

Sharing Supermodular Costs

Andreas S. Schulz^a

Nelson A. Uhan^b

Original submission: August 27, 2007

Revised submissions: April 16, 2008; February 18, 2009

Abstract

In this work, we apply ideas from cooperative game theory to study situations in which a set of agents face supermodular costs. These situations appear in a variety of scheduling contexts, as well as in some settings related to facility location and network design. Intuitively, cooperation amongst rational agents who face supermodular costs is unlikely. However, in circumstances where the failure to cooperate may lead to negative externalities, one might be interested in methods of encouraging cooperation. The least core value of a cooperative game is the minimum penalty we need to charge a coalition for acting independently that encourages cooperation by ensuring the existence of an efficient and stable cost allocation. The set of all such cost allocations is called the least core. In this paper, we study the computational complexity and approximability of computing the least core value of supermodular cost cooperative games. We show that computing the least core value of supermodular cost cooperative games is strongly NP-hard, and build a framework to approximate the least core value of these games using oracles that approximately determine maximally violated constraints. This framework yields a $(3 + \epsilon)$ -approximation algorithm for computing the least core value of supermodular cost cooperative games. As a by-product, we show how to compute accompanying approximate least core cost allocations for these games. We also apply our approximation framework to obtain better results for two particular classes of supermodular cost cooperative games that arise from scheduling and matroid optimization.

^aSloan School of Management and Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139.
e-mail: schulz@mit.edu

^bSchool of Industrial Engineering, Purdue University, West Lafayette, IN 47907. e-mail: nuhan@purdue.edu

1 Introduction

Consider a situation in which a set of agents has the option of sharing the cost of their joint actions. For example, a group of retailers, instead of individually managing each of their own storage facilities, may decide to jointly participate in a centralized inventory management scheme with a common storage facility, and share the cost of optimally running this facility. In these situations, the agents may or may not be motivated to cooperate, depending on the structure of their costs. Cooperative game theory offers a mathematical framework to study the cooperative behavior between multiple agents. A (*transferable utility*) *cooperative game* is a pair (N, v) where $N = \{1, \dots, n\}$ represents a set of agents, and $v : 2^N \rightarrow \mathbb{R}$ is a set function where for each $S \subseteq N$, $v(S)$ represents the cost to agents in S if they cooperate. By convention, $v(\emptyset) = 0$. A subset $S \subseteq N$ of agents is referred to as a *coalition*.

Cooperative game theory has been used extensively to study cost sharing for a myriad of application areas of operations research. For example, one increasingly popular stream of research has focused on the application of cooperative game theory to various problems in inventory management (e.g. Gerchak and Gupta 1991; Hartman et al. 2000; van den Heuvel et al. 2005; Chen and Zhang 2006). Another body of literature has used cooperative game theory to investigate the cost sharing issues in various scheduling-related problems (e.g. Curiel et al. 1989; Maniquet 2003; Mishra and Rangarajan 2005). Other applications of cooperative game theory to OR-related areas include assignment games (Shapley and Shubik 1971), linear production games (Owen 1975), minimum-cost spanning tree games (Bird 1976; Granot and Huberman 1981), network flow games (Kalai and Zemel 1982a,b), traveling salesman games (Potters et al. 1991), and facility location games (Goemans and Skutella 2004).

A significant part of cooperative game theory is centered around the question of how to distribute costs that are collectively incurred by a group of cooperating agents in a way that meets various desirable criteria. For example, one might want to allocate costs in a way that is “fair” or “stable.” Such cost allocation rules are called *solution concepts*. Different notions of desirable cost allocation properties lead to different solution concepts. One of the most prominent solution concepts in cooperative game theory is the *core* (Gillies 1959). Suppose $x \in \mathbb{R}^N$ is a cost allocation vector: for all $i \in N$, x_i represents the cost allocated to agent i . (For notational convenience, for any vector x we define $x(S) = \sum_{i \in S} x_i$ for any $S \subseteq N$.) The core of a cooperative game (N, v) is the set of all cost allocations x such that

$$x(N) = v(N), \tag{1.1a}$$

$$x(S) \leq v(S) \quad \text{for all } S \subseteq N. \tag{1.1b}$$

The condition (1.1a) requires that a cost allocation in the core is *efficient*: the total cost allocated to all agents, $x(N)$, is equal to the cost to all agents when they cooperate, $v(N)$. The conditions (1.1b) guarantee that a cost allocation in the core is “subgroup rational” or *stable*: no subset of agents, or coalition, would be better off by abandoning the rest of the agents and acting on its own. In other words, the core of a cooperative game is the set of all efficient and stable cost allocations. The existence of an efficient and stable cost allocation—in other words, a non-empty core—can be seen as a rudimentary indication that cooperation is attainable.

For many cooperative games, the core may be empty. Another solution concept, initially proposed by Shapley and Shubik (1966) and later named by Maschler et al. (1979) is called the *least core*. The least core of a cooperative game (N, v) is the set of cost allocations x that are optimal solutions to the linear program

$$\begin{aligned} \text{[LC]} \quad z^* = & \text{minimize } z \\ & \text{subject to } x(N) = v(N), \\ & x(S) \leq v(S) + z \quad \text{for all } S \subseteq N, S \neq \emptyset, N. \end{aligned}$$

The optimal value z^* of [LC] is the *least core value* of the game (N, v) . The linear program [LC] can be

equivalently written as

$$z^* = \min_{x: x(N)=v(N)} \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} e(x, S),$$

where $e(x, S) = x(S) - v(S)$ for all $S \subseteq N$. Note that the least core is always well-defined and non-empty, regardless of whether the core is empty or non-empty¹.

The least core and the least core value of a cooperative game have several interesting economic interpretations. The quantity $e(x, S)$ is the *dissatisfaction*² of a coalition S under a cost allocation x : it is the extra cost that S pays when costs are allocated according to x . A cost allocation in the least core therefore minimizes the maximum dissatisfaction of any coalition. In this sense, the least core contains those cost allocations that are “least objectionable.” If the core of a cooperative game is non-empty, the dissatisfaction of any coalition under a cost allocation in the core is nonpositive, and the absolute value of a coalition’s dissatisfaction represents its gain under such a cost allocation. In this case, the least core is a refinement of the core: the least core contains all efficient and stable cost allocations that maximize the minimum gain of any coalition. If the core of a cooperative game is empty, the least core value z^* can be viewed as the minimum penalty we need to charge a coalition for acting independently that encourages cooperation by ensuring the existence of an efficient and stable cost allocation. Alternatively, the least core value can be seen as the minimum cost of coalition formation for which exercising the “subgroup rationality” implied by (1.1b) is not considered worth the trouble.

The least core is closely related to some other solution concepts from cooperative game theory. For example, Maschler et al. (1979) studied the relationship between the least core and the *nucleolus* (Schmeidler 1969). Given a cost allocation $x \in \mathbb{R}^N$, the excess vector $\theta(x)$ is defined as the $2^n - 2$ dimensional vector whose components are $e(x, S)$ for all $S \subseteq N, S \neq \emptyset, N$, in nonincreasing order. The nucleolus is the cost allocation that lexicographically minimizes the excess vector $\theta(x)$. Maschler et al. (1979) showed that for a cooperative game (N, v) , the nucleolus can be computed by solving $|N|$ linear programs of the form [LC]. On a different note, Einy et al. (1999) showed that the least core is always contained in the *Mas-Colell bargaining set* (Mas-Colell 1989). The Mas-Colell bargaining set of a cooperative game, like the classical bargaining set (Aumann and Maschler 1964), is the set of all cost allocations that are stable with respect to a particularly defined system of objections and counterobjections.

Although the least core and the least core value have various appealing properties, they have their drawbacks as well. For example, one common criticism of solution concepts like the least core is that it is not necessarily unique, and as a result, there is typically no definite outcome prescribed by the least core. The lack of a unique solution, however, may allow for flexibility in practice. Another drawback of the least core is that like the core, it is based on a near-sighted “one-step” notion of stability. It may be possible that an initial defection by a coalition can prompt a series of additional defections by other coalitions, eventually leading to a situation in which the defecting parties are in fact worse off than before. Depending on the circumstances, a far-sighted notion of stability that considers opportunistic defections in a dynamic setting may be more desirable. One drawback of the least core value as a mechanism for encouraging cooperation is that it imposes the same defection penalty for every coalition, regardless of its size or power. For situations in which this is not appropriate, the *f-least core* of a cooperative game (N, v) (Faigle et al. 2000) offers a way to address this issue: it is the set of cost allocations x that are optimal solutions to the linear program

$$\begin{aligned} z_f^* = \text{minimize} \quad & z_f \\ \text{subject to} \quad & x(N) = v(N), \end{aligned}$$

¹The linear program [LC] is clearly feasible. Adding the inequalities $x_i \leq v(\{i\}) + z$ for all $i \in N$ and using the equality $x(N) = v(N)$, we have that $z^* \geq \frac{1}{|N|}(v(N) - \sum_{i \in N} v(\{i\}))$. So as long as costs are finite, the optimal value of [LC] is finite and the least core value is well defined.

²This quantity is sometimes referred to as the *excess* of a coalition of agents in the cooperative game theory literature.

$$x(S) \leq v(S) + z_f f(S) \quad \text{for all } S \subseteq N, S \neq \emptyset, N$$

for some function $f : 2^N \rightarrow \mathbb{R}$. Several forms of the f -least core have been considered in the literature for various cooperative games, including $f(S) = |S|$ for all $S \subseteq N$ (Shapley and Shubik 1966), and $f(S) = v(S)$ for all $S \subseteq N$ (Faigle and Kern 1993).

Perhaps the most important and remarkable cooperative games are those with *submodular* costs³. A set function $v : 2^N \rightarrow \mathbb{R}$ is submodular if

$$v(S \cup \{j\}) - v(S) \geq v(S \cup \{j, k\}) - v(S \cup \{k\})$$

for all $j, k \in N$ such that $j \neq k$ and all $S \subseteq N \setminus \{j, k\}$. In words, submodularity captures the notion of decreasing marginal costs. Submodular cost cooperative games enjoy a wealth of interesting and desirable properties, including that the core of these games is always non-empty (Shapley 1971). Intuitively, cooperation amongst agents who face submodular costs is likely: as the size of a coalition grows, the marginal cost associated with adding a particular agent decreases, increasing the appeal of cooperation.

In this paper, we consider the opposite situation—when agents face *supermodular*, or increasing marginal costs. A set function $v : 2^N \rightarrow \mathbb{R}$ is supermodular if

$$v(S \cup \{j\}) - v(S) \leq v(S \cup \{j, k\}) - v(S \cup \{k\}) \tag{1.3}$$

for all $j, k \in N$ such that $j \neq k$ and all $S \subseteq N \setminus \{j, k\}$. In other words, v is supermodular if $-v$ is submodular. We study *supermodular cost cooperative games*: cooperative games (N, v) where v is nonnegative and supermodular. Supermodularity often naturally arises in situations in which the costs are closely tied to congestion effects. It has been shown that several variants of the facility location problem have supermodular costs (Nemhauser et al. 1978), and as we will show later, various problems from scheduling and network design also exhibit supermodular costs. Using the same reasoning as before, our intuition tells us that cooperation amongst rational agents who face supermodular costs is unlikely: since the marginal cost associated with adding a particular agent increases as the size of a coalition grows, diminishing the appeal of cooperation. It is straightforward to see that for supermodular cost cooperative games, the core is empty (as long as costs are not modular⁴).

Even though cooperation may not be desirable from the perspectives of the individual agents, as in supermodular cost cooperative games, encouraging or enforcing cooperation may still be desirable, especially to an external party. For instance, this can occur when the failure to cooperate gives rise to negative externalities. Consider the following example. A set of agents needs to process its jobs on a machine that generates an excessive amount of pollution. The agents have the opportunity to share the cost of processing their jobs on an existing single machine, but the cost of processing their jobs is such that it is cheaper for each agent to open their own machine, and as a result, generate more pollution. An authority may be interested in reducing such negative externalities. One approach would be to incorporate the cost of the pollution externalities directly into the processing costs; however, these externality costs may be hard to precisely define. Instead, one might ask, “How much do we need to charge for opening an additional machine in order to encourage all agents to share a single machine?” For a cooperative game where cooperation is not desirable from the individual agents’ standpoints—such as a supermodular cost cooperative game, in which the cost to a coalition is typically more than the sum of the individual agents’ costs due to congestion effects—the analogous question is, “How much do we need to penalize a coalition for acting independently in order to encourage all the agents to cooperate?” As mentioned earlier, this notion is captured in the least core value of a cooperative game.

³In the literature, the profit counterparts of these games—in which v is supermodular (see next paragraph) and represents the *profit* to coalitions—are known as *convex games*.

⁴A set function is modular if it is both submodular and supermodular.

In this paper, we study the computational complexity and approximation of the least core value and the least core of supermodular cost cooperative games. Initiated by Meggido (1978) and carried forward by Deng and Papadimitriou (1994), computational complexity has been proposed as another measure for evaluating different solution concepts in cooperative game theory. Examining the computational complexity of various solution concepts allows us to determine whether they are reasonable within the context of *bounded rationality* (see Simon 1972, for an extensive discussion)—the hypothesis that economic agents realistically have limited reasoning ability and computational power for decision-making. A solution concept with high computational complexity—one that is NP-hard to compute, for instance—may be considered unsatisfactory in a world with boundedly rational agents. We show that computing the least core value of supermodular cost cooperative games is NP-hard and therefore, this concept is not adequate in some sense; on the other hand, we propose alternatives by looking at approximations of the least core value and accompanying approximate least core cost allocations of supermodular cost cooperative games that can be computed in polynomial time.

The computational complexity of computing a cost allocation in the least core has been studied previously in several contexts. Faigle et al. (2000) showed that computing an element in the least core of minimum-cost spanning tree games is NP-hard. Kern and Paulusma (2003) presented a polynomial description of the linear program [LC] for cardinality matching games. Faigle et al. (2001) showed that by using the ellipsoid method, a so-called pre-kernel element in the least core of a cooperative game can be computed in polynomial time if the maximum dissatisfaction can be computed in polynomial time for any given efficient cost allocation. Properties of the least core value, on the other hand, seem to have been largely ignored. Deng (1998) observed that polynomial-time algorithms for submodular function minimization can be used to compute the least core and least core value of submodular cost cooperative games in polynomial time.

