbol{p})$ computed by the mechanism M solely depends on the bids $\boldsymbol{b}$ 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 $\boldsymbol{b},\boldsymbol{b}'$ with $b_i = v_i$ and $b_j = b'_j$ for all $j \neq i$, we have $u_i(\boldsymbol{x},\boldsymbol{p}) \ge u_i(\boldsymbol{x}',\boldsymbol{p}')$, where $(\boldsymbol{x},\boldsymbol{p})$ and $(\boldsymbol{x}',\boldsymbol{p}')$ are the solutions output by M for bid vectors $\boldsymbol{b}$ and $\boldsymbol{b}'$, respectively. In this paper, we consider *cooperative cost sharing games*, i.e., 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 $\boldsymbol{b}, \boldsymbol{b}'$ with $b_i = v_i$ for every $i \in T$ and $b_i = b_i'$ for every $i \notin T$,

$$u_i(\boldsymbol{x}', \boldsymbol{p}') \geq u_i(\boldsymbol{x}, \boldsymbol{p}) \quad \forall i \in T \qquad \Longrightarrow \qquad u_i(\boldsymbol{x}', \boldsymbol{p}') = u_i(\boldsymbol{x}, \boldsymbol{p}) \quad \forall i \in T.$$

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 $\boldsymbol{b}, \boldsymbol{b}'$ with $b_i = v_i$ for every $i \in T$ and $b_i = b_i'$ for every $i \notin T$,

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

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 *social cost* [41] 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.

2.2 Scheduling

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. [20] to refer to specific scheduling settings.

Scheduling with Rejection Consider an arbitrary scheduling problem $\mathscr{P}$ 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. We call the resulting problem a *scheduling problem with rejection*. (Similar variants for network design problems are usually called *price-collecting*; in the scheduling context with due dates, these variants are sometimes called *scheduling with penalties*.)

3 Incremental Mechanisms

In the following, we describe our general framework for converting approximation algorithms to weakly group-strategyproof cost sharing mechanisms. We call the resulting mechanisms *incremental mechanisms* due to their affinity to Moulin's mechanisms [34]. We then apply our framework to several scheduling and network design problems. Our main result is the following:

Theorem 1. *Let* ALG *be a ρ-approximate algorithm for an optimization problem $\mathscr{P}$. Then, there is a weakly group-strategyproof and ρ-budget balanced cost sharing mechanism for $\mathscr{P}$.*

3.1 Construction and Properties

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 permutation for every subset $S \subseteq U$ by ordering the elements in S with respect to increasing τ-values. For a given approximation algorithm ALG, 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 $\bar{C}(\emptyset) = 0$. In the following, assume that we are given a ρ-approximation algorithm ALG and an order function τ.

The *incremental mechanism* $I(\text{ALG}, \tau)$ induced by ALG and τ receives the bid vector $\boldsymbol{b}$ 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 τ-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 $\boldsymbol{x}$ of the accepted players A and the corresponding payments $\boldsymbol{p}$ (where we implicitly set $p_i = 0$ for all $i \notin A$).

Algorithm 1: Incremental mechanism $I(\text{ALG}, \tau)$ induced by ALG and τ.

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

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 $\boldsymbol{x}$ of A and payments $\boldsymbol{p}$.

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

Lemma 1. *The incremental mechanism $I(\text{ALG}, \tau)$ 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

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

which proves ρ-budget balance. □

We remark that the cost shares assigned to the served players depend on the cost function induced by the approximation algorithm ALG and are not necessarily non-negative. Thus, an incremental mechanism does not necessarily satisfy the *no positive transfer* property. However, it is easy to verify that every approximation algorithm can be turned into an incremental mechanism which fulfills no positive transfer by the following modest modification: We redefine the offered price in Line 4 as $p_{i^*} := \max\{0, \bar{C}(A\cup\{i^*\}) - \sum_{i\in A} p_i\}$. Hence, if the incremental cost of adding a player turns out to be negative, we simply charge the player zero. This mechanism achieves ρ-budget balance if the underlying cost function is montone, i.e. if $C(S) \leq C(T)$ for all $S \subseteq T \subseteq U$, which can be shown by an inductive argument: Whenever a player i^* is added to the current set A, we have $\bar{C}(A\cup\{i^*\}) \leq \sum_{i\in A\cup\{i^*\}} p_i \leq \rho\cdot C(A) \leq \rho\cdot C(A\cup\{i^*\})$. However, since all the incremental mechanisms in this paper are derived from approximation algorithms with non-negative marginal costs, we adhere to our original definition.

