Discrete Optimization 10 (2013) 163–180

Contents lists available at SciVerse ScienceDirect

Discrete Optimization

journal homepage: www.elsevier.com/locate/disopt

Approximating the least core value and least core of cooperative games with supermodular costs

Andreas S. Schulz [a], Nelson A. Uhan [b,*]

[a] Sloan School of Management and Operations Research Center, Massachusetts Institute of Technology, 100 Main Street, E62-569, Cambridge, MA 02142, USA

[b] Mathematics Department, United States Naval Academy, 572C Holloway Road, Chauvenet Hall, Annapolis, MD 21402, USA

ARTICLE INFO

Article history:
Received 14 May 2011
Received in revised form 8 November 2012
Accepted 27 February 2013
Available online 23 March 2013

MSC:
91A12
90C27
90B35
68W25

Keywords:
Cooperative games
Least core
Approximation algorithms
Scheduling
Matroids

ABSTRACT

We study the approximation of the least core value and the least core of supermodular cost cooperative games. We provide a framework for approximation based on oracles that approximately determine maximally violated constraints. This framework yields a 3-approximation algorithm for computing the least core value of supermodular cost cooperative games, and a polynomial-time algorithm for computing a cost allocation in the 2-approximate least core of these games. This approximation framework extends naturally to submodular profit cooperative games. For scheduling games, a special class of supermodular cost cooperative games, we give a fully polynomial-time approximation scheme for computing the least core value. For matroid profit games, a special class of submodular profit cooperative games, we give exact polynomial-time algorithms for computing the least core value as well as a least core cost allocation.

Published by Elsevier B.V.

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 $N = \{1, \ldots, n\}$ represents a set of agents, and $v : 2^N \to \mathbb{R}$ 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 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. [1–5]). Another body of literature has used cooperative game theory to investigate the cost sharing issues in various scheduling-related problems (e.g. [6–9]). Other applications of cooperative game theory to OR-related areas include assignment games [10], linear production games [11], minimum-cost spanning tree games [12,13], network flow games [14,15], traveling salesman games [16], and facility location games [17].

* Corresponding author. Tel.: +1 410 293 6713; fax: +1 410 293 4883.
E-mail addresses: schulz@mit.edu (A.S. Schulz), uhan@usna.edu (N.A. Uhan).

1572-5286/$ – see front matter. Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.disopt.2013.02.002

1.1. The least core and the least core value

Cooperative game theory focuses on how to distribute costs that are collectively incurred by a group of cooperating agents in a "desirable" way. 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* [18]. 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 [19] and later named by Maschler et al. [20] 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

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

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-empty.[1]

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 considered not worth the trouble.

The least core is closely related to some other solution concepts from cooperative game theory. For example, Maschler et al. [20] studied the relationship between the least core and the *nucleolus* [21]. 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 $S \subseteq N$ such that $S \neq \emptyset, N$, in nonincreasing order. The nucleolus is the cost allocation that lexicographically minimizes the excess vector $\theta(x)$. Maschler et al. [20] 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. [22] showed that the least core is always contained in the *Mas-Colell bargaining set* [23]. The Mas-Colell bargaining set of a cooperative game, like the classical bargaining set [24], is the set of all cost allocations that are stable with respect to a particular system of objections and counterobjections.

1.2. Supermodular cost cooperative games

Perhaps the most important and remarkable cooperative games are those with *submodular* costs.[3] A set function $v : 2^N \to \mathbb{R}$ is submodular if

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

[1] The 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.

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

[3] In 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*.

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 [25]. 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 $v : 2^N \to \mathbb{R}$ is supermodular if

$$v(S \cup \{j\}) - v(S) \leq v(S \cup \{j, k\}) - v(S \cup \{k\}) \tag{1.2}$$

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 types of facility location, scheduling, and network design problems have supermodular costs [26,9]. Using the same reasoning as before, our intuition tells us that cooperation amongst rational agents who face supermodular costs is unlikely: now 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 modular[4]).

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. A governing 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 the 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.

1.3. Computational complexity of cooperative game solution concepts

Initiated by Meggido [27] and carried forward by Deng and Papadimitriou [28], computational complexity has been proposed as one measure for evaluating cooperative game solution concepts. Examining the computational complexity of various solution concepts allows us to determine whether they are reasonable within the context of *bounded rationality* (see [29] for an extensive discussion)—the hypothesis that economic agents, in reality, 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.

The computational complexity of computing a cost allocation in the least core has been studied previously in several contexts. Faigle et al. [30] showed that computing a cost allocation in the least core of minimum-cost spanning tree games is NP-hard. Kern and Paulusma [31] presented a polynomial description of the linear program (LC) for cardinality matching games. Faigle et al. [32] 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. Deng [33] 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.

Schulz and Uhan [9] studied the computational complexity of the least core value and least core of supermodular cost cooperative games. They showed that the problem of computing the least core value of these games is strongly NP-hard, and in fact, is inapproximable within a factor of $17/16 - \varepsilon$ for any $\varepsilon > 0$ unless P = NP. In addition, they studied scheduling games, a subclass of supermodular cost cooperative games. For these games, they showed that the Shapley value,[5] which is polynomial-time computable in this case, is also in the least core. They also showed that computing the least core value of scheduling games is still NP-hard.

---

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

[5] The Shapley value [34] of a cooperative game (N, v) is the cost allocation $\phi \in \mathbb{R}^N$, where

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

In words, the Shapley value of each agent i reflects agent i's average marginal contribution to the coalition N. The Shapley value is a classic, well-studied solution concept in cooperative game theory; for example, see [35].

1.4. Contributions of this work

Although computing the least core value of supermodular cost cooperative games is NP-hard, it may be the case that acceptable approximations are computable in polynomial time. In this work, we propose approximations of the least core value and accompanying approximate least core cost allocations of supermodular cost cooperative games that are computable in polynomial time. In Section 2, we design approximation algorithms for computing the least core value of supermodular cost cooperative games, using oracles that determine coalitions whose dissatisfaction is approximately maximum. We also show how to compute accompanying approximate least core cost allocations. In Section 3, we apply our framework to computing the least core value of *scheduling games*, for which we give a fully polynomial time approximation scheme. Finally, in Section 4, we consider *matroid profit games*: a class of cooperative games with submodular profits. Using the framework established in Section 2 with the appropriate natural modifications, we show that the least core value and a cost allocation in the least core of these games can be computed in polynomial time.