Organization of this work. In Section 2, we motivate the interest in supermodular cost cooperative games by providing a class of optimization problems whose optimal costs are supermodular. This class of optimization problems includes a variety of classical scheduling and network design problems. Then, in Section 3, we show that finding the least core value of supermodular cost cooperative games is NP-hard, and design approximation algorithms for computing the least core value of these games, using oracles that approximately determine maximally violated constraints. As a by-product, we also show how to compute accompanying approximate least core cost allocations.

In Sections 4 and 5, we apply our results to two special cases of supermodular cost cooperative games that arise from scheduling and matroid optimization. In Section 4, we study *scheduling games*, or cooperative games in which the costs are derived from the minimum sum of weighted completion times on a single machine. By improving on some of the results for general supermodular cost cooperative games, we are able to give an explicit formula for an element of the least core of scheduling games, and design a fully polynomial time approximation scheme for computing the least core value of these games. Finally, in Section 5, we consider a cooperative game with submodular profits: *matroid profit games*. Matroid profit games are cooperative games in which the profit to a coalition arises from the maximum weight of an independent set of a matroid. Some scheduling and network design problems have been shown to be special cases of finding a maximum weight independent set of a matroid. Using the framework established in Section 3 with the appropriate natural modifications, we show that the least core value and an element of the least core of these games can be computed in polynomial time.

2 A class of optimization problems with supermodular optimal costs

We begin by providing some motivation for looking at cooperative games with supermodular costs. The problem of minimizing a linear function over a *supermodular polyhedron*—a polyhedron of the form $\{x \in \mathbb{R}^N : x(S) \geq u(S) \text{ for all } S \subseteq N\}$, where $u : 2^N \rightarrow \mathbb{R}$ is supermodular—arises in many areas of

combinatorial optimization, especially in scheduling. For example, Wolsey (1985) and Queyranne (1993) showed that the convex hull of feasible completion time vectors on a single machine is a supermodular polyhedron. Queyranne and Schulz (1995) showed that the convex hull of feasible completion time vectors for unit jobs on parallel machines with nonstationary speeds is a supermodular polyhedron. The scheduling problem they considered includes various classical scheduling problems as special cases. Goemans et al. (2002) showed that for a scheduling environment consisting of a single machine and jobs with release dates, the convex hull of mean busy time vectors of preemptive schedules is a supermodular polyhedron.

In this section, we show that the optimal cost of minimizing a linear function over a supermodular polyhedron is a supermodular function. As a result, by studying supermodular cost cooperative games, we are able to gain insight into the sharing of optimal costs for a wide range of situations.

Theorem 2.1. *Let N be a finite set, and let $u : 2^N \rightarrow \mathbb{R}$ be a supermodular function. If $d_j \geq 0$ for all $j \in N$, then the function $v : 2^N \rightarrow \mathbb{R}$ defined by*

$$v(S) = \min \left\{ \sum_{j \in S} d_j x_j : x(A) \geq u(A) \text{ for all } A \subseteq S \right\} \quad \text{for all } S \subseteq N \quad (2.1)$$

is supermodular on N .

Proof. Let S be a subset of N with s elements. Without loss of generality, we assume that

$$S = \{1, \dots, j-1, j, j+1, \dots, k-1, k, k+1, \dots, s\},$$

and that the associated costs are nonincreasing: $d_1 \geq \dots \geq d_s$. Define $S^i = \{1, \dots, i\}$ for $i = 1, \dots, s$ and $S^0 = \emptyset$.

It is well known that minimizing a linear function over a supermodular polyhedron can be achieved by a greedy procedure (Edmonds 1970). In particular, the value of $v(S)$ is

$$\begin{aligned} v(S) &= \sum_{i=1}^s d_i (u(S^i) - u(S^{i-1})) \\ &= \sum_{i=1}^s d_i u(S^i) - \sum_{i=0}^{s-1} d_{i+1} u(S^i) \\ &= \sum_{i=1}^{s-1} (d_i - d_{i+1}) u(S^i) + d_s u(S^s) - d_1 u(S^0). \end{aligned}$$

Similarly, we have that

$$\begin{aligned} v(S \setminus \{j\}) &= \sum_{i=1}^{j-2} (d_i - d_{i+1}) u(S^i) + (d_{j-1} - d_{j+1}) u(S^{j-1}) \\ &\quad + \sum_{i=j+1}^{s-1} (d_i - d_{i+1}) u(S^i \setminus \{j\}) + d_s u(S^s \setminus \{j\}) - d_1 u(S^0), \\ v(S \setminus \{k\}) &= \sum_{i=1}^{k-2} (d_i - d_{i+1}) u(S^i) + (d_{k-1} - d_{k+1}) u(S^{k-1}) \\ &\quad + \sum_{i=k+1}^{s-1} (d_i - d_{i+1}) u(S^i \setminus \{k\}) + d_s u(S^s \setminus \{k\}) - d_1 u(S^0), \end{aligned}$$

$$\begin{aligned}
v(S \setminus \{j, k\}) &= \sum_{i=1}^{j-2} (d_i - d_{i+1})u(S^i) + (d_{j-1} - d_{j+1})u(S^{j-1}) \\
&\quad + \sum_{i=j+1}^{k-2} (d_i - d_{i+1})u(S^i \setminus \{j\}) + (d_{k-1} - d_{k+1})u(S^{k-1} \setminus \{j\}) \\
&\quad + \sum_{i=k+1}^{s-1} (d_i - d_{i+1})u(S^i \setminus \{j, k\}) + d_s u(S^s \setminus \{j, k\}) - d_1 u(S^0).
\end{aligned}$$

For any $l \in N$ and $A \subseteq N \setminus \{l\}$, we define $\Delta(A, l)$ to be the marginal value of adding l to A ; that is, $\Delta(A, l) = u(A \cup \{l\}) - u(A)$. Therefore,

$$\begin{aligned}
v(S \setminus \{j\}) - v(S \setminus \{j, k\}) &= (d_{k-1} - d_k)u(S^{k-1} \setminus \{j\}) + (d_k - d_{k+1})u(S^k \setminus \{j\}) - (d_{k-1} - d_{k+1})u(S^{k-1} \setminus \{j\}) \\
&\quad + \sum_{i=k+1}^{s-1} (d_i - d_{i+1})(u(S^i \setminus \{j\}) - u(S^i \setminus \{j, k\})) + d_s (u(S^s \setminus \{j\}) - u(S^s \setminus \{j, k\})) \\
&= (d_k - d_{k+1})\Delta(S^{k-1} \setminus \{j\}, k) + \sum_{i=k+1}^s (d_i - d_{i+1})\Delta(S^i \setminus \{j, k\}, k) + d_s \Delta(S^s \setminus \{j, k\}, k).
\end{aligned}$$

Similar to above, we consider the effects of adding k to $S \setminus \{k\}$:

$$\begin{aligned}
v(S) - v(S \setminus \{k\}) &= (d_{k-1} - d_k)u(S^{k-1}) + (d_k - d_{k+1})u(S^k) - (d_{k-1} - d_{k+1})u(S^{k-1}) \\
&\quad + \sum_{i=k+1}^{s-1} (d_i - d_{i+1})(u(S^i) - u(S^i \setminus \{k\})) + d_s (u(S^s) - u(S^s \setminus \{k\})) \\
&= (d_k - d_{k+1})\Delta(S^{k-1}, k) + \sum_{i=k+1}^{s-1} (d_i - d_{i+1})\Delta(S^i \setminus \{k\}, k) + d_s \Delta(S^s \setminus \{k\}, k).
\end{aligned}$$

By the supermodularity of u , we have that $\Delta(A, k) \leq \Delta(B, k)$ for any $A \subseteq B \subseteq N \setminus \{k\}$. This, in addition with the fact that $d_i - d_{i+1} \geq 0$ for all $i = 1, \dots, s-1$ and $d_s \geq 0$, implies that

$$\begin{aligned}
v(S) - v(S \setminus \{k\}) &= (d_k - d_{k+1})\Delta(S^{k-1}, k) + \sum_{i=k+1}^{s-1} (d_i - d_{i+1})\Delta(S^i \setminus \{k\}, k) + d_s \Delta(S^s \setminus \{k\}, k) \\
&\geq (d_k - d_{k+1})\Delta(S^{k-1} \setminus \{j\}, k) + \sum_{i=k+1}^s (d_i - d_{i+1})\Delta(S^i \setminus \{j, k\}, k) + d_s \Delta(S^s \setminus \{j, k\}, k) \\
&= v(S \setminus \{j\}) - v(S \setminus \{j, k\}).
\end{aligned}$$

Therefore, v is supermodular. □

As mentioned above, by the work of Wolsey (1985), Queyranne (1993), Queyranne and Schulz (1995), and Goemans et al. (2002), we immediately have the following corollary of Theorem 2.1.

Corollary 2.2. *If for all $S \subseteq N$, $v(S)$ is the objective value of optimally scheduling jobs in S for the problem⁵ (a) $1 \mid \mid \sum w_j C_j$, (b) $Q \mid p_j = 1 \mid \sum w_j C_j$, (c) $P \mid p_j = 1, r_j \text{ integral} \mid \sum w_j C_j$, (d) $P \mid \mid \sum C_j$, or (e) $1 \mid r_j, \text{pmtn} \mid \sum w_j M_j$, then v is supermodular.*

Unfortunately, Corollary 2.2(d) does not extend to the case with arbitrary weights and processing times. In addition, one can show that the scheduling problems $1 \mid r_j \mid \sum C_j$ and $1 \mid \text{prec} \mid \sum C_j$ do not have supermodular optimal costs.

Using almost identical techniques to those in the proof of Theorem 2.1, we can also show that maximizing a nonnegative linear function over a *submodular polyhedron*—a polyhedron of the form $\{x \in \mathbb{R}^N : x(S) \leq u(S) \text{ for all } S \subseteq N\}$ where $u : 2^N \rightarrow \mathbb{R}$ is submodular—has submodular optimal values.

Theorem 2.3. *Let N be a finite set, and let $u : 2^N \rightarrow \mathbb{R}$ be a submodular function. If $d_j \geq 0$ for all $j \in N$, then the function $v : 2^N \rightarrow \mathbb{R}$ defined by*

$$v(S) = \max \left\{ \sum_{j \in S} d_j x_j : x(A) \leq u(A) \text{ for all } A \subseteq S \right\} \quad \text{for all } S \subseteq N$$

is submodular on N .

An important example of maximizing a nonnegative linear function over a submodular polyhedron is finding a maximum weight independent set of a matroid; in fact, a version of Theorem 2.3 has been mentioned in the literature for this special case (see Nemhauser and Wolsey 1988, page 715). One example of finding a maximum weight independent set of a matroid is finding a maximum weight forest in an undirected graph (Birkhoff 1935; Whitney 1935). Later, in Section 5, we study a cooperative game in which the profit to a coalition arises from the maximum weight of an independent set of a matroid.

3 Complexity and approximation

We now turn our attention to the computational complexity and approximability of computing the least core value of an arbitrary supermodular cost cooperative game (N, v) . Note that an arbitrary supermodular function v may not be compactly encoded. Therefore, for the remainder of this section we assume that we have a value-giving oracle for v . In addition, for the remainder of the paper, we assume that there are at least two agents ($n \geq 2$).

3.1 Computational complexity

Theorem 3.1. *Computing the least core value of supermodular cost cooperative games is strongly NP-hard, even if an element of the least core is known.*

Proof. We show that any instance of the strongly NP-hard maximum cut problem on an undirected graph (Garey et al. 1976) can be reduced to an instance of computing the least core value of a supermodular cost cooperative game. Consider an arbitrary undirected graph $G = (N, E)$. Let $\kappa : 2^N \rightarrow \mathbb{R}$ be the *cut function* of G ; that is,

$$\kappa(S) = \left| \left\{ \{i, j\} \in E : i \in S, j \in N \setminus S \right\} \right|.$$

⁵We describe these problems using the notation of Graham et al. (1979), in which the features of a scheduling problem are captured in the three-field abbreviation $\alpha \mid \beta \mid \gamma$. The field α represents the machine environment: for example, “1” refers to a single machine, “P” refers to identical parallel machines, and “Q” refers to uniform parallel machines. The field β describes job characteristics: for instance, “ $p_j = 1$ ” indicates that all jobs have unit processing time, “ r_j ” indicates that jobs have release dates, and “pmtn” indicates that preemption of jobs is allowed. Finally, the field γ denotes the objective function to be minimized: for example, “ $\sum w_j C_j$ ” refers to the sum of weighted completion times objective, and “ $\sum w_j M_j$ ” refers to the sum of weighted mean busy times objective.

Also, let the function $\eta : 2^N \rightarrow \mathbb{R}$ be defined as

$$\eta(S) = \left| \{ \{i, j\} \in E : i \in S, j \in S \} \right|.$$

Clearly, η is nonnegative. Using the increasing marginal cost characterization of supermodularity (1.3), it is straightforward to see that η is supermodular. Using counting arguments, it is also straightforward to show that

$$\eta(S) + \eta(N \setminus S) + \kappa(S) = \eta(N)$$

for any $S \subseteq N$.