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

Proof. Fix a coalition $T \subseteq U$ and a bid vector $\boldsymbol{b}$ 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 $\boldsymbol{b}'$ (while $b_i = b'_i$ for all $i \notin T$). The runs of the incremental mechanism on $\boldsymbol{b}$ and $\boldsymbol{b}'$ 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. □

Theorem 1 follows from Lemmas 1 and 2. The following example shows that in general, incremental mechanisms are not group-strategyproof.

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 the completion time scheduling problem on one machine with unit processing times). Let τ be the offer function which orders players by their index. The induced 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 condition of group-strategyproofness.*

3.2 Direct Applications

Our framework is directly applicable to the cost sharing games of many combinatorial optimization problems. We now state some examples from scheduling and network design.

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. Graham's *largest processing time* (LPT) algorithm is a $4/3$-approximation for this problem [21]. LPT is a *list scheduling* algorithm: it first orders the jobs, in this case 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 an incremental mechanism which beats the corresponding lower bound of essentially 2 for Moulin mechanisms [5]. Let $I^{\text{LPT}} := M(\text{LPT}, \tau)$ be the incremental mechanism induced by LPT and the order function τ which sorts the jobs in LPT's list scheduling order.

Corollary 1. *The incremental mechanism I^{LPT} induced by Graham's* LPT *algorithm is weakly group-strategyproof and $4/3$-budget balanced for the makespan scheduling problem $P||C_{\max}$.*

Non-Preemptive Completion Time Scheduling The weighted completion time scheduling problem $P||\sum w_i C_i$ 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 [29]. 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 $n/2$ [7]. Using incremental mechanisms, we can heavily improve upon this. We define the incremental mechanism $I^{\text{SM}} := M(\text{SM}, \tau)$ induced by Smith's rule and the order function τ which orders all jobs in the list scheduling order.

Corollary 2. *The incremental mechanism I^{SM} induced by Smith's algorithm is weakly group-strategyproof and budget balanced for $P||\sum C_i$ and $1||\sum w_i C_i$, and 1.21-budget balanced for $P||\sum w_i C_i$.*

Preemptive Min-Sum Scheduling If we allow preemption and introduce a release date r_i for every job $i \in U$, the *shortest remaining processing time* (SRPT) policy is a standard algorithm. At any point of time, SRPT executes the m available jobs with the smallest remaining processing times. It is optimal for minimizing the sum of completion or flow times in the single-machine case with no weights [43]. In the unweighted parallel machine case, SRPT approximates the minimum total completion time by a factor of 2 [15]. As for all min-sum scheduling problems, the lower bound for the budget balance factor of Moulin mechanisms is $\Omega(n)$ [7]. We obtain the following strongly 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 $I^{\text{SRPT}} := I(\text{SRPT}, \tau)$ be the incremental mechanism induced by the SRPT algorithm and τ. We prove in Section 5 that the induced cost shares are non-negative.

Corollary 3. *The incremental mechanism I^{SRPT} induced by the* SRPT *algorithm and τ is weakly group-strategyproof and budget balanced for $1|r_i, pmtn|\sum F_i$ and $1|r_i, pmtn|\sum C_i$, and 2-budget balanced for $P|r_i, pmtn|\sum C_i$.*

Spanning Tree, Steiner Tree and TSP Given a connected undirected graph $G = (V, E)$ with edge weights $w_e \geq 0$ for all edges $e \in E$, the *minimum spanning tree problem (MST)* is to find a cycle free subset of edges $\mathscr{T} \subseteq E$ that spans all vertices of G and minimizes $w(\mathscr{T}) := \sum_{e \in \mathscr{T}} w_e$. In the cost sharing setting, we identify each player with a vertex which he wishes to connect to the network, i.e., $U = V$. Prim's algorithm (PRIM) [39] solves the MST problem optimally. It proceeds as follows: Pick an arbitrary vertex $v \in V$ as starting connected component. Then iteratively pick a minimum weight edge that connects a new vertex to the 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. 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 if run on S. We define $I^{\text{PRIM}} := I(\text{PRIM}, \tau)$ as the incremental mechanism induced by PRIM and τ.