2. Approximating the least core value and the least core

Before we begin, 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 polynomial-time value-giving oracle for v. In addition, for the remainder of the paper, we assume that there are at least two agents ($n \geq 2$).

In this section, our objectives are two-fold. First, we aim to approximate the least core value – the optimal value of (LC) – of supermodular cost cooperative games in polynomial time. Second, we seek to approximate a cost allocation in the least core – an optimal solution to (LC) – of these games in polynomial time. What do we mean by an approximate least core cost allocation? Suppose z^* is the least core value of a cooperative game (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,

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

Recall that a cost allocation in the least core of (N, v) can be seen as 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.

Note that one algorithm may not necessarily accomplish these objectives in tandem: for example, a polynomial-time algorithm for computing a cost allocation in the ρ-approximate least core of a cooperative game is not necessarily also a ρ-approximation algorithm[6] for computing the least core value of the same game. In fact, the computational complexity of these two objectives may be different: as we mentioned in the introduction, computing a cost allocation in the least core of scheduling games can be accomplished in polynomial time, while computing the least core value of these games is NP-hard [9].

2.1. Approximation by fixing a cost allocation

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). 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 coalition S^* whose dissatisfaction is maximum:*
>
> $$e(x, S^*) = \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} e(x, S) = \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \{x(S) - v(S)\}.$$

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

[6] A 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. The parameter ρ is known as the performance guarantee of the algorithm.

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

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

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 [36]. It turns out that any cost allocation x in B_v is in the 2-approximate least core of (N, v).

Theorem 2.1. *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, the cost allocation x is in 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). Since we have $x^*(S) \leq v(S) + z^*$ and $x^*(N \setminus S) \leq v(N \setminus S) + z^*$ for any $S \subseteq N, S \neq \emptyset, N$, and $x^*(N) = v(N)$, it follows 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 $2z^* \geq e(x, S^*)$. □

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 2.2. *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 the least core value z^* 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 2.1, it follows that $z = \rho e(x, \bar{S}) \leq \rho e(x, S^*) \leq 2\rho z^*$. □

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. Buchbinder et al. [37] gave a deterministic 3-approximation algorithm and a randomized 2-approximation algorithm for maximizing nonnegative submodular functions. With Theorem 2.2, this immediately implies the following corollary.

Corollary 2.3. *Suppose (N, v) is a supermodular cost cooperative game. Then, there exists a deterministic 6-approximation algorithm and a randomized 4-approximation algorithm for computing the least core value of (N, v).*

Given that any cost allocation in B_v is in the 2-approximate least core of a supermodular cost cooperative game (N, v), one might wonder if there always exists a cost allocation that is both in B_v and the least core of (N, v). The following example shows that this is not the case.

Example 2.4. Consider the following cooperative game (N, v) with 5 players:

$$\begin{array}{llll}
v(\{1,2,3,4,5\}) = 22, & v(\{1,2,5\}) = 12, & v(\{1,2\}) = 10, & v(\{3,5\}) = 3,\\
v(\{1,2,3,4\}) = 15, & v(\{1,3,4\}) = 6, & v(\{1,3\}) = 2, & v(\{4,5\}) = 6,\\
v(\{1,2,3,5\}) = 15, & v(\{1,3,5\}) = 6, & v(\{1,4\}) = 5, & v(\{1\}) = 1,\\
v(\{1,2,4,5\}) = 17, & v(\{1,4,5\}) = 8, & v(\{1,5\}) = 3, & v(\{2\}) = 7,\\
v(\{1,3,4,5\}) = 13, & v(\{2,3,4\}) = 12, & v(\{2,3\}) = 8, & v(\{3\}) = 1,\\
v(\{2,3,4,5\}) = 17, & v(\{2,3,5\}) = 10, & v(\{2,4\}) = 11, & v(\{4\}) = 4,\\
v(\{1,2,3\}) = 11, & v(\{2,4,5\}) = 13, & v(\{2,5\}) = 9, & v(\{5\}) = 2,\\
v(\{1,2,4\}) = 14, & v(\{3,4,5\}) = 10, & v(\{3,4\}) = 5, & v(\emptyset) = 0.
\end{array}$$

It turns out that v is supermodular, and the least core value of (N, v) is 19/5. However, when we add the constraints $x(S) \geq v(S)$ for all $S \subseteq N$ to (LC), the optimal value of this augmented linear program is $31/8 > 19/5$. Therefore, for this cooperative game (N, v), there does not exist a cost allocation in B_v that is also in the least core. □

2.2. Approximation without fixing a cost allocation

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 the following theorem.

Theorem 2.5. *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 $x(N) = v(N)$. 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.

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

> **Exact separation problem for polytope 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 $c^\mathsf{T} y > c^\mathsf{T} x$ for all $x \in K$.*

The exact non-emptiness problem for K is:

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

Grötschel et al. [38] showed that for a polytope K, by using the ellipsoid method, a polynomial-time algorithm for the exact separation problem for K implies a polynomial-time algorithm for the exact non-emptiness problem for K.

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 polytope 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 $c^\mathsf{T} y > c^\mathsf{T} x$ for all $x \in K$.*

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

> **Approximate non-emptiness problem for polytope 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 [38], 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. For the following steps in detail, see Appendix A. 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 $Q_{\bar{\gamma}-\varepsilon}$ 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

[7] Note 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 $x(N) = v(N)$.