Now consider the supermodular cost cooperative game (N, v) , where $v(S) = 2\eta(S)$ for all $S \subseteq N$. For each player $i \in N$, we define the cost allocation $x_i = \deg(i)$, where $\deg(i)$ denotes the degree of node i in G . In addition, let $z = \max_{S \subseteq N, S \neq \emptyset, N} \kappa(S)$. Note that $x(N) = \sum_{i \in N} \deg(i) = v(N)$, and for all $S \subseteq N$, $S \neq \emptyset, N$,

$$z \geq \kappa(S) = (2\eta(S) + \kappa(S)) - 2\eta(S) = x(S) - v(S).$$

Therefore, (x, z) is a feasible solution to [LC]. Now suppose (x^*, z^*) is an optimal solution to [LC]. Adding the inequalities $x^*(S) \leq v(S) + z^*$ and $x^*(N \setminus S) \leq v(N \setminus S) + z^*$ for any $S \subseteq N$, $S \neq \emptyset, N$, and using the equality $x^*(N) = v(N)$, we have that

$$2z^* \geq v(N) - v(S) - v(N \setminus S) = 2\kappa(S) \quad \text{for all } S \subseteq N, S \neq \emptyset, N.$$

Therefore, $z^* \geq z$. It follows that $z^* = z = \max_{S \subseteq N, S \neq \emptyset, N} \kappa(S)$, and x is an element of the least core of (N, v) . In other words, finding the least core value of (N, v) is equivalent to finding the value of a maximum cut in $G = (N, E)$. \square

In our proof of the above theorem, we show that for any instance of the maximum cut problem on an undirected graph, there exists a supermodular cost cooperative game whose least core value is exactly equal to the value of the maximum cut. Since the maximum cut problem is not approximable within a factor of 1.0624 (Håstad 2001), we immediately obtain the following inapproximability result:

Corollary 3.2. *There is no ρ -approximation algorithm⁶ for computing the least core value of supermodular cost cooperative games, where $\rho < 1.0624$, unless $P = NP$.*

3.2 Approximation by fixing a cost allocation

The above negative results indicate that it is rather unlikely that we will be able to compute the least core value of supermodular cost cooperative games exactly in polynomial time, even if an element of the least core is known. This motivates us to design methods with polynomial running time that approximate the least core value of these games.

As a first attempt at approximation, we fix a cost allocation x such that $x(N) = v(N)$, and then try to determine the minimum value of z such that (x, z) is feasible in the least core linear program [LC]. Recall that the dissatisfaction of a subset of agents S under a cost allocation x is defined as $e(x, S) = x(S) - v(S)$. For any cooperative game (N, v) , we define the following problem:

x -maximum dissatisfaction problem for cooperative game (N, v) (x -MD). *Given cost allocation x such that $x(N) = v(N)$, find a subset of agents S^* whose dissatisfaction is maximum:*

$$e(x, S^*) = \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} e(x, S) = \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \{x(S) - v(S)\}.$$

⁶A ρ -approximation algorithm ($\rho \geq 1$) is an algorithm that always finds a solution whose objective value is within a factor ρ of the optimal value, and whose running time is polynomial in the input size.

We want to find a value z that is as close to $e(x, S^*)$ as possible, but *larger* than $e(x, S^*)$, since (x, z) is feasible if and only if $z \geq e(x, S^*)$. Note that an algorithm for the x -MD problem acts as a separation oracle for the vector (x, z) to the linear program [LC]: if $z \geq e(x, S^*)$, then (x, z) is feasible in [LC]; otherwise, we have $z < e(x, S^*)$, which implies that $x(S^*) \leq v(S^*) + z$ is a constraint violated by (x, z) .

How should we fix x ? We would like to ensure that the cost allocation x we choose is at least in the vicinity of the least core of (N, v) , so that we do not prematurely weaken the resulting approximation to the least core value. Suppose z^* is the least core value of (N, v) . For any $\rho \geq 1$, we define the ρ -approximate least core of (N, v) as the set of all cost allocations x such that

$$\begin{aligned} x(N) &= v(N), \\ x(S) &\leq v(S) + \rho z^* \quad \text{for all } S \subseteq N, S \neq \emptyset, N, \end{aligned}$$

or equivalently,

$$x(N) = v(N), \quad \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} e(x, S) \leq \rho z^*.$$

A cost allocation in the least core of (N, v) is the “least objectionable” in the sense that it is an efficient cost allocation that minimizes the maximum dissatisfaction of any coalition. A cost allocation in the ρ -approximate least core of (N, v) approximates being “least objectionable” by ensuring that the maximum dissatisfaction of a coalition is at most a factor ρ away from the least core value z^* —the minimum possible maximum dissatisfaction of a coalition under any efficient cost allocation.

For any set function $v : 2^N \rightarrow \mathbb{R}$, we define the polytope

$$B_v = \{x \in \mathbb{R}^N : x(N) = v(N), x(S) \geq v(S) \text{ for all } S \subseteq N\}.$$

For an arbitrary set function v , computing an element of B_v may require an exponential number of oracle calls, or B_v may be empty. Fortunately, when v is supermodular, the vertices of B_v are computable in polynomial time, and even have explicit formulas (Edmonds 1970). It turns out that any cost allocation x in B_v is an element of the 2-approximate least core of (N, v) .

Theorem 3.3. *Suppose (N, v) is a supermodular cost cooperative game, and x is a cost allocation in B_v . Let $e(x, S^*)$ be the optimal value of the x -maximum dissatisfaction problem for (N, v) , and let z^* be the least core value of (N, v) . Then, x is an element of the 2-approximate least core of (N, v) , or equivalently, $e(x, S^*) \leq 2z^*$.*

Proof. Let (x^*, z^*) be an optimal solution to [LC]. As in the proof of Theorem 3.1, we have that

$$2z^* \geq v(N) - v(S) - v(N \setminus S) \quad \text{for all } S \subseteq N, S \neq \emptyset, N.$$

Since $x \in B_v$, we can deduce that for any $S \subseteq N, S \neq \emptyset, N$,

$$2z^* \geq v(N) - v(S) - v(N \setminus S) = x(S) - v(S) + x(N \setminus S) - v(N \setminus S) \geq e(x, S).$$

Since the above lower bound on $2z^*$ holds for any $S \subseteq N, S \neq \emptyset, N$, it follows that $2z^* \geq e(x, S^*)$. \square

We use this observation, in conjunction with a ρ -approximation algorithm for the x -maximum dissatisfaction problem for (N, v) , to approximate the least core value of (N, v) .

Theorem 3.4. *Suppose (N, v) is a supermodular cost cooperative game, and x is a cost allocation in B_v . If there exists a ρ -approximation algorithm for the x -maximum dissatisfaction problem for (N, v) , then there exists a 2ρ -approximation algorithm for computing the least core value of (N, v) .*

Proof. Let \bar{S} be the output from a ρ -approximation algorithm for the x -maximum dissatisfaction problem for (N, v) , and let $z = \rho e(x, \bar{S})$. We show that (x, z) is a feasible solution to the linear program [LC], and that z is within a factor of 2ρ of z^* , the least core value of (N, v) . Since $x \in B_v$, we have that $x(N) = v(N)$. Since \bar{S} is output from a ρ -approximation algorithm for the x -maximum dissatisfaction problem for (N, v) , it follows that $z = \rho e(x, \bar{S}) \geq e(x, S^*) \geq x(S) - v(S)$ for all $S \subseteq N$, $S \neq \emptyset, N$. So (x, z) is a feasible solution to [LC]. By Theorem 3.3, it follows that $z = \rho e(x, \bar{S}) \leq \rho e(x, S^*) \leq 2\rho z^*$. \square

Note that the x -maximum dissatisfaction problem for a supermodular cost cooperative game is an instance of submodular function maximization. In addition, for any $x \in B_v$, the objective function $e(x, \cdot)$ of the x -maximum dissatisfaction problem is nonnegative. Feige et al. (2007) gave a $5/2$ -approximation algorithm for maximizing nonnegative submodular functions. With Theorem 3.4, this immediately implies the following corollary.

Corollary 3.5. *Suppose (N, v) is a supermodular cost cooperative game. Then, there exists a 5 -approximation algorithm for computing the least core value (N, v) .*

3.3 Approximation without fixing a cost allocation

Until now, we have considered approximating the least core value of a supermodular cost cooperative game (N, v) by fixing a cost allocation x and then finding z such that (x, z) is feasible in the least core linear program [LC]. Suppose that, instead of fixing a cost allocation in advance, we compute a cost allocation along with an approximation to the least core value. Let us assume that we have a ρ -approximation algorithm for the x -maximum dissatisfaction problem for (N, v) , for every x such that $x(N) = v(N)$.⁷ By using the ellipsoid method with binary search, we can establish one of the main results of this work:

Theorem 3.6. *Suppose (N, v) is a supermodular cost cooperative game, and there exists a ρ -approximation algorithm for the x -maximum dissatisfaction problem for (N, v) , for every cost allocation x such that $x(N) = v(N)$. Let z^* be the least core value of (N, v) . Then,*

- (a) *there exists a ρ -approximation algorithm for computing the least core value of (N, v) , and*
- (b) *there exists a polynomial-time algorithm for computing a cost allocation in the ρ -approximate least core of (N, v) .*

Proof sketch. For a complete proof, see Appendix A.1.

The idea behind the proof is as follows. Suppose that K is a polytope. The exact separation problem for K is:

Exact separation problem for K . *Given $y \in \mathbb{Q}^n$, either (i) assert $y \in K$, or (ii) find a hyperplane that separates y from K : find $c \in \mathbb{Q}^n$ such that $c^\top y > c^\top x$ for all $x \in K$.*

The exact non-emptiness problem for K is:

Exact non-emptiness problem for K . *Either (i) find a vector $y \in K$ or (ii) assert K is empty.*

Grötschel et al. (1988) showed that for a polytope K , by using the ellipsoid method, a polynomial-time algorithm for any one of the following three problems—the exact separation problem for K , the exact

⁷Note that since v is supermodular and $v(\emptyset) = 0$, for any x such that $x(N) = v(N)$, we have that $\sum_{i \in N} (x_i - v(\{i\})) \geq \sum_{i \in N} x_i - v(N) = 0$. Therefore, there must exist $i \in N$ such that $x_i - v(\{i\}) \geq 0$, and so $\max_{S \subseteq N, S \neq \emptyset, N} e(x, S) \geq 0$. This ensures that the notion of a ρ -approximation algorithm for the x -maximum dissatisfaction problem is sensible, for any given cost allocation x such that $x(N) = v(N)$.

non-emptiness problem for K , or optimizing a linear function over K —implies a polynomial-time algorithm for the other two problems.

Now suppose \bar{K} is a polytope that “approximates” K . Note that this “approximation” can be any arbitrary polytope. The approximate separation problem for K and its approximation \bar{K} is:

Approximate separation problem for K and its approximation \bar{K} . Given $y \in \mathbb{Q}^n$, either (i) assert $y \in \bar{K}$, or (ii) find a hyperplane that separates y from K : find $c \in \mathbb{Q}^n$ such that $c^\top y > c^\top x$ for all $x \in K$.

The approximate non-emptiness problem for K and its approximation \bar{K} is:

Approximate non-emptiness problem for K and its approximation \bar{K} . Either (i) find a vector $y \in \bar{K}$ or (ii) assert K is empty.

Using similar techniques to those used in Grötschel et al. (1988), we can show that the ellipsoid method can be used with a polynomial-time algorithm for the approximate separation problem to solve the approximate non-emptiness problem in polynomial time.

We use this general result for the least core value problem. Fix a supermodular cost cooperative game (N, v) , and let z^* be its least core value. For any fixed $\gamma \geq 0$, define the polytope

$$Q_\gamma = \{x \in \mathbb{R}^N : x(N) = v(N), x(S) \leq v(S) + \gamma \text{ for all } S \subseteq N, S \neq \emptyset, N\}.$$

It is straightforward to show that a ρ -approximation algorithm for the x -maximum dissatisfaction problem for (N, v) can be used as a polynomial-time algorithm for the approximate separation problem for Q_γ and its approximation $Q_{\rho\gamma}$. Therefore, there exists a polynomial-time algorithm for the approximate non-emptiness problem for Q_γ and its approximation $Q_{\rho\gamma}$. Suppose that \mathcal{A} is such an algorithm. Since v is nonnegative, supermodular, and $v(\emptyset) = 0$, the least core value z^* of (N, v) is in the interval $[0, v(N)]$. Since the polytope Q_γ is nonempty for all $\gamma \geq z^*$, and the polytope Q_γ is empty for all $\gamma < z^*$, using \mathcal{A} with binary search on $[0, v(N)]$ to find $\bar{\gamma}$ such that $Q_{\bar{\gamma}-\epsilon}$ is empty but $Q_{\rho\bar{\gamma}}$ is non-empty gives us an algorithm for computing a ρ -approximation to the least core value of (N, v) and a cost allocation in the ρ -approximate least core (the parameter $\epsilon \in \mathbb{Q}_{>0}$ is the precision required, and $\log \epsilon^{-1}$ is polynomial in n and $\log v(N)$). The number of calls to \mathcal{A} that is needed for this binary search is polynomial in n and $\log v(N)$. \square

Using an approximate separation oracle in conjunction with the ellipsoid method to achieve approximate optimization has been studied previously for a variety of problems (e.g. Jansen 2003; Jain et al. 2003; Fleischer et al. 2006). It appears at first that this technique can be used for any arbitrary linear program with an exponential number of constraints; however, the proofs for these results all depend on the structure of the problem. For example, Jansen (2003) considered polytopes of the form $K = \{x \in \mathbb{R}^n : Ax \leq b\} \cap B$, where $n, m \in \mathbb{Z}_{>0}$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}_{>0}^m$, and B is an arbitrary polytope, and approximations $\bar{K} = \{x \in \mathbb{R}^n : Ax \leq \alpha b\} \cap B$ for some $\alpha \geq 1$. In addition, it is assumed that the zero vector is in the polytope K . With this assumption, simply multiplying a feasible point in \bar{K} by the scalar $1/\alpha$ yields a feasible point in K ; without this assumption, it is not clear how to convert a feasible point in \bar{K} to a feasible point in K . In the proof of Theorem 3.6, this kind of conversion is unnecessary since we are able to exploit the structure of the constraints and their relation to the objective function in the linear program [LC]: for any vector x in the approximation $Q_{\rho\bar{\gamma}}$, the vector $(x, \rho\bar{\gamma})$ is a feasible solution to the linear program [LC].