Corollary 4. *The incremental mechanism I^{PRIM} induced by Prim's algorithm is weakly group-strategyproof and budget balanced for the minimum spanning tree problem.*

We can use Prim's algorithm to obtain constantly budget balanced cost sharing methods for the Steiner tree and traveling salesman problems. The *Steiner tree problem* asks for a minimum weight tree that spans a subset of prespecified terminal vertices; the tree may contain some non-terminal vertices, called *Steiner 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. We can therefore use the cost sharing mechanism based on Prim's algorithm to obtain 2-budget balanced incremental mechanisms for these problems.

Corollary 5. *Prim's algorithm yields a 2-budget balanced and weakly group-strategyproof incremental mechanism for the Steiner tree problem.*

Proof. Run the cost sharing mechanism $M := I^{\text{PRIM}}$ on the set of terminals. Let $\mathscr{T}$ be the MST computed for the final output set S^M and let p_i^{PRIM} denote the cost share of player $i \in S^M$. We output $\mathscr{T}$ and charge every player $i \in S^M$ a cost share of $p_i = p_i^{\text{PRIM}}$. The sum of the cost shares collected equals the weight $w(\mathscr{T})$ of the MST.

Let $\mathscr{T}^*$ be a minimum weight Steiner tree on S^M. Double every edge of $\mathscr{T}^*$ to obtain an Euler tour on the terminals S^M. By traversing this Euler tour and shortcutting all Steiner vertices, we obtain a spanning tree $\mathscr{T}'$ on S^M whose weight is at most $2w(\mathscr{T}^*)$; note that we can assume without loss of generality that the edge weights satisfy the triangle inequality. Since $\mathscr{T}$ is an optimal MST, we conclude

$$w(\mathscr{T}) \leq w(\mathscr{T}') \leq 2w(\mathscr{T}^*),$$

which proves 2-budget balance. □

Corollary 6. *Prim's algorithm yields a 2-budget balanced and weakly group-strategyproof incremental mechanism for the traveling salesman problem.*

Proof. Run the cost sharing mechanism $M := I^{\text{PRIM}}$ on the set of vertices. Let $\mathscr{T}$ be the MST computed for the final output set S^M and let p_i^{PRIM} be the cost share of player i. Double every edge of $\mathscr{T}$ to obtain an Euler tour on the vertices in S^M. By traversing this Euler tour and shortcutting vertices that have been visited before, we obtain a traveling salesman tour $\mathscr{T}'$ whose weight is at most $2w(\mathscr{T})$. We return $\mathscr{T}'$ and charge every player $i \in S^M$ a cost share of $p_i := 2p_i^{\text{PRIM}}$. Note that $\sum_{i \in S^M} p_i = 2w(\mathscr{T})$.

Let $\mathscr{T}^*$ be a minimum weight traveling salesman tour on S^M. By deleting an arbitrary edge, we obtain a tree spanning all vertices in S^M and thus $w(\mathscr{T}) \leq w(\mathscr{T}^*)$. We have

$$w(\mathscr{T}') \leq 2w(\mathscr{T}) \leq 2w(\mathscr{T}^*),$$

which concludes the proof. □

We remark that the budget balance factors we obtain for the minimum spanning tree, Steiner tree, and traveling salesman problems match those of previously known cost sharing mechanisms for these problems [28, 30, 33]. However, our incremental mechanisms stand out due to their great simplicity.

4 Approximating Social Cost

In Section 3, we introduced the general framework for incremental mechanisms and emphasized that we can use the full power of approximation algorithms to obtain weakly group-strategyproof mechanisms; we undermined this finding with some straightforward and well-studied examples. In this section, we offer additional techniques and theoretical results which will enhance the use of more elaborate order functions and enable us to prove approximate social cost of incremental mechanisms. These characterizations will allow us to obtain completely novel and outstanding results for several completion time scheduling problems, which are presented in Section 5.