(the parameter $\varepsilon \in \mathbb{Q}_{>0}$ is the precision required, and $\log \varepsilon^{-1}$ 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 $\log v(N)$. □

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. [39–42]). 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 [40] considered polytopes of the form $K = \{x \in \mathbb{R}^n : Ax \leq b\} \cap B$ with approximations $\bar{K} = \{x \in \mathbb{R}^n : Ax \leq \alpha b\} \cap B$ for some $\alpha \geq 1$, where $n, m \in \mathbb{Z}_{>0}$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m_{>0}$, and B is an arbitrary polynomial-time-separable polytope. 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 2.5, 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 2.5 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 2.1, the x-maximum dissatisfaction problem for a supermodular cost cooperative game (N, v) is an instance of submodular function maximization. Unlike in Section 2.1, however, the objective functions for the instances of the x-maximum dissatisfaction problem that need to be solved for Theorem 2.5 are not necessarily nonnegative. As mentioned previously, Buchbinder et al. [37] designed a deterministic 3-approximation algorithm for maximizing nonnegative submodular functions. A straightforward examination reveals that their analysis of this algorithm holds even if the submodular function $f : 2^N \to \mathbb{R}$ to be maximized is not nonnegative, but only satisfies $f(\emptyset) \geq 0$ and $f(N) \geq 0$. 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 2.6. *Suppose (N, v) is a supermodular cost cooperative game.*

(a) *there exists a deterministic 3-approximation algorithm for computing the least core value of (N, v), and*
(b) *there exists a deterministic polynomial-time algorithm for computing a cost allocation in the 3-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 whose performance guarantee compares favorably to those of the fixed-cost-allocation-based approximation algorithms discussed in Section 2.1. Interestingly, however, the comparison for the accompanying cost allocations of these approximation algorithms is reversed: the cost allocation computed by the ellipsoid-method-based approximation algorithm is guaranteed to be in the 3-approximate least core, while the cost allocation used in the fixed-cost-allocation-based approximation algorithms is guaranteed to be in the 2-approximate least core, and can even be computed without the ellipsoid method or an algorithm for submodular function maximization.

3. A special case from single-machine scheduling

In this section, we study the least core value of a particular supermodular cost cooperative game that arises from scheduling situations. Consider a setting where each agent has 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 all the agents to cooperate.

Scheduling games have been studied by several authors. For example, Maniquet [7] and Mishra and Rangarajan [8] gave axiomatic characterizations of various cost sharing rules for these games. Mishra and Rangarajan [8] also gave a simplified characterization of the Shapley value of scheduling games that implies it is computable in polynomial time. Schulz and Uhan [9], using a result of Wolsey [43] and Queyranne [44], showed that scheduling games are indeed supermodular cost cooperative games, and that computing the least core value of these games is still NP-hard. In addition, Schulz and Uhan [9] showed that for scheduling games, the Shapley value is a least core cost allocation.

We will apply the results of Section 2.1, in which approximation is based on fixing a cost allocation, to finding the least core value of scheduling games. Before we begin, we restate some results from [9], which we will use later on. To simplify the analysis, for the remainder of this section we assume without loss of generality that

$$\frac{w_1}{p_1} \geq \cdots \geq \frac{w_n}{p_n};$$

in other words, the agents' jobs are indexed in the order that minimizes the sum of weighted completion times [45].

Theorem 3.1 ([9]). *Suppose (N, v) is a scheduling game. Consider the cost allocation $\bar{x}$ defined as*

$$\bar{x}_i = \frac{1}{2} w_i \sum_{j=1}^{i} p_j + \frac{1}{2} p_i \sum_{j=i}^{n} w_j \quad \text{for all } i = 1, \ldots, n. \tag{3.1}$$

(a) *For any $S \subseteq N$,*

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

(b) *The cost allocation $\bar{x}$ is in the least core of (N, v).*

It is straightforward to show that $\bar{x} \in B_v$; in fact, it is a convex combination of two vertices of B_v. So, by Theorem 2.2, a ρ-approximation algorithm for the $\bar{x}$-maximum dissatisfaction problem implies a 2ρ-approximation algorithm for computing the least core value of (N, v). However, we can do better, since the cost allocation $\bar{x}$ is in the least core, and therefore

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

As a result, we obtain the following strengthening of Theorem 2.2.

Theorem 3.2. *Suppose there exists a ρ-approximation algorithm for the $\bar{x}$-maximum dissatisfaction problem for scheduling games. Then there exists a ρ-approximation algorithm for computing the least core value of scheduling games.*

By Theorem 3.1(a), we know that $\bar{x}$-MD is in fact

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

Therefore, 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 [46], and has a fully polynomial time approximation scheme (FPTAS) [47]. Although the two problems are equivalent from the optimization perspective, it is not possible to use the FPTAS for the two-machine scheduling problem directly as an FPTAS for the $\bar{x}$-MD problem: there exists an instance in which the precision required to use the FPTAS for the two-machine problem as an FPTAS for the $\bar{x}$-MD problem results in a running time exponential in the input size [48]. Nevertheless, as we show in the next theorem, an FPTAS for the $\bar{x}$-MD problem is indeed possible. The analysis uses standard techniques for designing approximation schemes (for example, see [49]).

Theorem 3.3. *There exists a fully polynomial time approximation scheme for the $\bar{x}$-maximum dissatisfaction problem for scheduling games.*

Proof. 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. $1, \ldots, n$). 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, $E_1, \ldots, E_n$. A schedule σ for jobs $\{1, \ldots, k\}$ 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: $v(\{1, \ldots, k\})$ *minus* the sum of weighted completion times on two machines for σ.

Suppose jobs $1, \ldots, k-1$ 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 *exact dynamic program*.

Input: scheduling game (N, v) with weights w_i, processing times p_i for all $i \in N$.
Output: $g(S^*) = \max\{g(S) : S \subseteq N, S \neq \emptyset, N\}$.

$E_1 = \{(p_1, 0, 0), (0, p_1, 0)\}$
For $k = 2, \ldots, n$
  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 $g(S^*) = c^*$.