With Theorem 3.6 in hand, it remains to show how to solve the x -maximum dissatisfaction problem for a supermodular cost cooperative game (N, v) , for every cost allocation x such that $x(N) = v(N)$. As we

noted in Section 3.2, the x -maximum dissatisfaction problem for a supermodular cost cooperative game (N, v) is an instance of submodular function maximization. Unlike in Section 3.2, however, the objective functions for the instances of the x -maximum dissatisfaction problem that need to be solved for Theorem 3.6 are not necessarily nonnegative. Feige et al. (2007) designed a local-search based approximation algorithm for maximizing a submodular function $f : 2^N \rightarrow \mathbb{R}$ with $f(\emptyset) \geq 0$ and $f(N) \geq 0$, that has a performance guarantee of $(3 + \epsilon)$ for any $\epsilon > 0$. Since $e(x, \emptyset) = e(x, N) = 0$ for any cost allocation x such that $x(N) = v(N)$, we obtain the following corollary.

Corollary 3.7. *Suppose (N, v) is a supermodular cost cooperative game. Then for any $\epsilon > 0$,*

- (a) *there exists a $(3 + \epsilon)$ -approximation algorithm for computing the least core value of (N, v) , and*
- (b) *there exists a polynomial-time algorithm for computing a cost allocation in the $(3 + \epsilon)$ -approximate least core of (N, v) .*

By computing a cost allocation on the fly using the ellipsoid method, we are able to design an approximation algorithm for computing the least core value of a supermodular cost cooperative game with a worst-case performance guarantee of $(3 + \epsilon)$, which compares favorably to the worst-case performance guarantee of 5 for the fixed-cost-allocation-based approximation algorithm designed in Section 3.2. Interestingly, however, the comparison for the accompanying cost allocations of these approximation algorithms is reversed: the cost allocation that is computed by the ellipsoid-method-based approximation algorithm is guaranteed to be in the $(3 + \epsilon)$ -approximate least core, while the cost allocation used in the fixed-cost-allocation-based approximation algorithm is guaranteed to be in the 2-approximate least core.

4 A special case from single-machine scheduling

In this section, we study a particular supermodular cost cooperative game that arises from scheduling situations. Consider a setting where agents each have a job that needs to be processed on a machine, and any coalition of agents can potentially open their own machine. Suppose each agent $i \in N$ has a job whose processing time is $p_i \in \mathbb{R}_{>0}$ and weight is $w_i \in \mathbb{R}_{\geq 0}$. Jobs are independent, and are scheduled non-preemptively on a single machine, which can process at most one job at a time. A *scheduling game* is a cooperative game (N, v) where the cost $v(S)$ to a coalition S is the minimum sum of weighted completion times of jobs in S . If weight w_i is interpreted as agent i 's per-unit-time waiting cost, then $v(S)$ can be seen as the minimum total waiting cost for agents in S . The least core value of scheduling games has a natural interpretation: it is the minimum amount we need to charge any coalition for opening a new machine in order to encourage cooperation.

Various cooperative games that arise from scheduling situations have been studied previously. In sequencing games (e.g. Curiel et al. 1989), agents—each with a job that needs to be processed—start with a feasible solution on a fixed number of machines, and the profit assigned to a coalition of agents is the maximal cost savings the coalition can achieve by rearranging themselves. Scheduling games have received somewhat limited attention in the past; several authors have developed axiomatic characterizations of various cost sharing rules for these games (Maniquet 2003; Mishra and Rangarajan 2005).

From Corollary 2.2, it follows that scheduling games are indeed supermodular cost cooperative games, and the results from Section 3 apply. We will apply the results of Section 3.2, in which approximation is based on fixing a cost allocation, to finding the least core value of scheduling games. Before doing so, however, we establish some useful and interesting properties of the least core of scheduling games. These properties will help us determine the computational complexity of this special case; they will also help us choose a specific cost allocation \bar{x} in B_v with especially nice features. In particular, we will be able to design approximation algorithms for the \bar{x} -maximum dissatisfaction problem for scheduling games, as well as show a stronger translation between the approximability of the \bar{x} -maximum dissatisfaction problem and the approximability of the least core value of these games.

4.1 Key properties of the least core of scheduling games

The structure of the cost function for scheduling games allows us to explicitly express an element of the least core of scheduling games and recast the least core linear program as the maximization of a set function defined solely in terms of the cost function v .

Smith (1956) showed that scheduling jobs in nonincreasing order of w_j/p_j minimizes the sum of weighted completion times on a single machine. To simplify the analysis, for the remainder of this paper we assume without loss of generality that

$$\frac{w_1}{p_1} \geq \dots \geq \frac{w_n}{p_n}.$$

We consider the cost allocation \bar{x} defined as follows:

$$\bar{x}_i = \frac{1}{2}(v(S^i) - v(S^{i-1})) + \frac{1}{2}(v(N \setminus S^{i-1}) - v(N \setminus S^i)) \quad (4.1)$$

$$= \frac{1}{2}w_i \sum_{j=1}^i p_j + \frac{1}{2}p_i \sum_{j=i}^n w_j \quad (4.2)$$

for $i = 1, \dots, n$, where $S^i = \{1, \dots, i\}$ and $S^0 = \emptyset$. It is straightforward to show that $\bar{x} \in B_v$; in fact, it is a convex combination of two vertices of B_v . Since $\bar{x} \in B_v$, we have that $\bar{x}(S) \geq v(S)$ for all $S \subseteq N$. As it turns out, for scheduling games, we are able to show a more precise relationship between the cost allocation $\bar{x}(S)$ of a coalition S and its cost $v(S)$.

Lemma 4.1. *Suppose (N, v) is a scheduling game. Then, the cost allocation \bar{x} as defined in (4.2) satisfies*

$$\bar{x}(S) - v(S) = \frac{1}{2}(v(N) - v(S) - v(N \setminus S))$$

for all $S \subseteq N$.

Proof. Since jobs are assumed to be indexed according to nonincreasing weight-to-processing time ratio, by Smith's rule we know that for any $S \subseteq N$, $v(S) = \sum_{i \in S} \sum_{j \in S}^i w_i p_j$. Therefore,

$$\begin{aligned} 2(\bar{x}(S) - v(S)) &= \sum_{i \in S} \sum_{j=1}^i w_i p_j + \sum_{i \in S} \sum_{j=i}^n p_i w_j - 2 \sum_{i \in S} \sum_{\substack{j=1 \\ j \in S}}^i w_i p_j \\ &= \sum_{i \in S} \sum_{j=1}^i w_i p_j + \sum_{i \in S} \sum_{j=i}^n p_i w_j - \sum_{i \in S} \sum_{\substack{j=1 \\ j \in S}}^i w_i p_j - \sum_{i \in S} \sum_{\substack{j=i \\ j \in S}}^n p_i w_j \\ &= \sum_{i \in S} \sum_{\substack{j=1 \\ j \in N \setminus S}}^i w_i p_j + \sum_{i \in S} \sum_{\substack{j=i \\ j \in N \setminus S}}^n p_i w_j \\ &= \sum_{i \in S} \sum_{\substack{j=1 \\ j \in N \setminus S}}^i w_i p_j + \sum_{i \in N \setminus S} \sum_{\substack{j=1 \\ j \in S}}^i w_i p_j \\ &= \sum_{i \in N} \sum_{j=1}^i w_i p_j - \sum_{i \in S} \sum_{j=1}^i w_i p_j - \sum_{i \in N \setminus S} \sum_{j=1}^i w_i p_j \end{aligned}$$

$$\begin{aligned}
& + \sum_{i \in S} \sum_{\substack{j=1 \\ j \in N \setminus S}}^i w_i p_j + \sum_{i \in N \setminus S} \sum_{\substack{j=1 \\ j \in S}}^i w_i p_j \\
& = \sum_{i \in N} \sum_{j=1}^i w_i p_j - \sum_{i \in S} \sum_{\substack{j=1 \\ j \in S}}^i w_i p_j - \sum_{i \in N \setminus S} \sum_{\substack{j=1 \\ j \in N \setminus S}}^i w_i p_j \\
& = v(N) - v(S) - v(N \setminus S). \quad \square
\end{aligned}$$

With this lemma in hand, we can show the following key properties of the least core of scheduling games.

Theorem 4.2. *Suppose (N, v) is a scheduling game.*

- (a) *The cost allocation \bar{x} as defined in (4.2) is an element of the least core of (N, v) .*
- (b) *The least core value of (N, v) is*

$$z^* = \frac{1}{2} \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \{v(N) - v(S) - v(N \setminus S)\}. \quad (4.3)$$

Proof. Let \bar{z} be the value of the right-hand side of (4.3). First, we show that (\bar{x}, \bar{z}) is a feasible solution to [LC]. By Lemma 4.1, we have that $\bar{x}(N) = v(N)$, and for any $S \subseteq N$, $S \neq \emptyset, N$,

$$\bar{z} \geq \frac{1}{2}(v(N) - v(S) - v(N \setminus S)) = \bar{x}(S) - v(S).$$

Now suppose (x^*, z^*) is an optimal solution to [LC]. As in the proof of Theorem 3.1, we obtain the following lower bound on $2z^*$:

$$2z^* \geq v(N) - v(S) - v(N \setminus S) \quad \text{for all } S \subseteq N, S \neq \emptyset, N.$$

Therefore, $z^* \geq \bar{z}$. It follows that \bar{x} is an element of the least core of (N, v) , and the least core value of (N, v) is \bar{z} . \square

In addition to being an element of the least core, it happens that the cost allocation \bar{x} as defined in (4.2) is the Shapley value⁸ of scheduling games (Mishra and Rangarajan 2005). This is quite special: for a supermodular cost cooperative game (N, v) , the Shapley value is not necessarily an element of the least core of (N, v) . Example 4.3 illustrates this point.

One might wonder if the cost allocation \bar{x} as defined in (4.1) is an element of the least core for general supermodular cost cooperative games. Note that the definition of \bar{x} in (4.1) depends on the ordering of N . For a given permutation $\sigma : N \rightarrow N$ where $\sigma(i)$ denotes the position of player $i \in N$, we define the cost allocation \bar{x}^σ as follows:

$$\bar{x}_{\sigma^{-1}(i)}^\sigma = \frac{1}{2}(v(S^i) - v(S^{i-1})) + \frac{1}{2}(v(N \setminus S^{i-1}) - v(N \setminus S^i))$$

for $i = 1, \dots, n$, where $S^i = \{\sigma^{-1}(1), \dots, \sigma^{-1}(i)\}$, and $S^0 = \emptyset$. The cooperative game (N, v) defined in Example 4.3 is an instance of a supermodular cost cooperative game (in particular, v is of the form (2.1)) for

⁸The Shapley value (Shapley 1953) of a cooperative game (N, v) is the cost allocation ϕ , where

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!} (v(S \cup \{i\}) - v(S)) \quad \text{for all agents } i \in N.$$

In words, the Shapley value of each agent i reflects agent i 's average marginal contribution to the coalition N . The Shapley value is one of the most important solution concepts in cooperative game theory; for example, see Roth (1988).

which the cost allocation \bar{x}^σ is not a least core element of (N, v) , for all permutations σ of N . We can also show that when (N, v) is a scheduling game, not every cost allocation in B_v is necessarily an element of the least core of (N, v) .

Example 4.3. Consider the cooperative game (N, v) defined as follows. There are four players: $N = \{1, 2, 3, 4\}$. Each agent $i \in N$ has a processing time $p_i = i$. The cost $v(S)$ to a coalition S is the optimal value of the scheduling problem $P2 \parallel \sum C_j$, restricted to jobs in S . By Corollary 2.2, v is supermodular. The Shapley value of this game is $\phi_1 = 3/2, \phi_2 = 17/6, \phi_3 = 23/6$, and $\phi_4 = 29/6$, and the optimal value of the ϕ -maximum dissatisfaction problem for this game is $\max_{S \subseteq N, S \neq \emptyset, N} e(\phi, S) = 5/3$. However, the least core value of this game is $3/2$. It is also straightforward to check that $\max_{S \subseteq N, S \neq \emptyset, N} e(\bar{x}^\sigma, S) = 2$ for all permutations σ of N .

4.2 Computational complexity

Although computing the least core value of supermodular cost cooperative games is strongly NP-hard, it is still unclear if this remains the case for scheduling games. In the previous subsection, we showed that we can efficiently compute an element of the least core of scheduling games. In fact, we have an explicit formula for a least core element. Computing the least core value of scheduling games, however, remains NP-hard.

Theorem 4.4. *Computing the least core value of scheduling games is NP-hard, even when $w_j = p_j$ for all $j \in N$.*

Proof. By Theorem 4.2, the least core value of scheduling games is

$$z^* = \frac{1}{2} \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \{v(N) - v(S) - v(N \setminus S)\} = \frac{1}{2}v(N) - \frac{1}{2} \min_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \{v(S) + v(N \setminus S)\}.$$

Note that the minimization problem above is equivalent to the problem of minimizing the sum of weighted completion times of jobs in N , with weight w_j and processing time p_j for each job $j \in N$, on two identical parallel machines. Bruno et al. (1974) showed that this two-machine problem is NP-hard, even when $w_j = p_j$ for all jobs $j \in N$. \square

The above result is in stark contrast to the underlying problem defining the costs in scheduling games—minimizing the sum of weighted completion times on a single machine—for which *any* order is optimal when each job has its weight equal to its processing time.

4.3 Tighter bounds on approximation based on fixing a cost allocation

In Section 3.2, we showed that for any supermodular cost cooperative game (N, v) and a cost allocation x in B_v , a ρ -approximation algorithm for the x -maximum dissatisfaction problem implies a 2ρ -approximation algorithm for computing the least core value of (N, v) . It is reasonable to believe, though, that for scheduling games, we may be in a position to do better, since the cost allocation \bar{x} as defined in (4.2) is in B_v , and is in fact an element of the least core. This is indeed the case: from Lemma 4.1 and Theorem 4.2, it follows that

$$z^* = \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \{\bar{x}(S) - v(S)\} = \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} e(\bar{x}, S).$$

This is exactly the \bar{x} -maximum dissatisfaction problem for scheduling games! Therefore, we obtain the following strengthening of Theorem 3.4.