It is known that truthful mechanisms cannot approximate social cost by less than logarithmic factors for a large class of cost sharing games, in particular those which contain the *public excludable good* problem, where the serving cost is $C(S) = 1$ for every non-empty subset of players $\emptyset \neq S \subseteq U$ [14]. Due to their paticularly simple structure, there exist even stronger lower bounds for incremental mechanisms on these instances (see Example 2). However, the suffering point of all of the above problems is the high submodularity of their cost functions, i.e. players profit from the presence of other players. These problems are typically well solved by cross-monotonic cost sharing methods which in turn achieve reasonable (poly-logarithmic) social cost approximation factors. The class of problems where Moulin mechanisms fail are those with supermodular or superadditive costs, as e.g. min sum scheduling problems [7]. This is where incremental mechanisms come into play. In this Section, we present not only the first sublinearly budget balanced cost sharing mechanisms for comletion time scheduling, but also the very first mechanisms to achieve constant social cost approximation factors in general.

4.1 No Positive Transfer

As we have mentioned, incremental mechanisms in our original definition do not guarantee the *no positive transfer* property. This property is easily seen to be fulfilled if the input approximation algorithm is increasing, i.e., $\bar{C}(S) \leq \bar{C}(T)$ for all $S \subseteq T \subseteq U$. However, a much weaker restriction on the approximation algorithm suffices if we somewhat constrain the order function τ. This restriction will enable us later to find clever and well-performing combinations of algorithms and order functions.

Consider a set $S \subseteq U$ and order the players in S according to increasing $\tau(\cdot, S)$ values. Let $S =: \{i_1, \ldots, i_p\}$ be the ordered set, i.e., $\tau(i_k, S) < \tau(i_l, S)$ for all $1 \leq k < l \leq p$. We also say $S = \{i_1, \ldots, i_p\}$ *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; we 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 1 illustrates the restriction imposed by the consistency property. The most simple example of a consistent order function is one in which the order of every subset of players is induced by a fixed global order on U.

Consider the execution of the incremental mechanism $I(\textsc{alg}, \tau)$ induced by $\textsc{alg}$ 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 3. *At the beginning of every iteration of* $I(\textsc{alg}, \tau)$*, we have* $R_{|A|} = A$.

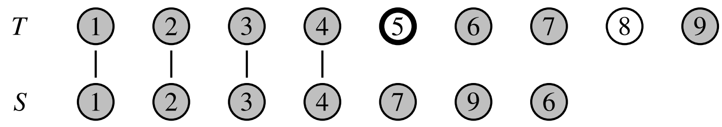

Figure 1: Illustration of the consistency property for two sets $S \subseteq T := \{1,\dots,9\}$; elements belonging to S are depicted in gray. Both sets are ordered according to τ. Note that consistency requires that S has to be ordered like T up to the first element of T that is missing in S (indicated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

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 incremental mechanism $I(\text{ALG},\tau)$ coincides with the order induced by τ on the final output set S^M.

Corollary 7. *Let S^M be the set of players output by $I(\text{ALG},\tau)$. During the course of $I(\text{ALG},\tau)$, players are added to A by increasing $\tau(\cdot,S^M)$-values.*

Proof. Suppose S^M is ordered by τ as $S^M =: \{i_1,\dots,i_p\}$. Let i be the kth player that is added to A in the execution of $I(\text{ALG},\tau)$. 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 3, $R_{k-1} = A$ 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 $i = i_k$. □

Corollary 7 has an important consequence: If the cost function $\bar{C}(\cdot)$ of the approximation algorithm ALG does not decrease as players are added to S^M one by one (in the order of τ), then the final payments charged by the incremental mechanism $I(\text{ALG},\tau)$ to the players in S^M are non-negative. However, since potentially every subset $S \subseteq U$ might be chosen as the output set, we have to 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 $\mathscr{P}$. We say that ALG is *τ-increasing* if for every $S \subseteq U$ and $1 \leq k \leq |S|$, we have $\bar{C}(S_k) \geq \bar{C}(S_{k-1})$.

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

The following theorem now follows directly from Corollary 7 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 $\mathscr{P}$. Then, the incremental mechanism $I(\text{ALG}, \tau)$ is a weakly group-strategyproof and ρ-budget balanced cost sharing mechanism for $\mathscr{P}$, which satisfies the no positive transfer property.*

Given an approximation algorithm ALG, the budget balance factor of $I(\text{ALG}, \tau)$ is independent of the order function τ used. However, the choice of the order function may very well influence the social cost of the output solution. If the cost function $\bar{C}$ induced by ALG is increasing, i.e., $\bar{C}(S) \leq \bar{C}(T)$ for all $S \subseteq T \subseteq U$, we can choose τ solely to achieve a good social cost approximation factor. If not, the no positive transfer property restricts the choice of τ to consistent offer functions with respect to which ALG is increasing.