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

Let $\delta = (1+\varepsilon/(2n))^{-1}$ for some $\varepsilon \in (0, 1)$. 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 $(\delta^{-r}, \delta^{-s}, \delta^{-t})$ for all $r = 1, \ldots, L$, $s = 1, \ldots, L$, and $t = 1, \ldots, M$. For each $k = 1, \ldots, n$, we divide the state set E_k into the boxes formed by the grid:

$$\{(a, b, c) \in \mathbb{R}^3 : \delta^{-r+1} \le a \le \delta^{-r},\ \delta^{-s+1} \le b \le \delta^{-s},\ \delta^{-t+1} \le c \le \delta^{-t}\}$$
$$\text{for all } r = 1, \ldots, L,\ s = 1, \ldots, L, \text{ and } t = 1, \ldots, M.$$

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

$$\delta a_1 \le a_2 \le \frac{a_1}{\delta}, \qquad \delta b_1 \le b_2 \le \frac{b_1}{\delta}, \qquad \delta c_1 \le c_2 \le \frac{c_1}{\delta}. \tag{3.2}$$

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 following *trimmed dynamic program* is based on this simplified state space.

Input: scheduling game (N, v) with weights w_i, processing times p_i for all $i \in N$, precision $\varepsilon \in (0, 1)$.
Output: an approximation $g(\bar{S})$ to $\max\{g(S) : S \subseteq N, S \neq \emptyset, N\}$.

$\delta = (1 + \varepsilon/(2n))^{-1}$
$E_1^\delta = \{(p_1, 0, 0), (0, p_1, 0)\}$
For $k = 2, \ldots, n$
  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 $g(\bar{S}) = \bar{c}$.

The key observation is that every element in the state space of the exact dynamic program is relatively close to an element in the state space of the trimmed dynamic program. In particular,

$$\text{For every } (a, b, c) \in E_k, \text{ there exists a vector } (a', b', c') \in E_k^\delta \text{ such that } a' \ge \delta^k a,\ b' \ge \delta^k b, \text{ and } c' \ge \delta^k c. \tag{3.3}$$

We show (3.3) by induction. The base case $k = 1$ holds by (3.2). Assume the induction hypothesis holds for $1, \ldots, k-1$. 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 $\alpha' \ge \delta^{k-1}\alpha$, $\beta' \ge \delta^{k-1}\beta$, and $\gamma' \ge \delta^{k-1}\gamma$. In the kth 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

$$a' \ge \delta(\alpha' + p_k) \ge \delta^k\alpha + \delta p_k \ge \delta^k(\alpha + p_k) = \delta^k a,$$
$$b' \ge \delta\beta' \ge \delta^k\beta = \delta^k b,$$
$$c' \ge \delta(\gamma' + w_k\beta') \ge \delta^k\gamma + \delta^k w_k\beta = \delta^k(\gamma + w_k\beta) = \delta^k c.$$

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, and (3.3) holds.

We now analyze the performance of the trimmed dynamic program. Let $c^* = g(S^*)$, the optimal value of g. Note that there exists a vector $(a^*, b^*, c^*) \in E_n$. By (3.3), there exists a vector $(a', b', c') \in E_n^\delta$ such that $c' \ge \delta^n c^*$. Recall that $\delta = (1 + \varepsilon/(2n))^{-1}$ for some $\varepsilon \in (0, 1)$. Since $(1 + \varepsilon/(2n))^n \le 1 + \varepsilon$, we have that $c' \ge (1 + \varepsilon/(2n))^{-n} c^* \ge (1 + \varepsilon)^{-1} c^*$.

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

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

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

Combining Theorems 3.2 and 3.3, gives us the following result.

Theorem 3.4. *There exists a fully polynomial time approximation scheme for computing the least core value of scheduling games.*

4. 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

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

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

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

The least core value of (N, v) is the optimal value z^* to this linear program. Note that it still reflects the minimum penalty we need to charge a coalition for acting independently in order 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 2 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 $e(x, S^*) = \max_{S \subseteq N, S \neq \emptyset, N} e(x, S)$.

4.1. Matroid profit games

Consider the following example of a profit cooperative game (N, v). 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 [50]. 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 coalition $S \subseteq N$.

In this section, we study a generalization of the cooperative game described above, defined as follows. 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 coalition $S \subseteq N$. 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. [51] 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 2 do not apply.

Throughout this section, we assume that the matroid $(N, \mathcal{I})$ and its restrictions are given by an independence oracle that asserts whether or not a given subset $S \subseteq N$ belongs to $\mathcal{I}$. It is well known that v as defined here is a submodular function (for example, see p. 715 in [52]), and so 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 4.1. *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 $A = \{i \in N : x_i < 0\}$. Consider the following algorithm for the x-maximum dissatisfaction problem for (N, v):

Input: matroid profit game (N, v) with matroid $(N, \mathcal{I})$ and weights $w_i \in \mathbb{R}_{\geq 0}$ for all $i \in N$.
Output: an optimal solution $\bar{S}$ to x-MD for (N, v).

1. Compute a maximum $\bar{w}$-weight independent set T^* of $(N, \mathcal{I})$, where
$$\bar{w}_i = \begin{cases} w_i & \text{if } i \in A \\ w_i - x_i & \text{if } i \in N \setminus A. \end{cases}$$
2. Let $\bar{T} = T^* \cup A$.
   - If $\bar{T} \neq \emptyset, N$, output $\bar{S} = \bar{T}$.
   - Otherwise, output $\bar{S} = \arg\max\{e(x, S) : S \in \{T, N \setminus T\}\}$ for some $T \subseteq N, T \neq \emptyset, N$.