Theorem 4.5. *Suppose there exists a ρ -approximation algorithm for the \bar{x} -maximum dissatisfaction problem for scheduling games, where the cost allocation \bar{x} is as defined in (4.2). Then there exists a ρ -approximation algorithm for computing the least core value of scheduling games.*

Some of the results from the previous subsections give us insight into how to design oracles that approximately solve \bar{x} -MD for scheduling games. In particular, by Lemma 4.1, we know that \bar{x} -MD is in fact

$$\max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} e(\bar{x}, S) = \frac{1}{2} \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \{v(N) - v(S) - v(N \setminus S)\}.$$

As mentioned in the proof of Theorem 4.4, maximizing $e(\bar{x}, S)$ is equivalent to minimizing the sum of weighted completion times of jobs in N on two identical parallel machines. This two-machine problem is NP-complete (Bruno et al. 1974), and has a fully polynomial time approximation scheme⁹ (FPTAS) (Sahni 1976). Although the two problems are equivalent from the optimization perspective, as is often the case with equivalent minimization and maximization problems, it is not immediately obvious if they are equivalent in terms of approximability. We present a dynamic program that solves \bar{x} -MD for scheduling games exactly in pseudopolynomial time, and then convert this dynamic program into an FPTAS. The analysis uses standard techniques for designing approximation schemes (for example, see Schuurman and Woeginger).

Theorem 4.6. *There exists a fully polynomial time approximation scheme for the \bar{x} -maximum dissatisfaction problem for scheduling games.*

Proof. See Appendix A.2. □

Combining Theorem 4.5 and Theorem 4.6, gives us the following result.

Theorem 4.7. *There exists a fully polynomial time approximation scheme for computing the least core value of scheduling games.*

5 Submodular profits and a special case from matroid optimization

Up to this point, we have only considered cooperative games in which agents are assigned a cost for their joint actions. But what about cooperative games in which agents act together to collect a reward, or profit? Consider a cooperative game (N, v) where $v(S)$ represents the *profit* allocated to the agents in S . For these games, solution concepts should reflect the rationality of a profit allocation; for example, the core for a profit cooperative game (N, v) is defined as the set of all profit allocations x such that

$$\begin{aligned} x(N) &= v(N), \\ x(S) &\geq v(S) \quad \text{for all } S \subseteq N. \end{aligned}$$

The least core for a profit cooperative game (N, v) is defined in a similar manner: it is the set of all profit allocations x that are optimal for the problem

$$\begin{aligned} \text{[LC-profit]} \quad z^* &= \text{minimize } z \\ &\text{subject to } x(N) = v(N) \\ &\quad x(S) \geq v(S) - z \quad \text{for all } S \subseteq N, S \neq \emptyset, N. \end{aligned}$$

⁹A *fully polynomial time approximation scheme* is an algorithm that finds a solution whose objective function value is within a factor $(1 + \epsilon)$ of the optimal value for any $\epsilon > 0$, and whose running time is polynomial in the input size and $1/\epsilon$.

The least core value of (N, v) is the optimal value z^* to this optimization problem. Note that it still reflects the minimum penalty to a coalition needed to ensure the existence of an efficient and stable profit allocation.

If v is nonnegative, submodular and $v(\emptyset) = 0$, we call (N, v) a *submodular profit cooperative game*. It is straightforward to see that all the results established for supermodular cost cooperative games in Section 3 also hold true for submodular profit cooperative games, with the following natural modifications. For a cooperative game (N, v) with v representing profits, the dissatisfaction for any subset of agents S under a profit allocation x is defined as $e(x, S) = v(S) - x(S)$. We define the polytope B_v as $\{x \in \mathbb{R}^n : x(N) = v(N), x(S) \leq v(S) \text{ for all } S \subseteq N\}$. The x -maximum dissatisfaction problem for a cooperative game (N, v) with v representing profits is still to find a subset S^* such that $e(x, S^*) = \max_{S \subseteq N, S \neq \emptyset, N} e(x, S)$.

5.1 Matroid profit games

Consider the cooperative game (N, v) , defined as follows. Each agent $i \in N$ has a job with unit processing time and a deadline $d_i \in \mathbb{Z}_{>0}$. In addition, each agent $i \in N$ has an associated profit $w_i \in \mathbb{R}_{\geq 0}$, which is earned if job i is completed by its deadline. The profit $v(S)$ to any subset of agents S is the maximum profit attainable by scheduling jobs in S on a single machine. It turns out that if we define

$$\mathcal{I} = \{S \subseteq N : \text{every job in } S \text{ can be completed by its deadline}\},$$

then (N, \mathcal{I}) is a matroid (Gabow and Tarjan 1984). For any family of sets \mathcal{I} , define $\mathcal{I}|S = \{T \in \mathcal{I} : T \subseteq S\}$. In this cooperative game, $v(S)$ is the maximum w -weight of an independent set in $(S, \mathcal{I}|S)$ for any subset of agents S .

In this section, we study the following generalization of the cooperative game described above. Let (N, \mathcal{I}) be a matroid with weights $w_i \in \mathbb{R}_{\geq 0}$ for each $i \in N$. We define $v(S)$ as the maximum w -weight of an independent set in $(S, \mathcal{I}|S)$, for every subset of agents $S \subseteq N$. Then (N, v) defines a cooperative game where the profit to a coalition S is represented by $v(S)$. We call such games *matroid profit games*. Cooperative games that arise from matroid optimization have been considered previously. Nagamochi et al. (1997) studied the computational complexity of various solution concepts for *minimum base games*, in which for a given matroid (N, \mathcal{I}) , the cost $v(S)$ to a coalition S is the *minimum* weight of a basis in $(S, \mathcal{I}|S)$. In these games, the costs to a coalition are *not* necessarily supermodular, and so the results of Section 3 do not apply.

Throughout this section, we assume that the matroid (N, \mathcal{I}) and its restrictions are given by an independence testing oracle that asserts whether or not a given subset $S \subseteq N$ belongs to \mathcal{I} . By Theorem 2.3, matroid profit games are submodular profit cooperative games. It turns out that the x -maximum dissatisfaction problem for matroid profit games is quite tractable: we show that it can be solved exactly in polynomial time for any profit allocation x such that $x(N) = v(N)$.

Theorem 5.1. *Suppose (N, v) is a matroid profit game. Then for any profit allocation x such that $x(N) = v(N)$, the x -maximum dissatisfaction problem for (N, v) can be solved in polynomial time.*

Proof. Fix some profit allocation x such that $x(N) = v(N)$, and let $A = \{i \in N : x_i < 0\}$. Note that any optimal solution of the following relaxation of the x -maximum dissatisfaction problem

$$\max_{S \subseteq N} e(x, S) = \max_{S \subseteq N} \{v(S) - x(S)\} \quad (5.1)$$

must contain all elements of A , since v is nondecreasing. Since A is fixed, the problem (5.1) is equivalent to

$$\max_{S \subseteq N} \{v(S) - x(S \setminus A)\}. \quad (5.2)$$

Let

$$\bar{w}_i = \begin{cases} w_i & \text{if } i \in A \\ w_i - x_i & \text{if } i \in N \setminus A. \end{cases}$$

One can show that a maximum \bar{w} -weight independent set of (N, \mathcal{I}) is an optimal solution for the problem (5.2). Let T^* be such an independent set, and let $\bar{T} = T^* \cup (A \setminus T^*)$. It is straightforward to show that if $\bar{T} \neq \emptyset, N$, then \bar{T} is an optimal solution to x -MD for (N, v) ; otherwise, $\arg \max\{e(x, S) : S \in \{T, N \setminus T\}\}$ for an arbitrary $T \subseteq N$ such that $T \neq \emptyset, N$ is an optimal solution to x -MD for (N, v) . \square

Alternatively, one can solve the x -maximum dissatisfaction problem for matroid profit games by solving the following integer linear program:

$$\text{maximize } \sum_{i \in N} (w_i \alpha_i - x_i \beta_i) \quad (5.3a)$$

$$\text{subject to } \sum_{i \in S} \alpha_i \leq r(S) \quad \text{for all } S \subseteq N, \quad (5.3b)$$

$$\alpha_i \leq \beta_i \quad \text{for all } i \in N, \quad (5.3c)$$

$$\alpha_i \in \{0, 1\} \quad \text{for all } i \in N, \quad (5.3d)$$

$$\beta_i \in \{0, 1\} \quad \text{for all } i \in N, \quad (5.3e)$$

where $r : 2^N \rightarrow \mathbb{Z}$ is the rank function of the matroid (N, \mathcal{I}) . Using ideas similar to those in the proof of Theorem 5.1, one can show that the linear programming relaxation of the integer linear program (5.3a)-(5.3e) is in fact integral. The LP relaxation of (5.3a)-(5.3e) can be solved in polynomial time with the ellipsoid method, using a polynomial-time algorithm for submodular function minimization (for example, see Grötschel et al. 1988) to separate the inequalities (5.3b), which are exponential in number.

By the discussion earlier in this section, if (N, v) is a submodular profit cooperative game and we have a ρ -approximation algorithm for the x -maximum dissatisfaction problem for (N, v) for any given profit allocation x such that $x(N) = v(N)$, then we have a ρ -approximation algorithm for computing the least core value of (N, v) . This translation in approximability is accomplished by using the ellipsoid method to find feasible solutions to [LC-profit], with an approximation algorithm for the x -maximum dissatisfaction problem as an approximate separation oracle. Therefore, by Theorem 5.1, we immediately obtain the following theorem:

Theorem 5.2. *Suppose (N, v) is a matroid profit game. Then there exists a polynomial-time algorithm for*

- (a) *computing the least core value of (N, v) , and*
- (b) *computing a cost allocation in the least core of (N, v) .*

6 Conclusion

In this paper, we considered cooperative games with supermodular costs. As we demonstrated, supermodular costs arise in a variety of situations, and especially in scheduling and network design. In such situations, cooperation amongst agents is unlikely: as a coalition grows, the cost of adding a particular agent increases, making the prospect of cooperation less appealing. In circumstances where the failure to cooperate can cause negative externalities, one may be interested in methods of encouraging cooperation. One way of encouraging cooperation in these situations is through the least core value of a cooperative game. The least core value of a cooperative game is the minimum penalty we need to charge a coalition for acting independently that ensures the existence of an efficient and stable cost allocation. The set of all such cost allocations is the least core. We showed that computing the least core value of supermodular cost cooperative games is strongly NP-hard, and

provided a general framework for approximating the least core value of these games. Using this framework with the approximation algorithm for submodular function maximization of Feige et al. (2007), we obtained a $(3 + \epsilon)$ -approximation algorithm for computing the least core value of supermodular cost cooperative games. We also showed as a by-product that we can compute a cost allocation in the 2-approximate least core of supermodular cost cooperative games in polynomial time. Finally, we applied this general framework to two special cases. For scheduling games, we gave an explicit formula for an element of the least core, and used this to design a fully polynomial-time approximation scheme for computing the least core value of these games. For matroid profit games, we showed how to compute the least core value as well as a cost allocation in the least core of these games in polynomial time.

There are several interesting directions for future research that extend from this work. For example, it would be interesting to study the f -least core for supermodular cost cooperative games, for various forms of f , as it provides a natural way to model different penalties for defection for different coalitions. Another interesting direction of research related to this work is to study the nucleolus of supermodular cost cooperative games. The computational complexity of computing the nucleolus of supermodular cost cooperative games is open. It would also be interesting to investigate whether our framework for least core approximation can be used in a fruitful manner to compute cost allocations that approximate the nucleolus of supermodular cost cooperative games. Last, but not least, the computational complexity of computing an element of the least core of supermodular cost cooperative games remains open.

Acknowledgments

The authors would like to thank the associate editor and two anonymous referees for their time, effort and feedback that helped significantly to improve the presentation of this paper. In particular, the authors are grateful to one anonymous referee for pointing out the integer linear programming formulation mentioned in Section 5.1. This research was supported by the National Science Foundation (DMI-0426686).

References

- R. J. Aumann and M. Maschler. The bargaining set for cooperative games. In M. Dresher, L. S. Shapley, and A. W. Tucker, editors, *Advances in Game Theory*, volume 52 of *Annals of Mathematics Studies*, pages 443–476, Princeton, 1964. Princeton University Press.
- C. G. Bird. Cost-allocation for a spanning tree. *Networks*, 6:335–350, 1976.
- G. Birkhoff. On the structure of abstract algebras. In *Proceedings of the Cambridge Philosophical Society*, volume 31, pages 433–454, 1935.
- J. Bruno, E. G. Coffman, and R. Sethi. Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, 17:382–387, 1974.
- X. Chen and J. Zhang. Duality approaches to economic lot sizing games. Manuscript, 2006.
- I. Curiel, G. Pederzoli, and S. Tijs. Sequencing games. *European Journal of Operational Research*, 40: 344–351, 1989.
- X. Deng. Combinatorial optimization and coalition games. In *Handbook of Combinatorial Optimization*, volume 2, pages 77–103. Kluwer, 1998.
- X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19:257–266, 1994.