4.2 Bounding Social Cost

An important criterion for the performance of a cost sharing mechanism is the social cost approximation factor incurred by its choice of the served player set. For proving a social cost approximation guarantee for a mechanism M, we need to upper bound the ratio between the social cost of the set S^M chosen by the mechanism and the cost of a socially optimal set

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

In this section, we present a tool for bounding the social cost approximation factor for incremental mechanisms that fulfill a weak monotonicity property 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 $S \subseteq T \subseteq U$, $\sum_{i \in S} \xi_i(T) \geq \bar{C}(S)$.

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

Theorem 3. *Let τ be a consistent order function and* 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 incremental mechanism $I(\text{ALG}, \tau)$ approximates social cost by a factor of α if*

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

Proof. We can bound the social cost approximation factor by

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

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

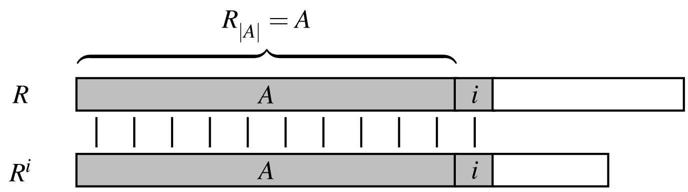

Figure 2: Illustration of the consistency property as used in the proof of Theorem 3.

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 3, 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 2 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

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

□

For some cost functions, incremental mechnanisms cannot be expected to achieve attractive social cost approximation factors. A prominent example is the public excludable good problem, where the serving cost is $C(S) = 1$ for every non-empty subset of players $\emptyset \neq S \subseteq U$.

Example 2. *Consider an instance of the public excludable good problem with $|U| = n$ players. Set $v_i = 1 - \varepsilon$ for an arbitrarily small constant $\varepsilon > 0$. Assuming truthful bidding, any incremental mechanism serves the empty set, incurring a social cost of $\Pi(\emptyset) \approx 0 + n = n$, whereas serving the whole set induces a social cost of $\Pi(U) = 1 - 0 = 1$. We obtain a social cost approximation factor of n.*

5 Completion Time Scheduling

In this section, we study the performance of incremental mechanisms for parallel machine scheduling problems with total completion time objectives, also taking into account their social cost approximation guarantees. We distinguish between the model with weights in which all jobs arrive at time zero and no preemption is allowed, and the model in which jobs have release dates and may be preempted.

5.1 Weighted Completion Time

We reconsider the (weighted) completion time scheduling problems introduced in Section 3.2. We already showed that the incremental mechanism $I^{\text{SM}} := I(\text{SM}, \tau)$ 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 I^{SM} also achieves a surprisingly small social cost approximation factor.

Theorem 4. *The incremental mechanism I^{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.*

We first prove the following lemma.

Lemma 4. *Let* ALG *be an algorithm for $P||\sum w_i C_i$ 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 of Theorem 4. It follows from Corollary 2 that I^{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 4 and Theorem 3. □

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

Example 3. *Consider an instance of* $1|p_i = 1|\sum C_i$ *on an even number of n jobs with valuations* $v_i = i$ *for all* $i \in [n]$*. Assume that* I^{SM} *orders the jobs according to increasing valuations (note that we can easily enforce this by slightly perturbing the processing times) and thus accepts all jobs. Consequently,* $\Pi(S^M) = \bar{C}([n]) = n(n+1)/2$*. However, if we exclude the first* $n/2$ *jobs from the scheduled set, we obtain a social cost of*

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

yielding a social cost approximation ratio that approaches 2.

5.2 Completion Time 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.

Single Machine Case Consider the problem $1|r_i, pmtn|\sum C_i$ 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 [43]. 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 3.2, 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 $i \in S$. Let $I^{\text{SRPT}} := I(\text{SRPT}, \tau)$ be the incremental mechanism induced by SRPT and τ. In this section, we prove the following theorem.

Theorem 5. *The incremental mechanism* I^{SRPT} *induced by* SRPT *and* τ *is weakly group-strategyproof, budget balanced, and* 4*-approximate for* $1|r_i, pmtn|\sum C_i$*.*

The proof of Theorem 5 relies on Lemmas 5 and 6 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 6 is used to prove the social cost approximation factor.