First, since v is submodular, we have that $e(x, S) + e(x, N \setminus S) = v(S) + v(N \setminus S) - v(N) \geq 0$ for all $S \subseteq N$ such that $S \neq \emptyset, N$, and so

$$e(x, S) \geq 0 \quad \text{or} \quad e(x, N \setminus S) \geq 0 \quad \text{or both for all } S \subseteq N, S \neq \emptyset, N. \tag{4.1}$$

In addition, we have that

$$e(x, \emptyset) = e(x, N) = 0. \tag{4.2}$$

Therefore,

$$\max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} e(x, S) = \underbrace{\max_{S \subseteq N} e(x, S)}_{\text{(i)}} \geq 0. \tag{4.3}$$

Next, since v is nondecreasing and $x_i < 0$ for all $i \in A$, any optimal solution to (i) must contain all elements of A, and so

$$\max_{S \subseteq N} e(x, S) = \max_{S \subseteq N} \{v(S) - x(S)\} = \underbrace{\max_{S \subseteq N} \{v(S) - x(S \setminus A)\}}_{\text{(ii)}} - x(A). \tag{4.4}$$

We show that the independent set T^* computed in Step 1 of the above algorithm is an optimal solution to (ii). Let S^* be an optimal solution to (ii), and suppose that $v(S^*) - x(S^* \setminus A) > v(T^*) - x(T^* \setminus A)$. Note that without loss of generality, S^* is an independent set of $(N, \mathcal{I})$. Otherwise, there exists some $i \in S^*$ that is not in a maximum weight independent set of $(S^*, \mathcal{I}|S^*)$. Take such i. If $x_i \geq 0$, then $i \in S^* \setminus A$ can be removed without decreasing the objective value of (ii); if $x_i < 0$, then $i \in A$, and removing it does not affect the objective value of (ii). Therefore,

$$\begin{aligned} \bar{w}(S^*) &= w(S^*) - x(S^* \setminus A) = v(S^*) - x(S^* \setminus A) \\ &> v(T^*) - x(T^* \setminus A) = w(T^*) - x(T^* \setminus A) = \bar{w}(T^*), \end{aligned}$$

which contradicts the assumption that T^* is a maximum $\bar{w}$-weight independent set of $(N, \mathcal{I})$. So T^* is an optimal solution to (ii), and therefore an optimal solution to (i). By (4.1)–(4.3), it follows that the output $\bar{T}$ of the above algorithm is an optimal solution to x-MD for (N, v).

Since a maximum weight independent set of a matroid can be found in polynomial time [53,54], it follows that the above algorithm solves the x-maximum dissatisfaction problem for a matroid profit game in polynomial time. □

By the appropriate analogue of Theorem 2.5, 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). Therefore, by Theorem 4.1, we immediately obtain the following theorem.

Theorem 4.2. *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 profit allocation in the least core of (N, v).*

5. Conclusion

We provided a general framework for approximating the least core value of supermodular cost cooperative games. Using this framework with the approximation algorithms for submodular function maximization of Buchbinder et al. [37], we obtained a 3-approximation algorithm for computing the least core value of supermodular cost cooperative games. We also showed 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 a fully polynomial-time approximation scheme for computing the least core value. For matroid profit games, we showed how to compute the least core value as well as a cost allocation in the least core in polynomial time.

There are several interesting directions for future research that extend from this work. 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) [30] offers a way to address this issue: it is the set of cost allocations x that are optimal solutions to the linear program

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

for some function $f : 2^N \to \mathbb{R}$. Several forms of the f-least core have been considered in the literature for various cooperative games, including $f(S) = |S|$ for all $S \subseteq N$ [19], and $f(S) = v(S)$ for all $S \subseteq N$ [55]. 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 approximate the nucleolus of supermodular cost cooperative games. Last, but not least, the computational complexity of computing a cost allocation in the least core of supermodular cost cooperative games remains open.

Acknowledgment

This research was supported by the National Science Foundation (DMI-0426686).

Appendix. Proof of Theorem 2.5

We begin in Appendix A.1 by establishing some definitions from polyhedral theory. Then, in Appendix A.2, 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 2.5. Finally, in Appendix A.3, we show how to use the results from Appendix A.2 to approximately solve the least core linear program (LC). The ideas here closely follow and generalize the analyses found in [38,40].

A.1. 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

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

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

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

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

Lemma A.1 ([38, 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 ν, then K has facet complexity at most $3n^2\nu$.*

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.2. 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 polytope 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 $c^{\mathsf{T}}y > c^{\mathsf{T}}x$ 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: $\varepsilon \in \mathbb{Q}$ such that $\varepsilon \in (0, 1)$, bounded rational polyhedron $K \subseteq \mathbb{R}^n$ given by an oracle for S-APP-SEP, $R \in \mathbb{Q}$ such that $K \subseteq E(R^2I, 0)$ (where I denotes the identity matrix).
Output: either (i) $y \in \bar{K}$, or (ii) positive definite $A \in \mathbb{Q}^{n\times n}$, $a \in \mathbb{Q}^n$ such that $K \subseteq E(A, a)$ and $\text{vol}(E(A, a)) \leq \varepsilon$.

1. Set the following values:

$$N = \lceil 5n|\log\varepsilon| + 5n^2|\log 2R|\rceil, \qquad p = 8N. \tag{A.1}$$

2. Generate the sequence of ellipsoids $E(A_0, a_0), E(A_1, a_1), \ldots, E(A_N, a_N)$ as follows:
   - Initialize the sequence:
$$a_0 = 0, \qquad A_0 = R^2 I. \tag{A.2}$$
   - For $k = 0, \ldots, N-1$, 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
$$\|c_k\|_\infty = 1, \qquad c_k^{\mathsf{T}} a_k > c_k^{\mathsf{T}} x \quad \text{for all } x \in K, \tag{A.3}$$
     then compute
$$a_{k+1} \approx a_k - \frac{1}{n+1}\frac{A_k c_k}{\sqrt{c_k^{\mathsf{T}} A_k c_k}} \tag{A.4}$$
$$A_{k+1} \approx \frac{2n^2+3}{2n^2}\left(A_k - \frac{2}{n+1}\frac{A_k c_k c_k^{\mathsf{T}} A_k}{c_k^{\mathsf{T}} A_k c_k}\right) \tag{A.5}$$
     where "$\approx$" means the computations are done with p digits after the binary point.
   - If $k = N$, return a_N and A_N. Stop.