- J. Edmonds. Submodular functions, matroids, and certain polyhedra. In R. Guy, H. Hanani, N. Sauer, and J. Schönheim, editors, *Combinatorial Structures and Their Applications (Calgary International Conference on Combinatorial Structures and Their Applications)*, pages 69–87, 1970.
- E. Einy, R. Holzman, and D. Monderer. On the least core and the Mas-Colell Bargaining set. *Games and Economic Behavior*, 28:181–188, 1999.
- U. Faigle and W. Kern. On some approximately balanced combinatorial cooperative games. *Mathematical Methods of Operations Research*, 38:141–152, 1993.
- U. Faigle, W. Kern, and D. Paulusma. Note on the computational complexity of least core concepts for min-cost spanning tree games. *Mathematical Methods of Operations Research*, 52:23–38, 2000.
- U. Faigle, W. Kern, and J. Kuipers. On the computation of the nucleolus of a cooperative game. *International Journal of Game Theory*, 30:79–98, 2001.
- U. Feige, V. Mirrokni, and J. Vondrák. Maximizing non-monotone submodular functions. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007)*, pages 461–471, 2007.
- L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006)*, pages 611–620, 2006.
- H. Gabow and R. Tarjan. Efficient algorithms for a family of matroid intersection problems. *Journal of Algorithms*, 5:80–131, 1984.
- M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.
- Y. Gerchak and D. Gupta. On apportioning costs to customers in centralized continuous review inventory systems. *Journal of Operations Management*, 10:546–551, 1991.
- D. B. Gillies. Solutions to general non-zero-sum games. In A. W. Tucker and R. D. Luce, editors, *Contributions to the Theory of Games, Volume IV*, volume 40 of *Annals of Mathematics Studies*, pages 47–85, Princeton, 1959. Princeton University Press.
- M. X. Goemans and M. Skutella. Cooperative facility location games. *Journal of Algorithms*, 50:194–214, 2004.
- M. X. Goemans, M. Queyranne, A. S. Schulz, M. Skutella, and Y. Wang. Single machine scheduling with release dates. *SIAM Journal on Discrete Mathematics*, 15:165–192, 2002.
- R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.
- D. Granot and G. Huberman. Minimum cost spanning tree games. *Mathematical Programming*, 21:1–18, 1981.
- M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1988.
- B. Hartman, M. Dror, and M. Shaked. Cores of inventory centralization games. *Games and Economic Behavior*, 31:26–49, 2000.

- J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48:798–859, 2001.
- K. Jain, M. Mahdian, and M. Salavatipour. Packing Steiner trees. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2003)*, pages 266–274, 2003.
- K. Jansen. Approximate strong separation with application in fractional graph coloring and preemptive scheduling. *Theoretical Computer Science*, 302:239–256, 2003.
- E. Kalai and E. Zemel. Totally balanced games and games of flow. *Mathematics of Operations Research*, 7: 476–478, 1982a.
- E. Kalai and E. Zemel. Generalized network problems yielding totally balanced games. *Operations Research*, 30:998–1008, 1982b.
- W. Kern and D. Paulusma. Matching games: the least core and the nucleolus. *Mathematics of Operations Research*, 28:294–308, 2003.
- F. Maniquet. A characterization of the Shapley value in queueing problems. *Journal of Economic Theory*, 109:90–103, 2003.
- A. Mas-Colell. An equivalence theorem for a bargaining set. *Journal of Mathematical Economics*, 18: 129–139, 1989.
- M. Maschler, B. Peleg, and L. S. Shapley. Geometric properties of the kernel, nucleolus, and related solution concepts. *Mathematics of Operations Research*, 4:303–338, 1979.
- N. Meggido. Computational complexity of the game theory approach to cost allocation for a tree. *Mathematics of Operations Research*, 3:189–196, 1978.
- D. Mishra and B. Rangarajan. Cost sharing in a job scheduling problem using the Shapley value. In *Proceedings of the 6th ACM Conference on Electronic Commerce*, pages 232–239, 2005.
- H. Nagamochi, D.-Z. Zeng, N. Kabutoya, and T. Ibaraki. Complexity of the minimum base game on matroids. *Mathematics of Operations Research*, 22:146–164, 1997.
- G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, N.Y., 1988.
- G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of algorithms for maximizing submodular set functions—I. *Mathematical Programming*, 14:265–294, 1978.
- G. Owen. On the core of linear production games. *Mathematical Programming*, 9:358–370, 1975.
- J. Potters, I. Curiel, and S. Tijs. Traveling salesman games. *Mathematical Programming*, 53:199–211, 1991.
- M. Queyranne. Structure of a simple scheduling polyhedron. *Mathematical Programming*, 58:263–285, 1993.
- M. Queyranne and A. S. Schulz. Scheduling unit jobs with compatible release dates on parallel machines with nonstationary speeds. In E. Balas and J. Clausen, editors, *Integer Programming and Combinatorial Optimization (IPCO 1995)*, volume 920 of *Lecture Notes in Computer Science*, pages 307–320, Berlin, 1995. Springer.
- A. E. Roth, editor. *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

- S. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23:116–127, 1976.
- D. Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics*, 17:1163–1170, 1969.
- P. Schuurman and G. J. Woeginger. Approximation schemes - a tutorial. Preliminary version of a chapter for *Lectures on Scheduling*, edited by R. H. Möhring, C. N. Potts, A. S. Schulz, G. J. Woeginger, and L. A. Wolsey.
- L. S. Shapley. A value for n -person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games, Volume II*, volume 28 of *Annals of Mathematics Studies*, pages 307–317, Princeton, 1953. Princeton University Press.
- L. S. Shapley. Cores of convex games. *International Journal of Game Theory*, 1:11–26, 1971.
- L. S. Shapley and M. Shubik. Quasi-cores in a monetary economy with nonconvex preferences. *Econometrica*, 34:805–827, 1966.
- L. S. Shapley and M. Shubik. The assignment game I: the core. *International Journal of Game Theory*, 1:111–130, 1971.
- H. Simon. Theories of bounded rationality. In R. Radner, editor, *Decision and Organization*. North-Holland, 1972.
- W. E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.
- W. van den Heuvel, P. Borm, and H. Hamers. Economic lot-sizing games. *European Journal of Operational Research*, 176:1117–1130, 2005.
- H. Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57:509–533, 1935.
- L. A. Wolsey. Mixed integer programming formulations for production planning and scheduling problems. Invited talk at the 12th International Symposium on Mathematical Programming, MIT, Cambridge, MA, 1985.

A Supplement

A.1 Proof of Theorem 3.6

A.1.1 Overview

We begin in Appendix A.1.2 by establishing some definitions from polyhedral theory. Then, in Appendix A.1.3, we show that an approximate separation oracle for a given polytope, in conjunction with the ellipsoid method, can be used to either find an element in an “approximation” of that polytope, or determine that the polytope is empty. The result we show is actually more general than needed for the proof of Theorem 3.6. Finally, in Appendix A.1.4, we show how to use the results from Appendix A.1.3 to approximately solve the least core linear program [LC]. The ideas here closely follow and generalize the analyses found in Grötschel et al. (1988) and Jansen (2003).

A.1.2 Preliminaries

To simplify the exposition, for the remainder of this appendix, we assume that v is integer-valued.

For a symmetric matrix $A \in \mathbb{R}^{n \times n}$, we denote the *spectral norm* of A as

$$\|A\| = \max \{|\lambda| : \lambda \text{ is an eigenvalue of } A\} = \max \{|x^\top A x| : \|x\| = 1\}.$$

Finally, we define for any vector $a \in \mathbb{R}^n$ and positive definite matrix A , the *ellipsoid*

$$E(A, a) = \{x \in \mathbb{R}^n : (x - a)^\top A^{-1}(x - a) \leq 1\}.$$

Suppose $K \subseteq \mathbb{R}^n$ is a polyhedron, and φ and v are positive integers. We say that K has *facet complexity at most φ* if there exists a system of inequalities with rational coefficients that has solution set K and such that the encoding length of each inequality of the system is at most φ . We say that K has *vertex complexity at most v* if there exist finite sets V, E of rational vectors such that $K = \text{conv}(V) + \text{cone}(E)$ and such that each of the vectors in V and E has encoding length at most v . We will use the following well-known lemma that relates the facet complexity and the vertex complexity of a polyhedron.

Lemma A.1 (Grötschel et al. 1988, 6.2.4). *Let $K \subseteq \mathbb{R}^n$ be a polyhedron.*

- (a) *If K has facet complexity at most φ , then K has vertex complexity at most $4n^2\varphi$.*
- (b) *If K has vertex complexity at most v , then K has facet complexity at most $3n^2v$.*

A *well-described polyhedron* is a triple $(K; n, \varphi)$ where $K \subseteq \mathbb{R}^n$ is a polyhedron with facet complexity at most φ . The encoding length of a well-described polyhedron $(K; n, \varphi)$ is $\varphi + n$.

A.1.3 Approximate separation implies approximate non-emptiness

Let us assume that $(K; n, \varphi)$ is a bounded, rational, well-described polyhedron in \mathbb{R}^n . In other words, $K \subseteq \mathbb{R}^n$ is a rational polytope with facet complexity at most φ . Let \bar{K} be an “approximation” to K . Consider the following problem:

Strong approximate separation problem for K and its approximation \bar{K} (S-APP-SEP). *Given $y \in \mathbb{Q}^n$, either (i) assert $y \in \bar{K}$, or (ii) find a hyperplane that separates y from K : find $c \in \mathbb{Q}^n$ such that $c^\top y > c^\top x$ for all $x \in K$ and $\|c\|_\infty = 1$.*

Suppose we have an oracle for S-APP-SEP. We use this approximate separation oracle in the ellipsoid method as follows.

Algorithm A.2 (Central-cut ellipsoid method with approximate separation oracle (APP-ELL)).

Input: $\epsilon \in \mathbb{Q}$ such that $\epsilon \in (0, 1)$, bounded rational polyhedron $K \subseteq \mathbb{R}^n$ given by an oracle for S-APP-SEP, $R \in \mathbb{Q}$ such that $K \subseteq E(R^2 I, 0)$ (where I denotes the identity matrix).

Output: either

1. $y \in \bar{K}$, or
2. positive definite $A \in \mathbb{Q}^{n \times n}$, $a \in \mathbb{Q}^n$ such that $K \subseteq E(A, a)$ and $\text{vol}(E(A, a)) \leq \epsilon$.

1. Set the following values:

$$N = \lceil 5n |\log \epsilon| + 5n^2 |\log 2R| \rceil \quad (\text{A.1})$$

$$p = 8N \quad (\text{A.2})$$

2. Generate the sequence of ellipsoids $E(A_0, a_0), E(A_1, a_1), \dots, E(A_N, a_N)$ as follows:

- Initialize the sequence:

$$a_0 = 0 \quad (\text{A.3})$$

$$A_0 = R^2 I \quad (\text{A.4})$$

- For $k = 0, \dots, N - 1$, call S-APP-SEP oracle for K with input $y = a_k$.
- If the S-APP-SEP oracle asserts $a_k \in \bar{K}$, return a_k . Stop.
- If the S-APP-SEP oracle returns $c_k \in \mathbb{Q}^n$ such that

$$\|c_k\|_\infty = 1 \quad (\text{A.5})$$

$$c_k^\top a_k > c_k^\top x \quad \text{for all } x \in K \quad (\text{A.6})$$

then compute

$$a_{k+1} \approx a_k - \frac{1}{n+1} \frac{A_k c_k}{\sqrt{c_k^\top A_k c_k}} \quad (\text{A.7})$$

$$A_{k+1} \approx \frac{2n^2 + 3}{2n^2} \left(A_k - \frac{2}{n+1} \frac{A_k c_k c_k^\top A_k}{c_k^\top A_k c_k} \right) \quad (\text{A.8})$$

where “ \approx ” means the computations are done with p digits after the binary point.

- If $k = N$, return a_N, A_N . Stop.

To prove the correctness of the algorithm, we need the following lemma.

Lemma A.3 (Grötschel et al. 1988, 3.2.8-3.2.10). *Let $K \subseteq \mathbb{R}^n$ be a convex set such that $K \subseteq E(R^2 I, 0)$. Let N, p be defined as in (A.1)-(A.2). Suppose A_k and a_k ($k = 0, 1, \dots, N$) are defined as in (A.3)-(A.4) and (A.7)-(A.8), and c_k ($k = 0, 1, \dots, N$) satisfy (A.5)-(A.6). Then, the following statements hold for $k = 0, 1, \dots, N$:*

- (a) A_k is positive definite.
- (b) $\|a_k\| \leq R2^k$, $\|A_k\| \leq R^2 2^k$, and $\|A_k^{-1}\| \leq R^{-2} 4^k$.
- (c) $K \subseteq E(A_k, a_k)$.
- (d) $\text{vol}(E(A_{k+1}, a_{k+1})) \leq e^{-\frac{1}{5n}} \text{vol}(E(A_k, a_k))$.

Using the above lemma, we can show:

Theorem A.4. *Algorithm A.2 (APP-ELL) is correct.*

Proof. Lemma A.3 immediately implies that A_k and a_k ($k = 0, 1, \dots, N$) as defined in (A.3)-(A.4) and (A.7)-(A.8) are well-defined and have polynomial encoding lengths.

If the algorithm stops with $k < N$, the algorithm terminates correctly by construction. If the algorithm returns a_N and A_N , then Lemma A.3 implies that $K \subseteq E(A_N, a_N)$ and

$$\begin{aligned} \text{vol}(E(A_N, a_N)) &\leq e^{-\frac{N}{5n}} \text{vol}(E(A_0, a_0)) \\ &\leq e^{-\frac{N}{5n}} (2R)^n \\ &< 2^{-\frac{N}{5n}} (2R)^n \\ &\leq \epsilon. \end{aligned}$$

So if $k = N$, the algorithm terminates correctly. □

Before proceeding further, we need the following lemma.

Lemma A.5 (Grötschel et al. 1988, pp. 175-176). *Let $(K; n, \varphi)$ be a well-described polyhedron. In addition, let $\epsilon = 2^{-48n^5\varphi}$. Suppose $K \subseteq E(A, a)$ where $\text{vol}(E(A, a)) \leq \epsilon$. Then there exists $f \in \mathbb{Z}^n$ and $g \in \mathbb{Z}_{>0}$ such that $f \neq 0$ and $K \subseteq \{x \in \mathbb{R}^n : f^\top x = g\}$. Moreover, f and g can be found in time polynomial in n , φ , and the encoding length of A^{-1} .*

Consider the following problem:

Approximate non-emptiness problem for K and its approximation \bar{K} (APP-NEMPT). *Either (i) find a vector $y \in \bar{K}$ or (ii) assert K is empty.*

We are now ready to show the main result of this subappendix: we can use APP-ELL (Algorithm A.2) in conjunction with an oracle for S-APP-SEP to solve APP-NEMPT.