Lemma 5. *The order function* τ *is consistent. Moreover,* SRPT *is* τ*-increasing.*

Lemma 6. *Let* ALG *be an algorithm for* $P|r_i, pmtn|\sum C_i$ *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. [38] prove that any preemptive schedule for $P|r_i, pmtn|\sum C_i$ can be turned into a non-preemptive schedule NP with at most twice the cost. With Lemma 4, 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 5 holds true, we can now prove Theorem 5.

Proof of Theorem 5. Lemma 5 together with Theorem 1 imply that I^{SRPT} is weakly group-strategyproof and budget balanced. To prove that I^{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 6 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 7. *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:

Proof of Lemma 5. 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 $i = j_r$ 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 7, 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 7, 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 7 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 3 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,

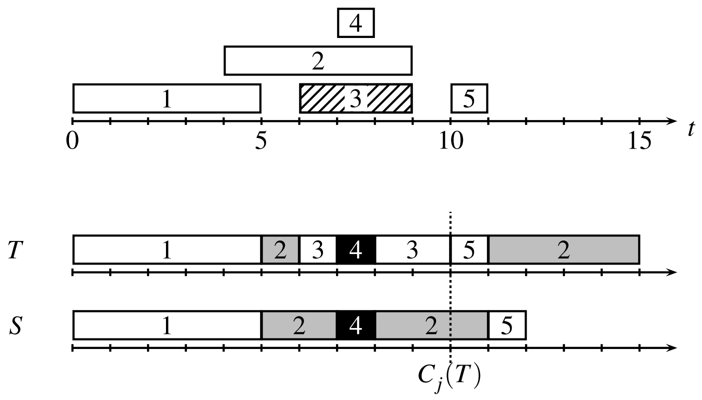

Figure 3: The effect of removing a single job $j = 3$ from the SRPT schedule on $T = \{1, \dots, 5\}$. The upper part represents the input instance for T; jobs are numbered by increasing release times. The lower part shows the two SRPT schedules for T and $S := T \setminus \{j\}$. The winning and losing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

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 $\mathscr{C}_j$ of jobs that are *competing* with j as $\mathscr{C}_j := \{i \in T \setminus \{j\} : [r_i, C_i) \cap [r_j, C_j) \neq \emptyset\}$. Note that $j \notin \mathscr{C}_j$. We partition the jobs in $\mathscr{C}_j$ into a set $\mathscr{W}_j$ of *winning jobs* and a set $\mathscr{L}_j$ of *losing jobs* with respect to j: $\mathscr{W}_j := \{i \in \mathscr{C}_j : i \prec j\}$ and $\mathscr{L}_j := \mathscr{C}_j \setminus \mathscr{W}_j$. 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 8. *Consider the two* SRPT *schedules on job sets T and $S := T \setminus \{j\}$. For every job $i \in \mathscr{C}_j$ that is active at time $t \in [r_j, C_j)$,*

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

Proof. We partition the time interval $[r_j, C_j)$ into a sequence of maximal subintervals $I_1, I_2, \dots, I_f$ 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 $A(s_1) \cap \mathscr{W}_j = \emptyset$ 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 $i \in \mathscr{W}_j$ that is active at time $t \in [r_j, e_{\ell-1})$. This implies that a job $i \in \mathscr{W}_j$ 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 $A^S(s_\ell) \cap \mathscr{W}_j = A(s_\ell) \cap \mathscr{W}_j$. Moreover, $x_i^S(t) \geq x_j(t)$ for every job $i \in \mathscr{L}_j$ 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 $i \in \mathscr{L}_j$ 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, $A^S(s_\ell) \cap \mathscr{L}_j = A(s_\ell) \cap \mathscr{L}_j$. We now distinguish two cases:

(i) First, assume $\sigma(s_\ell) =: k \in \mathscr{W}_j$. Job k then has smallest remaining processing time, i.e.,

$x_k(s_\ell) \le x_i(s_\ell)$ for all $i \in A(s_\ell)$. We conclude that

$$x_k^S(s_\ell) = x_k(s_\ell) \le x_i(s_\ell) = x_i^S(s_\ell) \;\; \forall i \in A(s_\ell) \cap \mathscr{W}_j = A^S(s_\ell) \cap \mathscr{W}_j$$
$$x_k^S(s_\ell) = x_k(s_\ell) \le x_j(s_\ell) \le x_i^S(s_\ell) \;\; \forall i \in A(s_\ell) \cap \mathscr{L}_j = A^S(s_\ell) \cap \mathscr{L}_j.$$