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

Lemma A.3 (*[38, 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). Suppose A_k and a_k ($k = 0, 1, \ldots, N$) are defined as in (A.2) and (A.4)–(A.5), and c_k ($k = 0, 1, \ldots, N$) satisfy (A.3). Then, the following statements hold for $k = 0, 1, \ldots, N$:*

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

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 ($k = 0, 1, \ldots, N$) as defined in (A.2) and (A.4)–(A.5) 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

$$\mathrm{vol}(E(A_N, a_N)) \le e^{-\frac{N}{5n}}\mathrm{vol}(E(A_0, a_0)) \le e^{-\frac{N}{5n}}(2R)^n < 2^{-\frac{N}{5n}}(2R)^n \le \varepsilon.$$

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

Before proceeding further, we need the following lemma.

Lemma A.5 (*[38, pp. 175–176]*). *Let $(K; n, \varphi)$ be a well-described polyhedron. In addition, let $\varepsilon = 2^{-48n^5\varphi}$. Suppose $K \subseteq E(A, a)$ where $\mathrm{vol}(E(A, a)) \le \varepsilon$. Then there exists $f \in \mathbb{Z}^n$ and $g \in \mathbb{Z}_{>0}$ such that $f \ne 0$ and $K \subseteq \{x \in \mathbb{R}^n : f^{\mathsf{T}} x = g\}$. 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 polytope 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 appendix: 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 $\varepsilon = 2^{-48n^5\varphi}$. 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 $\text{vol}(E) \leq \varepsilon$. 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 $K \subseteq \{x \in \mathbb{R}^n : (f^1)^\mathsf{T} x = g^1\}$. Without loss of generality, assume that $f^1_1 \neq 0$.

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

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

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

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

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

$$K \subseteq \left\{x \in \mathbb{R}^n : (f^1)^\mathsf{T} x = g^1, \ldots, (f^k)^\mathsf{T} x = g^k, (f^{k+1})^\mathsf{T} x = g^{k+1}\right\}.$$

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

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

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

$$\begin{pmatrix} F_1^{-1}g - F_1^{-1}F_2 w \\ w \end{pmatrix} \in K.$$

Note that for any vertex u^* of K_k, there exists $z^* \in \mathbb{R}^k$ such that $\binom{z^*}{u^*}$ 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 $\varepsilon = 2^{-48n^5\varphi'}$, using the following modified approximate separation oracle for K_k:

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

Output: either (i) assert $y \in \bar{K}$, where

$$y = \begin{pmatrix} F_1^{-1}g - F_1^{-1}F_2 w \\ w \end{pmatrix}$$

or (ii) find $\bar{c} \in \mathbb{Q}^{n-k}$ such that $\|\bar{c}\|_\infty = 1$ and $\bar{c}^\mathsf{T} w > \bar{c}^\mathsf{T} u$ for all $u \in K_k$.

1. Apply S-APP-SEP oracle for K on
$$y = \begin{pmatrix} F_1^{-1}g - F_1^{-1}F_2 w \\ w \end{pmatrix}$$
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 $c^\mathsf{T} y > c^\mathsf{T} x$ for all $x \in K$. Let $c^1 \in \mathbb{Q}^k$ and $c^2 \in \mathbb{Q}^{n-k}$ such that $c = \binom{c^1}{c^2}$. Therefore,
$$(c^1)^\mathsf{T}(F_1^{-1}g - F_1^{-1}F_2 w) + (c^2)^\mathsf{T} w > (c^1)^\mathsf{T}(F_1^{-1}g - F_1^{-1}F_2 u) + (c^2)^\mathsf{T} u \quad \text{for all } u \in K^k,$$
or equivalently,
$$((c^2)^\mathsf{T} - (c^1)^\mathsf{T}F_1^{-1}F_2)w > ((c^2)^\mathsf{T} - (c^1)^\mathsf{T}F_1^{-1}F_2)u \quad \text{for all } u \in K^k.$$
Return
$$\bar{c} = \frac{c^2 - (F_1^{-1}F_2)^\mathsf{T} c^1}{\|c^2 - (F_1^{-1}F_2)^\mathsf{T} c^1\|_\infty}$$
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 $\text{vol}(E^k) \leq \varepsilon$. 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 $(f^{k+1})^\mathsf{T} = (0 \ \cdots \ 0 \ \bar{f}^{k+1})^\mathsf{T}$. It follows that $K \subseteq \{x \in \mathbb{R}^n : (f^{k+1})^\mathsf{T} x = g^{k+1}\}$. Therefore,

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

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

When $k = n$, we have that

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

Since $f^1, \ldots, f^n$ are linearly independent, K must be equal to the unique vector y in $\{x \in \mathbb{R}^n : (f^1)^\mathsf{T} x = g^1, \ldots, (f^n)^\mathsf{T} x = g^n\}$, 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 Lemmas A.3(b) and A.5, we can find the vectors $f^1, \ldots, f^n$ and the scalars $g^1, \ldots, g^n$ 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. □

A.3. Approximately solving the least core optimization problem

For a cooperative game (N, v), 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

$$Q_\gamma = \{x \in \mathbb{R}^N : x(N) = v(N), \ x(S) \leq v(S) + \gamma \text{ for all } S \subseteq N, S \neq \emptyset, N\}. \tag{A.6}$$

We restate the strong approximate separation problem and approximate non-emptiness problem for polytopes of the form Q_γ, using $Q_{\rho\gamma}$ as its approximation:

Strong approximate separation problem for Q_γ of cooperative game (N, v) 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 of cooperative game (N, v) 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 $\log v(N)$. □