Theorem A.6. *Suppose there exists an algorithm that can solve S-APP-SEP in time polynomial in n and φ . Then, there exists an algorithm that can solve APP-NEMPT in time polynomial in n and φ .*

Proof. By assumption, K has facet complexity at most φ . Therefore, by Lemma A.1, K has vertex complexity at most $4n^2\varphi$. Apply APP-ELL (Algorithm A.2) to K with $R = 2^{4n^2\varphi}$ and $\epsilon = 2^{-48n^5\varphi}$. If APP-ELL returns a vector $y \in \bar{K}$, then we have solved APP-NEMPT, and we can stop. Otherwise, APP-ELL returns an ellipsoid $E \subseteq \mathbb{R}^n$ such that $K \subseteq E$ and $\text{vol}(E) \leq \epsilon$. Then, by Lemma A.5, we can find $f^1 \in \mathbb{Z}^n$ and $g^1 \in \mathbb{Z}_{>0}$ such that $f^1 \neq 0$ and $K \subseteq \{x \in \mathbb{R}^n : (f^1)^\top x = g^1\}$. Without loss of generality, assume that $f_1^1 \neq 0$.

Suppose that we have found k linearly independent vectors $f^1, \dots, f^k \in \mathbb{Z}^n$ and $g^1, \dots, g^k \in \mathbb{Z}_{>0}$ such that $f^i \neq 0$ for $i = 1, \dots, k$ and

$$K \subseteq \{x \in \mathbb{R}^n : (F_1 \quad F_2)x = g\}$$

where $F_1 \in \mathbb{Z}^{k \times k}$ is upper triangular with non-zero diagonal entries and $F_2 \in \mathbb{Z}^{k \times (n-k)}$ such that

$$(F_1 \quad F_2) = \begin{pmatrix} (f^1)^\top \\ \vdots \\ (f^k)^\top \end{pmatrix} \quad \text{and} \quad g = \begin{pmatrix} g^1 \\ \vdots \\ g^k \end{pmatrix}.$$

We show how to find $f^{k+1} \in \mathbb{Z}^n$, $g^{k+1} \in \mathbb{Z}_{>0}$ such that f^1, \dots, f^k, f^{k+1} are linearly independent, $f^{k+1} \neq 0$, and

$$K \subseteq \{x \in \mathbb{R}^n : (f^1)^\top x = g^1, \dots, (f^k)^\top x = g^k, (f^{k+1})^\top x = g^{k+1}\}.$$

Let

$$K_k = \left\{ u \in \mathbb{R}^{n-k} : \exists z \in \mathbb{R}^k \text{ such that } \begin{pmatrix} z \\ u \end{pmatrix} \in K \right\}.$$

Therefore, $w \in K_k$ if and only if

$$\begin{pmatrix} z \\ w \end{pmatrix} \in K \subseteq \{x \in \mathbb{R}^n : (F_1 \quad F_2)x = g\}$$

for some $z \in \mathbb{R}^k$, which happens if and only if

$$\begin{pmatrix} F_1^{-1}g - F_1^{-1}F_2w \\ w \end{pmatrix} \in K.$$

Note that for any vertex u^* of K_k , there exists $z^* \in \mathbb{R}^k$ such that $\begin{pmatrix} z^* \\ u^* \end{pmatrix}$ is a vertex of K . Therefore, since K has vertex complexity at most $4n^2\varphi$, K_k has vertex complexity at most $4n^2\varphi$. This implies that K_k has facet complexity at most $\varphi' = 3n^2(4n^2\varphi)$. Apply APP-ELL to K_k with $R = 2^{4n^2\varphi'}$ and $\epsilon = 2^{-48n^5\varphi'}$, using the following modified approximate separation oracle for K_k :

Input: $w \in \mathbb{R}^{n-k}$.

Output: either

1. assert $y \in \bar{K}$, where

$$y = \begin{pmatrix} F_1^{-1}g - F_1^{-1}F_2w \\ w \end{pmatrix}$$

2. find $\bar{c} \in \mathbb{Q}^{n-k}$ such that $\|\bar{c}\|_\infty = 1$ and $\bar{c}^\top w > \bar{c}^\top u$ for all $u \in K_k$.

1. Apply S-APP-SEP oracle for K on

$$y = \begin{pmatrix} F_1^{-1}g - F_1^{-1}F_2w \\ w \end{pmatrix}$$

2. If the S-APP-SEP oracle asserts $y \in \bar{K}$, then assert $y \in \bar{K}$. Stop.
3. Otherwise, the S-APP-SEP oracle returns $c \in \mathbb{Q}^n$ such that $c^\top y > c^\top x$ for all $x \in K$. Let $c^1 \in \mathbb{Q}^k$ and $c^2 \in \mathbb{Q}^{n-k}$ such that

$$c = \begin{pmatrix} c^1 \\ c^2 \end{pmatrix}$$

Therefore,

$$(c^1)^\top (F_1^{-1}g - F_1^{-1}F_2w) + (c^2)^\top w > (c^1)^\top (F_1^{-1}g - F_1^{-1}F_2u) + (c^2)^\top u \quad \text{for all } u \in K^k.$$

Or equivalently,

$$((c^2)^\top - (c^1)^\top F_1^{-1}F_2)w > ((c^2)^\top - (c^1)^\top F_1^{-1}F_2)u \quad \text{for all } u \in K^k.$$

Return

$$\bar{c} = \frac{c^2 - (F_1^{-1}F_2)^\top c^1}{\|c^2 - (F_1^{-1}F_2)^\top c^1\|_\infty}$$

as the vector representing a hyperplane that separates w and K^k . Stop.

If APP-ELL returns a vector $y \in \bar{K}$, then we have solved APP-NEMPT and we are done. Otherwise, APP-ELL returns an ellipsoid $E^k \subseteq \mathbb{R}^{n-k}$ such that $K^k \subseteq E^k$ and $\text{vol}(E^k) \leq \epsilon$. Therefore, by Lemma A.5 we can find $\bar{f}^{k+1} \in \mathbb{Z}^{n-k}$ and $g^{k+1} \in \mathbb{Z}_{>0}$ such that $K^k \subseteq \{y \in \mathbb{R}^{n-k} : \bar{f}^{k+1}y = g^{k+1}\}$. Without loss of generality, let $\bar{f}_1^{k+1} \neq 0$. Let $f^{k+1} \in \mathbb{Z}^n$ such that

$$f^{k+1} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \bar{f}^{k+1} \end{pmatrix}.$$

It follows that $K \subseteq \{x \in \mathbb{R}^n : (f^{k+1})^\top x = g^{k+1}\}$. By the induction hypothesis,

$$K \subseteq \{x \in \mathbb{R}^n : (f^1)^\top x = g^1, \dots, (f^k)^\top x = g^k, (f^{k+1})^\top x = g^{k+1}\}$$

and f^1, \dots, f^k, f^{k+1} are linearly independent.

When $k = n$, we have that

$$K \subseteq \{x \in \mathbb{R}^n : (f^1)^\top x = g^1, \dots, (f^n)^\top x = g^n\}.$$

Since f^1, \dots, f^n are linearly independent, K must be equal to the unique vector y in $\{x \in \mathbb{R}^n : (f^1)^\top x = g^1, \dots, (f^n)^\top x = g^n\}$, or empty. Running the S-APP-SEP oracle for K on y , we either determine that $y \in \bar{K}$ or K is empty.

By Lemma A.3(b) and Lemma A.5, we can find the vectors f^1, \dots, f^n and the scalars g^1, \dots, g^n in time polynomial in n and φ . In addition, the inputs ϵ and R defined above imply that the calls to APP-ELL above run in time polynomially bounded by n, φ , and the running time of the S-APP-SEP oracle (which is assumed to be polynomial in n and φ). Since at most n calls to APP-ELL are made, the prescribed method above solves APP-NEMPT in time polynomial in n and φ . \square

A.1.4 Approximately solving the least core optimization problem

We define Q to be the feasible region of the linear program [LC]:

$$Q = \{x \in \mathbb{R}^N, z \in \mathbb{R} : x(N) = v(N), x(S) \leq v(S) + z \text{ for all } S \subseteq N, S \neq \emptyset, N\}.$$

In addition, for any fixed $\gamma \geq 0$, let

$$Q_\gamma = \{x \in \mathbb{R}^N : x(N) = v(N), x(S) \leq v(S) + \gamma \text{ for all } S \subseteq N, S \neq \emptyset, N\}.$$

We define the strong approximate separation problem and approximate non-emptiness problem for Q_γ using $Q_{\rho\gamma}$ as its ‘‘approximation’’

Strong approximate separation problem for Q_γ and its approximation $Q_{\rho\gamma}$ (S-APP-SEP- Q_γ).
 Given $x \in \mathbb{Q}^N$ such that $x(N) = v(N)$, either (i) assert $x \in Q_{\rho\gamma}$ or (ii) find a hyperplane separating x and Q_γ .

Approximate non-emptiness problem for Q_γ and its approximation $Q_{\rho\gamma}$ (APP-NEMPT- Q_γ).
 Either (i) find $x \in Q_{\rho\gamma}$ or (ii) assert Q_γ is empty.

Since the facet complexity of Q_γ is polynomially bounded by n and the encoding length of $v(N) + \gamma$, Theorem A.6 implies the following theorem.

Theorem A.7. Fix γ so that its encoding length is polynomially bounded by n and $\log v(N)$. Suppose S-APP-SEP- Q_γ can be solved in time polynomial in n and $\log v(N)$. Then APP-NEMPT- Q_γ can be solved in time polynomial in n and $\log v(N)$.

The following lemma is a consequence of Theorem A.7 and the fact that an approximation algorithm for the x -maximum dissatisfaction problem can be used to solve the approximate separation problem for x and Q_γ .

Lemma A.8. Fix γ so that its encoding length is polynomially bounded by n and $\log v(N)$. Suppose (N, v) is a cooperative game, and there exists a ρ -approximation algorithm for the x -maximum dissatisfaction problem for (N, v) , for all cost allocations x such that $x(N) = v(N)$. Then APP-NEMPT- Q_γ can be solved in time polynomial in n and $\log v(N)$.

Proof. Fix some cost allocation x such that $x(N) = v(N)$. Suppose we run a ρ -approximation algorithm for the x -maximum dissatisfaction problem for (N, v) , and it outputs \bar{S} . If $e(x, \bar{S}) \leq \gamma$, then for all $S \subseteq N, S \neq \emptyset, N$, we have that

$$x(S) - v(S) \leq \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} e(x, S) \leq \rho e(x, \bar{S}) \leq \rho\gamma,$$

and therefore $x \in Q_{\rho\gamma}$. Otherwise, $e(x, \bar{S}) > \gamma$, and for all $y \in Q_\gamma$ we have that

$$x(\bar{S}) - v(\bar{S}) > \gamma \geq y(\bar{S}) - v(\bar{S}).$$

So using a ρ -approximation algorithm for the x -maximum dissatisfaction problem for (N, v) allows us to solve S-APP-SEP- Q_γ in time polynomial in n and $\log v(N)$, which by Theorem A.7, allows us to solve APP-NEMPT- Q_γ in time polynomial in n and $\log v(N)$. \square

Finally, we are ready to prove Theorem 3.6. We do this by showing that using a polynomial-time algorithm for APP-NEMPT- Q_γ in conjunction with binary search yields an appropriate cost allocation and approximation to the least core value of (N, v) .

Proof of Theorem 3.6. Suppose that (N, v) is a supermodular cost cooperative game, and \mathcal{A} is an algorithm that solves APP-NEMPT- Q_γ in time polynomial in n and $\log v(N)$ for every $\gamma \geq 0$ whose encoding length is polynomially bounded by n and $\log v(N)$. Since we assume that a ρ -approximation algorithm for the x -maximum dissatisfaction problem for (N, v) exists for every cost allocation x such that $x(N) = v(N)$, by Lemma A.8, such an algorithm \mathcal{A} exists.

Consider the following algorithm:

Input: supermodular cost cooperative game (N, v) with v integer-valued; algorithm \mathcal{A} that solves APP-NEMPT- Q_γ for every $\gamma \geq 0$ whose encoding length is polynomially bounded by n and $\log v(N)$.

Output: a feasible solution (x, z) to the least core linear program [LC] for (N, v) .

1. Set the following values:

$$m = 4(n + 1)^2(2(n + 1) + \lceil \log(v(N) + 1) \rceil + 1), \quad (\text{A.9a})$$

$$M = 2^m, \quad (\text{A.9b})$$

$$\epsilon = (2M)^{-2}. \quad (\text{A.9c})$$

2. Using \mathcal{A} , find $\bar{\gamma} \in \mathbb{Q}$ by binary search on $[0, v(N)]$ such that $Q_{\bar{\gamma}-\epsilon}$ is empty, but $Q_{\rho\bar{\gamma}}$ is non-empty. Denote the vector that \mathcal{A} finds in $Q_{\rho\bar{\gamma}}$ by \bar{x} .

3. Find $p, q \in \mathbb{Z}$ such that

$$1 \leq q \leq 2M \quad \text{and} \quad \left| \bar{\gamma} - \frac{p}{q} \right| < \frac{1}{2Mq}. \quad (\text{A.10})$$

Use \mathcal{A} to solve APP-NEMPT- $Q_{p/q}$. If \mathcal{A} finds a vector in $Q_{\rho p/q}$, denote that vector by \hat{x} .

4. Output:

- If \mathcal{A} finds a vector $\hat{x} \in Q_{\rho p/q}$ in Step 3, and $p/q < \bar{\gamma}$, then output $(x, z) = (\hat{x}, \rho p/q)$.
- Otherwise, output $(x, z) = (\bar{x}, \rho \bar{\gamma})$.