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 $\sigma(s_\ell) \in \mathscr{L}_j$ is impossible.) Then $x_j(s_\ell) \le x_i(s_\ell)$ for every $i \in A(s_\ell)$ and $A(s_\ell) \cap \mathscr{W}_j = \emptyset$. But then we also have $A^S(s_\ell) \cap \mathscr{W}_j = \emptyset$ and thus $A^S(s_\ell) \subseteq \mathscr{L}_j$. If $A^S(s_\ell) = \emptyset$, the claim follows. Otherwise, let $k := \sigma^S(s_\ell) \in \mathscr{L}_j$ be the job that is executed at time s_ℓ in the schedule for S. Since $x_k^S(s_\ell) \ge x_j(s_\ell)$ 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 8, we can now easily prove Lemma 7.

Proof of Lemma 7. Let $i \in S$ be a job with $C_i(T) < C_j(T)$. If i is not competing with j, then $r_j \ge C_i$ 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 8, 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 $r_i \ge C_j(T)$ since $C_i(S) \ge r_i$. Assume $r_i < C_j(T)$. Then i is competing with j and i is a losing job with respect to j. By Lemma 8, job i cannot be completed before time $C_j(T)$ in the SRPT schedule for S. Thus $C_i(S) \ge C_j(T)$. ☐

Parallel Machine Case The crucial insight in the single machine case is Lemma 7. 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-approximate proceeds exactly along the same lines as in Theorem 5 (in fact, Lemma 6 is formulated for the multiple machine case). The only difference is that SRPT produces a schedule whose total completion time is at most twice the optimum. We conclude with the following theorem:

Theorem 6. *The incremental mechanism I^{SRPT} induced by the* SRPT *algorithm and τ is weakly group-strategyproof,* 2*-budget balanced and* 4*-approximate for $P|r_i, pmtn|\sum C_i$.*

6 Connections to Other Frameworks

6.1 Acyclic Mechanisms

Our incremental mechanisms were motivated by and are a subclass of acyclic mechanisms. In a certain sense, they can be viewed as being complementary to Moulin mechanisms in the scope of acyclic mechanisms. In this section, we briefly review the definition of acyclic mechanisms by Mehta, Roughgarden, and Sundararajan [33] and discuss how incremental mechanisms fit into their framework.

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 $\boldsymbol{b}$ 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

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

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

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 $\boldsymbol{x}$ of S and payments $\boldsymbol{p} := (\xi_i(S))_{i \in U}$.
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.

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 $i \in S$:

(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 [33] in the following theorem:

Theorem 7 ([33])**.** *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.*

Relation to Incremental Mechanisms Our interest in 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 consists of one big cluster that contains all players in S. Then, Definition 4 reduces to (P2), which is exactly equivalent to the definition of cross-monotonicity (cf. [34]). 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. This is exactly the subclass of acyclic mechanisms that we decided to study.

Following these observations, we defined *order functions* to be offer functions that produce only singleton clusters, i.e., offer functions $\tau(i, S)$ in which each $i \in S$ receives a distinct offer time with respect to S, or, in other words, that are injective in i for every fixed S. We studied this special case of acyclic mechanisms and termed the cost sharing mechanisms that are induced by order functions and incremental approximate cost shares *singleton mechanisms*.

In this paper, we study the subclass of singleton mechanisms in which every player is charged the *incremental* cost of adding him to the current solution, i.e. *incremental* mechanisms. It can easily be verified that consistent order functions are valid for the induced incremental cost sharing methods 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, incremental mechanisms fulfill all properties of acyclic mechanisms, including, e.g., *weak group-strategyproofness against collectors*, which was identified by Bleischwitz et al. [6].

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 optimization problem with rejection. Along with our results in mechanism design, we therefore obtain several approximation algorithms for scheduling problems with rejection.

We sketch the reduction at the example of a scheduling problem. Let $\mathscr{P}$ be an arbitrary scheduling problem. For every job $i \in U$, let z_i be the rejection penalty for the price-collecting variant of $\mathscr{P}$. We define a cost sharing game on $\mathscr{P}$ 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.