Finally, we are ready to prove Theorem 2.5. 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 2.5. 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:
$$m = 4(n+1)^2(2(n+1) + \lceil \log(v(N)+1) \rceil + 1), \tag{A.7a}$$
$$M = 2^m, \tag{A.7b}$$
$$\varepsilon = (2M)^{-2}. \tag{A.7c}$$
2. Using $\mathcal{A}$, find $\bar{\gamma} \in \mathbb{Q}$ by binary search on $[0, v(N)]$ such that $Q_{\bar{\gamma}-\varepsilon}$ 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
$$1 \leq q \leq 2M \quad \text{and} \quad \left| \bar{\gamma} - \frac{p}{q} \right| < \frac{1}{2Mq}. \tag{A.8}$$
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 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 $\lceil \log \frac{v(N)}{\varepsilon} \rceil + 1$ iterations. This implies that the denominator of any trial value of $\bar{\gamma}$ is at most
$$2^{\left\lceil \log \frac{v(N)}{\varepsilon} \right\rceil + 1} \leq 2^{2+\log \frac{v(N)}{\varepsilon}} = \frac{4v(N)}{\varepsilon}.$$
Since the binary search is performed on the interval $[0, v(N)]$, the numerator of any trial value of $\bar{\gamma}$ is at most $4v(N)^2/\varepsilon$. By (A.7a)–(A.7c), 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.8), and since $M \geq 1$ and $\bar{\gamma} \in [0, v(N)]$, we have that
$$p < \bar{\gamma}q + \frac{1}{2M} \leq 2Mv(N) + 1,$$
$$p > \bar{\gamma}q - \frac{1}{2M} \geq -\frac{1}{2}.$$
Therefore, $|p| < 2Mv(N) + 1$. Since $|q| \leq 2M$, the claim follows by (A.7a)–(A.7c).

Next, we analyze the running time of the above algorithm. The algorithm makes a total of $O(\log \frac{v(N)}{\varepsilon})$ calls to $\mathcal{A}$, which runs in time polynomial in n and $\log v(N)$ each time it is called. It follows by (A.7a)–(A.7c) 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 [38, pp. 134–137], finding integers p and q to satisfy (A.8) 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. $\bar{\gamma} - \varepsilon < z^* < \bar{\gamma}$. 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
$$\bar{\gamma} - \frac{r}{s} = \bar{\gamma} - z^* < \varepsilon = \frac{1}{(2M)^2} \leq \frac{1}{2Mq},$$
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 $z \leq \rho z^*$. □

For submodular profit cooperative games (described in Section 4), the results of Appendix A.3 still hold, with the natural modifications. There is one issue that needs careful consideration. When v is nonnegative and supermodular with $v(\emptyset) = 0$, v is also nondecreasing. As a result, $v(N)$ is an upper bound on the least core value of a supermodular cost cooperative game, and the encoding length of $v(N)$ is an upper bound on the encoding length of $v(S)$ for any $S \subseteq N$. However, this is no longer the case when v is nonnegative and submodular with $v(\emptyset) = 0$; in this case, we have that $v(S) \leq \sum_{i \in N} v(\{i\})$ for all $S \subseteq N$. Therefore, $\sum_{i \in N} v(\{i\})$ is an upper bound on the least core value of a submodular profit cooperative game, and the encoding length of $\sum_{i \in N} v(\{i\})$ is an upper bound on the encoding length of $v(S)$ for any $S \subseteq N$.

References

[1] Y. Gerchak, D. Gupta, On apportioning costs to customers in centralized continuous review inventory systems, Journal of Operations Management 10 (4) (1991) 546–551.
[2] B. Hartman, M. Dror, M. Shaked, Cores of inventory centralization games, Games and Economic Behavior 31 (1) (2000) 26–49.
[3] W. van den Heuvel, P. Borm, H. Hamers, Economic lot-sizing games, European Journal of Operational Research 176 (2) (2007) 1117–1130.
[4] X. Chen, J. Zhang, Duality approaches to economic lot sizing games, IOMS: Operations Management Working Paper OM-2006-01, Stern School of Business, New York University, 2006.
[5] M. Gopaladesikan, N.A. Uhan, J. Zou, A primal–dual algorithm for computing a cost allocation in the core of economic lot-sizing games, Operations Research Letters 40 (6) (2012) 453–458.
[6] I. Curiel, G. Pederzoli, S. Tijs, Sequencing games, European Journal of Operational Research 40 (3) (1989) 344–351.
[7] F. Maniquet, A characterization of the Shapley value in queueing problems, Journal of Economic Theory 109 (1) (2003) 90–103.
[8] D. Mishra, B. Rangarajan, Cost sharing in a job scheduling problem, Social Choice and Welfare 29 (3) (2007) 369–382.
[9] A.S. Schulz, N.A. Uhan, Sharing supermodular costs, Operations Research 58 (4) (2010) 1051–1056.
[10] L.S. Shapley, M. Shubik, The assignment game I: the core, International Journal of Game Theory 1 (1) (1971) 111–130.
[11] G. Owen, On the core of linear production games, Mathematical Programming 9 (1) (1975) 358–370.
[12] C.G. Bird, Cost-allocation for a spanning tree, Networks 6 (4) (1976) 335–350.
[13] D. Granot, G. Huberman, Minimum cost spanning tree games, Mathematical Programming 21 (1) (1981) 1–18.
[14] E. Kalai, E. Zemel, Generalized network problems yielding totally balanced games, Operations Research 30 (5) (1982) 998–1008.
[15] E. Kalai, E. Zemel, Totally balanced games and games of flow, Mathematics of Operations Research 7 (3) (1982) 476–478.
[16] J. Potters, I. Curiel, S. Tijs, Traveling salesman games, Mathematical Programming 53 (1–3) (1991) 199–211.
[17] M.X. Goemans, M. Skutella, Cooperative facility location games, Journal of Algorithms 50 (2) (2004) 194–214.
[18] D.B. Gillies, Solutions to general non-zero-sum games, in: A.W. Tucker, R.D. Luce (Eds.), Contributions to the Theory of Games, Volume IV, in: Annals of Mathematics Studies, vol. 40, Princeton University Press, Princeton, 1959, pp. 47–85.
[19] L.S. Shapley, M. Shubik, Quasi-cores in a monetary economy with nonconvex preferences, Econometrica 34 (4) (1966) 805–827.
[20] M. Maschler, B. Peleg, L.S. Shapley, Geometric properties of the kernel, nucleolus, and related solution concepts, Mathematics of Operations Research 4 (4) (1979) 303–338.
[21] D. Schmeidler, The nucleolus of a characteristic function game, SIAM Journal on Applied Mathematics 17 (6) (1969) 1163–1170.
[22] E. Einy, R. Holzman, D. Monderer, On the least core and the Mas-Colell bargaining set, Games and Economic Behavior 28 (2) (1999) 181–188.
[23] A. Mas-Colell, An equivalence theorem for a bargaining set, Journal of Mathematical Economics 18 (2) (1989) 129–139.
[24] R.J. Aumann, M. Maschler, The bargaining set for cooperative games, in: M. Dresher, L.S. Shapley, A.W. Tucker (Eds.), Advances in Game Theory, in: Annals of Mathematics Studies, vol. 52, Princeton University Press, Princeton, 1964, pp. 443–476.
[25] L.S. Shapley, Cores of convex games, International Journal of Game Theory 1 (1) (1971) 11–26.
[26] G.L. Nemhauser, L.A. Wolsey, M.L. Fisher, An analysis of algorithms for maximizing submodular set functions—I, Mathematical Programming 14 (1) (1978) 265–294.
[27] N. Meggido, Computational complexity of the game theory approach to cost allocation for a tree, Mathematics of Operations Research 3 (3) (1978) 189–196.
[28] X. Deng, C.H. Papadimitriou, On the complexity of cooperative solution concepts, Mathematics of Operations Research 19 (2) (1994) 257–266.
[29] H.A. Simon, Theories of bounded rationality, in: C.B. McGuire, R. Radner (Eds.), Decision and Organization: A Volume in Honor of Jacob Marschak, North-Holland, Amsterdam, 1972, pp. 161–176.