First, we establish that the above algorithm is well-defined, by proving the following claims:

1. *The binary search interval prescribed in Step 2 is valid.* Consider the uniform cost allocation x where $x_i = v(N)/n$ for all $i \in N$. Since v is nonnegative, $(x, v(N))$ is feasible for [LC]: for any $S \subseteq N$, $S \neq \emptyset$, N , we have that $v(S) + v(N) \geq |S|v(N)/n = x(S)$. Therefore, $z^* \leq v(N)$. Since v is supermodular and $v(\emptyset) = 0$, it follows that $z^* \geq 0$. So, the least core value of (N, v) lies in the interval $[0, v(N)]$.
2. *Every trial value of $\bar{\gamma}$ in the binary search of Step 2 has encoding length polynomially bounded by n and $\log v(N)$.* Since the binary search of Step 2 is on the interval $[0, v(N)]$, the numerator and denominator of any trial value of $\bar{\gamma}$ is nonnegative. In addition, the binary search of Step 2 undergoes $\lceil \log \frac{v(N)}{\epsilon} \rceil + 1$ iterations. This implies that the denominator of any trial value of $\bar{\gamma}$ is at most

$$2^{\lceil \log \frac{v(N)}{\epsilon} \rceil + 1} \leq 2^{2 + \log \frac{v(N)}{\epsilon}} = \frac{4v(N)}{\epsilon}.$$

Since the binary search is performed on the interval $[0, v(N)]$, the numerator of any trial value of $\bar{\gamma}$ is at most $4v(N)^2/\epsilon$. By (A.9a)-(A.9c), the claim follows.

3. *The integers p and q computed in Step 3 have encoding lengths polynomially bounded by n and $\log v(N)$.* By (A.10), and since $M \geq 1$ and $\bar{\gamma} \in [0, v(N)]$, we have that

$$\begin{aligned} p &< \bar{\gamma}q + \frac{1}{2M} \leq 2Mv(N) + 1, \\ p &> \bar{\gamma}q - \frac{1}{2M} \geq -\frac{1}{2}. \end{aligned}$$

Therefore, $|p| < 2Mv(N) + 1$. Since $|q| \leq 2M$, the claim follows by (A.9a)-(A.9c).

Next, we analyze the running time of the above algorithm. The algorithm makes a total of $O(\log \frac{v(N)}{\epsilon})$ calls to \mathcal{A} , which runs in time polynomial in n and $\log v(N)$ each time it is called. It follows by (A.9a)-(A.9c) that the total running time of \mathcal{A} throughout the algorithm is polynomial in n and $\log v(N)$. By using the method of continued fractions (Grötschel et al. 1988, pp. 134-137), finding integers p and q to satisfy (A.10) in Step 3 of the algorithm can be done in time polynomial in n and $\log v(N)$. Therefore, the above algorithm runs in time polynomial in n and $\log v(N)$.

Finally, we analyze the quality of the solution returned by the above algorithm. We start by showing that $\min\{p/q, \bar{\gamma}\} \leq z^*$ by considering two cases:

1. $\bar{\gamma} - \epsilon < z^* < \bar{\gamma}$. Consider p, q computed in Step 3 of the algorithm. Since v is integer-valued, nonnegative, and supermodular with $v(\emptyset) = 0$, $z^* = r/s$ for some $r \in \mathbb{Z}_{\geq 0}$ and $s \in \mathbb{Z}_{> 0}$. Note that since v is nonnegative, supermodular, and $v(\emptyset) = 0$, the facet complexity of Q is at most $\varphi = 2(n+1) + \lceil \log(v(N)+1) \rceil + 1$. Therefore, the vertex complexity of Q is at most $m = 4(n+1)^2\varphi$, and so $s \in (0, 2^m) = (0, M)$. Since

$$\bar{\gamma} - \frac{r}{s} = \bar{\gamma} - z^* < \epsilon = \frac{1}{(2M)^2} \leq \frac{1}{2Mq},$$

it follows that

$$\left| \frac{p}{q} - z^* \right| = \left| \frac{p}{q} - \frac{r}{s} \right| < \left| \frac{p}{q} - \bar{\gamma} \right| + \left| \bar{\gamma} - \frac{r}{s} \right| < \frac{1}{Mq} < \frac{1}{sq}.$$

Therefore, $z^* = \frac{p}{q}$. It follows that $\min\{p/q, \bar{\gamma}\} \leq z^*$.

2. $z^* \geq \bar{\gamma}$. Clearly, $\min\{p/q, \bar{\gamma}\} \leq z^*$.

With this fact in hand, we now show that the solution (x, z) computed in Step 4 of the above algorithm is feasible in the linear program [LC], and that $z \leq \rho z^*$. We consider the following cases:

1. $p/q < \bar{\gamma}$. In this case, we have that $p/q \leq z^*$. Consider the output of \mathcal{A} in Step 3 of the algorithm:
 - (a) \mathcal{A} finds $\hat{x} \in Q_{\rho p/q}$. Therefore, $(x, z) = (\hat{x}, \rho p/q)$ is feasible in [LC], and $z = \rho p/q \leq \rho z^*$.
 - (b) \mathcal{A} asserts that $Q_{p/q}$ is empty. Therefore, $z^* > p/q$. By the arguments above, we have that $z^* \geq \bar{\gamma}$. So, $(x, z) = (\bar{x}, \rho \bar{\gamma})$ is feasible in [LC], and $z = \rho \bar{\gamma} \leq \rho z^*$.
2. $p/q \geq \bar{\gamma}$. In this case, we have that $z^* \geq \bar{\gamma}$. So, $(x, z) = (\bar{x}, \rho \bar{\gamma})$ is feasible in [LC], and $z \leq \rho z^*$. \square

A.2 Proof of Theorem 4.6

For simplicity of exposition, we consider maximizing $g(S) = 2e(\bar{x}, S)$ for the remainder of this proof.

We determine the maximizer S^* of g by scheduling the jobs in N on two machines: the jobs scheduled on machine 1 will form S^* , and the jobs scheduled on machine 2 will form $N \setminus S^*$. As usual, we consider the jobs in order of nonincreasing weight-to-processing-time ratios (i.e. $1, \dots, n$). We can partition the jobs into S^* and $N \setminus S^*$ sequentially using the following dynamic program. The state space E is partitioned into n disjoint sets, E_1, \dots, E_n . A schedule σ for jobs $\{1, \dots, k\}$ on two machines corresponds to a state $(a, b, c) \in E_k$. The first coordinate a is the sum of processing times of all jobs scheduled by σ on machine 1. The second coordinate b is the sum of processing times of all jobs scheduled by σ on machine 2. The third coordinate c is the running objective value: $v(\{1, \dots, k\})$ minus the sum of weighted completion times on two machines for σ .

Suppose jobs $1, \dots, k-1$ have already been scheduled, and job k is under consideration. If job k is scheduled on machine 1, then the running objective value increases by $w_k(a+b+p_k) - w_k(a+p_k) = w_k b$. If job k is scheduled on machine 2, then the running objective value increases by $w_k(a+b+p_k) - w_k(b+p_k) = w_k a$. This suggests the following dynamic programming algorithm.

Algorithm A.9. (Exact dynamic program)

Input: scheduling game (N, v) with weights w_i , processing times p_i for all $i \in N$.

Output: the optimal value of \bar{x} -MD for scheduling games, $e(\bar{x}, S^*)$.

$$E_1 = \{(p_1, 0, 0), (0, p_1, 0)\}$$

For $k = 2, \dots, n$

For every vector $(a, b, c) \in E_{k-1}$

Put $(a + p_k, b, c + w_k b)$ and $(a, b + p_k, c + w_k a)$ in E_k

Find $(a, b, c) \in E_n$ with maximum c value, c^*

Return $e(\bar{x}, S^*) = \frac{1}{2}g(S^*) = \frac{1}{2}c^*$

Let $P = \sum_{i=1}^n p_i$ and $W = \sum_{i=1}^n w_i$. Each state corresponds to a point in $\{(a, b, c) \in \mathbb{Z}^3 : 0 \leq a \leq P, 0 \leq b \leq P, 0 \leq c \leq WP\}$. Note that for any state $(a, b, c) \in E_k$ for a given $k = 1, \dots, n$, if a is known, b is already determined, and vice-versa. Therefore, the running time of this dynamic program is $O(nWP^2)$.

Let $\delta = (1 + \epsilon/(2n))^{-1}$ for some $0 < \epsilon < 1$. Note that $\delta \in (0, 1)$. In addition, define $L = \lceil \log_{1/\delta} P \rceil$ and $M = \lceil \log_{1/\delta} WP \rceil$. Consider the grid formed by the points $(\delta^{-r}, \delta^{-s}, \delta^{-t})$, $r = 1, \dots, L, s = 1, \dots, L, t = 1, \dots, M$. We divide each of the state sets $E_k, k = 1, \dots, n$, into the boxes formed by the grid:

$$B(r, s, t) = \{(a, b, c) \in \mathbb{R}^3 : \delta^{-r+1} \leq a \leq \delta^{-r}, \delta^{-s+1} \leq b \leq \delta^{-s}, \delta^{-t+1} \leq c \leq \delta^{-t}\}$$

$$r = 1, \dots, L, \quad s = 1, \dots, L, \quad t = 1, \dots, M.$$

Observe that if (a_1, b_1, c_1) and (a_2, b_2, c_2) are in the same box,

$$\delta a_1 \leq a_2 \leq \frac{a_1}{\delta}, \quad \delta b_1 \leq b_2 \leq \frac{b_1}{\delta}, \quad \delta c_1 \leq c_2 \leq \frac{c_1}{\delta}. \quad (\text{A.11})$$

We simplify the state sets E_k by using a *single point in each box* as a representative for all vectors in the same box. We denote these simplified state sets by E_k^δ . The “trimmed” dynamic program is as follows.

Algorithm A.10. (Dynamic program with “trimmed” state space)

Input: scheduling game (N, v) with weights w_i , processing times p_i for all $i \in N$.

Output: an approximation to the optimal value of \bar{x} -MD for scheduling games, $e(\bar{x}, \bar{S})$.

Pick $\epsilon \in (0, 1)$, calculate δ

$$E_1^\delta = \{(p_1, 0, 0), (0, p_1, 0)\}$$

For $k = 2, \dots, n$

For every vector $(a, b, c) \in E_{k-1}^\delta$

Put corresponding representatives of $(a + p_k, b, c + w_k b)$ and $(a, b + p_k, c + w_k a)$ in E_k^δ

Find $(a, b, c) \in E_n^\delta$ with maximum c value, \bar{c}

Return $e(\bar{x}, \bar{S}) = \frac{1}{2}g(\bar{S}) = \frac{1}{2}\bar{c}$

The key property of the “trimmed” state space used in Algorithm A.10 is that every element in the original state space has an element in the “trimmed” state space that is relatively close. In particular,

Lemma A.11. For every $(a, b, c) \in E_k$, there exists a vector $(a', b', c') \in E_k^\delta$ such that

$$a' \geq \delta^k a, \quad b' \geq \delta^k b, \quad c' \geq \delta^k c.$$

Proof. By induction. The base case $k = 1$ holds by (A.11). Assume the induction hypothesis holds for $1, \dots, k-1$. Consider an arbitrary $(a, b, c) \in E_k$. The exact dynamic program puts $(a, b, c) \in E_k$ when it schedules job k . Therefore, $(a, b, c) = (\alpha + p_k, \beta, \gamma + w_k \beta)$ or $(a, b, c) = (\alpha, \beta + p_k, \gamma + w_k \alpha)$ for some $(\alpha, \beta, \gamma) \in E_{k-1}$.

Suppose $(a, b, c) = (\alpha + p_k, \beta, \gamma + w_k \beta)$ for some $(\alpha, \beta, \gamma) \in E_{k-1}$. By the induction hypothesis, there exists a vector $(\alpha', \beta', \gamma') \in E_{k-1}^\delta$ such that $\alpha' \geq \delta^{k-1} \alpha$, $\beta' \geq \delta^{k-1} \beta$, and $\gamma' \geq \delta^{k-1} \gamma$. In the k th phase, the trimmed dynamic program puts a state (a', b', c') in E_k^δ that is in the same box as $(\alpha' + p_k, \beta', \gamma' + w_k \beta')$. Therefore, since $\delta \in (0, 1)$, there exists a vector $(a', b', c') \in E_k^\delta$ such that

$$\begin{aligned} a' &\geq \delta(\alpha' + p_k) \geq \delta^k \alpha + \delta p_k \geq \delta^k (\alpha + p_k) = \delta^k a \\ b' &\geq \delta \beta' \geq \delta^k \beta = \delta^k b \\ c' &\geq \delta(\gamma' + w_k \beta') \geq \delta^k \gamma + \delta^k w_k \beta = \delta^k (\gamma + w_k \beta) = \delta^k c. \end{aligned}$$

The case where $(a, b, c) = (\alpha, \beta + p_k, \gamma + w_k \alpha)$ for some $(\alpha, \beta, \gamma) \in E_{k-1}$ follows similarly. Therefore, the induction step is complete. \square

With Lemma A.11 in hand, we are ready to analyze the performance of the trimmed dynamic programming algorithm. Let $c^* = g(S^*)$, the optimal value of g . Note that there exists a vector $(a^*, b^*, c^*) \in E_n$. By Lemma A.11, there exists a vector $(a', b', c') \in E_n^\delta$ such that $c' \geq \delta^n c^*$. Recall that $\delta = (1 + \epsilon/(2n))^{-1}$ for some $0 < \epsilon < 1$. Since $(1 + \epsilon/(2n))^n \leq 1 + \epsilon$, we have that $c' \geq (1 + \epsilon/(2n))^{-n} c^* \geq (1 + \epsilon)^{-1} c^*$.

As for the running time of the trimmed dynamic programming algorithm, note that each E_k^δ has at most one point from each box, or $O(L^2M)$ points. Therefore, the running time of this algorithm is $O(nL^2M)$. Since $\log z \geq (z - 1)/z$ for any $z \in (0, 1]$, we can bound L and M as follows:

$$L = \left\lceil \frac{\log P}{\log 1/\delta} \right\rceil \leq \left\lceil \left(1 + \frac{2n}{\epsilon}\right) \log P \right\rceil, \quad M = \left\lceil \frac{\log WP}{\log 1/\delta} \right\rceil \leq \left\lceil \left(1 + \frac{2n}{\epsilon}\right) (\log W + \log P) \right\rceil.$$

Therefore, the running time of this algorithm is polynomial in n , $\log W$, $\log P$, and $1/\epsilon$. □