Thus, the following results are immediate consequences of our mechanisms presented in Section 5.

Theorem 8. *The incremental mechanism I^{SM} induced by Smith's rule defines a* 2.42*-approximate algorithm for the weighted completion time scheduling problem $P||\sum w_i C_i$ with rejection, and* 2*-approximate algorithms for the scheduling problems $P||\sum C_i$ and $1||\sum w_i C_i$ with rejection.*

Theorem 9. *The incremental mechanism I^{SRPT} based on the* SRPT *policy defines a 4-approximate algorithm for the completion time scheduling problem $P|r_i, pmtn|\sum C_i$ with rejection.*

7 Conclusion

We introduced singleton mechanisms as a subclass of acyclic mechanisms that is complementary to Moulin mechanisms with respect to the size of *clusters*, i.e., maximal player sets whose order is undetermined beforehand. Remark that in this paper, we concentrated solely on incremental mechanisms with *incremental approximate cost shares*. For this type of singleton mechanisms, we gave a very general construction technique which allows to benefit from the whole range of the enormous theory on approximation algorithms.

We are confident that our proposed transformation technique can be applied for various combinatorial optimization problems. It would be interesting to see more examples for which social cost can be approximated. Here, the most promising problems are ones with supermodular cost functions, i.e. where congestion effects occur. Concurrently, these are the problems for which Moulin mechanisms usually perform only poorly.

Besides, we are interested in other mechanisms with singleton clusters. Stepping back to the full generality of acyclic mechanisms, some of the most interesting open problems are to find a general way to construct acyclic mechanisms from approximation algorithms and to find a general property for proving α-approximate social cost, alike the summability property for Moulin mechanisms.

We would like to thank the anonymous referees for helpful suggestions and comments.

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. In *Proc. of the 40th Sympos. on the Foundations of Computer Sci.*, pages 32–43, 1999.

[2] A. Archer, J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker. Approximation and collusion in multicast cost sharing. *Games Econ. Behav.*, 47(1):36–71, 2004.

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

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

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

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

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

[8] J. Brenner and G. Schäfer. Singleton acyclic mechanisms and their applications to scheduling problems. In *Proc. of the 1st Int. Sympos. on Algorithmic Game Theory*, volume 4997 of *Lecture Notes in Computer Sci.*, pages 315–326, 2008.

[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. In *Proc. of the 2nd Multidisciplinary Int. Conference on Scheduling: Theory & Applications*, pages 562–572, 2005.

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

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

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

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

[15] J. Du, J. Y. T. Leung, and G. H. Young. Minimizing mean flow time with release time constraint. *Theoretical Computer Sci.*, 75(3):347–355, 1990.

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

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

[18] J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *J. Comput. System Sci.*, 63(1):21–41, 2001. Special issue on internet algorithms.

[19] M. R. Garey and D. S. Johnson. Strong NP-completeness results: Motivation, examples and implications. *J. ACM*, 25(3):499–508, 1978.

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

[21] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.*, 17(2):416–429, 1969.

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

[23] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

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

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

[26] D. Hochbaum and D. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *J. ACM*, 34(1):144–162, 1987.

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

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

[29] T. Kawaguchi and S. Kyan. Worst case bound of an LRF schedule for the mean weighted flow time problem. *SIAM J. Computing*, 15(4):1119–1129, 1986.

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

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

[32] S. Leonardi and G. Schäfer. Cross-monotonic cost sharing methods for connected facility location games. *Theor. Comput. Sci.*, 326(1-3):431–442, 2004.

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

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

[35] H. Moulin and S. Shenker. Strategyproof sharing of submodular costs: budget balance versus efficiency. *Econ. Theory*, 18(3):511–533, 2001.

[36] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.

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

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

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

[40] K. Roberts. The characterization of implementable choice rules. In J. J. Laffont, editor, *Aggregation and Revelation of Preferences*. North-Holland, 1979.

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

[42] T. Roughgarden and M. Sundararajan. Optimal efficiency guarantees for network design mechanisms. In *Proc. of the 12th Int. Conf. on Integer Programming and Combinatorial Optimization*, pages 469–483, 2007.

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

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

[45] W. Vickrey. Counterspeculations, auctions, and competitive sealed tenders. *J. Finance*, 16(1):8–37, 1961.