[30] U. Faigle, W. Kern, D. Paulusma, Note on the computational complexity of least core concepts for min-cost spanning tree games, Mathematical Methods of Operations Research 52 (1) (2000) 23–38.
[31] W. Kern, D. Paulusma, Matching games: the least core and the nucleolus, Mathematics of Operations Research 28 (2) (2003) 294–308.
[32] U. Faigle, W. Kern, J. Kuipers, On the computation of the nucleolus of a cooperative game, International Journal of Game Theory 30 (1) (2001) 79–98.
[33] X. Deng, Combinatorial optimization and coalition games, in: D.-Z. Du, P.M. Pardalos (Eds.), Handbook of Combinatorial Optimization, Vol. 2, Kluwer Academic Publishers, Dordrecht, 1998, pp. 77–103.
[34] L.S. Shapley, A value for n-person games, in: H.W. Kuhn, A.W. Tucker (Eds.), Contributions to the Theory of Games, Volume II, in: Annals of Mathematics Studies, vol. 28, Princeton University Press, Princeton, 1953, pp. 307–317.
[35] A.E. Roth (Ed.), The Shapley Value: Essays in Honor of Lloyd S. Shapley, Cambridge University Press, Cambridge, 1988.
[36] J. Edmonds, Submodular functions, matroids, and certain polyhedra, in: R. Guy, H. Hanani, N. Sauer, J. Schönheim (Eds.), Combinatorial Structures and their Applications (Calgary International Conference on Combinatorial Structures and their Applications), Gordon and Breach, New York, 1970, pp. 69–87.
[37] N. Buchbinder, M. Feldman, J. Naor, R. Schwartz, A tight linear time (1/2)-approximation for unconstrained submodular maximization, in: Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, IEEE, 2012, pp. 649–658.
[38] M. Grötschel, L. Lovász, A. Schrijver, Geometric Algorithms and Combinatorial Optimization, Springer, Berlin, 1988.
[39] B. Carr, S. Vempala, Randomized metarounding, Random Structures and Algorithms 20 (3) (2002) 343–352.
[40] K. Jansen, Approximate strong separation with application in fractional graph coloring and preemptive scheduling, Theoretical Computer Science 302 (1–3) (2003) 239–256.
[41] K. Jain, M. Mahdian, M. Salavatipour, Packing Steiner trees, in: Proceedings of the 14th Annual ACM–SIAM Symposium on Discrete Algorithms, SODA 2003, SIAM, Philadelphia, 2003, pp. 266–274.
[42] L. Fleischer, M.X. Goemans, V.S. Mirrokni, M. Sviridenko, Tight approximation algorithms for maximum separable assignment problems, Mathematics of Operations Research 36 (3) (2011) 416–431.
[43] L.A. Wolsey, Mixed integer programming formulations for production planning and scheduling problems, in: Invited Talk at the 12th International Symposium on Mathematical Programming, MIT, Cambridge, MA, 1985.
[44] M. Queyranne, Structure of a simple scheduling polyhedron, Mathematical Programming 58 (1–3) (1993) 263–285.
[45] W.E. Smith, Various optimizers for single-stage production, Naval Research Logistics Quarterly 3 (1–2) (1956) 59–66.
[46] J. Bruno, E.G. Coffman, R. Sethi, Scheduling independent tasks to reduce mean finishing time, Communications of the ACM 17 (7) (1974) 382–387.
[47] S. Sahni, Algorithms for scheduling independent tasks, Journal of the ACM 23 (1) (1976) 116–127.
[48] N.A. Uhan, Algorithmic and game-theoretic perspectives on scheduling, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2008.
[49] P. Schuurman, G.J. Woeginger, Approximation schemes – a tutorial, Manuscript, http://www.win.tue.nl/~gwoegi/papers/ptas.pdf.
[50] H. Gabow, R. Tarjan, Efficient algorithms for a family of matroid intersection problems, Journal of Algorithms 5 (1) (1984) 80–131.
[51] H. Nagamochi, D.-Z. Zeng, N. Kabutoya, T. Ibaraki, Complexity of the minimum base game on matroids, Mathematics of Operations Research 22 (1) (1997) 146–164.
[52] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, Wiley, New York, NY, 1988.
[53] R. Rado, Note on independence functions, Proceedings of the London Mathematical Society s3–s7 (1) (1957) 300–320.
[54] J. Edmonds, Matroids and the greedy algorithm, Mathematical Programming 1 (1) (1971) 127–136.
[55] U. Faigle, W. Kern, On some approximately balanced combinatorial cooperative games, Mathematical Methods of Operations Research 38 (2) (1993) 141–152.