Encouraging Cooperation in Sharing Supermodular Costs

Andreas S. Schulz* Nelson A. Uhan**

August 27, 2007

Abstract

The least core value of a cooperative game is the minimum penalty we need to charge a coalition for defecting that ensures the existence of a fair and efficient cost allocation. The set of all such cost allocations is called the least core. In this paper, we study the computational complexity and algorithmic aspects of computing the least core value of supermodular cost cooperative games, and uncover some structural properties of the least core of these games. We motivate the study of these games by showing that a particular class of optimization problems has supermodular optimal costs. This class includes a variety of problems in combinatorial optimization, especially in machine scheduling. We show that computing the least core value of supermodular cost cooperative games is NP-hard, and build a framework to approximate the least core value of these games using oracles that approximately determine maximally violated constraints. With recent work on maximizing submodular functions, our framework yields a $(3+\epsilon)$-approximation algorithm for computing the least core value of general supermodular cost games.

We also apply our approximation framework to two particular classes of cooperative games: schedule planning games and matroid profit games. Schedule planning games are cooperative games in which the cost to a coalition is derived from the minimum sum of weighted completion times on a single machine. By specializing some of the results for general supermodular cost cooperative games, we are able to show that the Shapley value is an element of the least core of schedule planning games, and design a fully polynomial time approximation scheme for computing the least core value of these games. Matroid profit games are cooperative games with submodular profits: the profit to a coalition arises from the maximum weight of a matroid basis. We show that an element of the least core and the least core value of matroid profit games can be computed in polynomial time.

*Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139. e-mail: schulz@mit.edu

**Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139. e-mail: uhan@mit.edu

1 Introduction

Consider a situation where a set of agents agree to share the cost of their joint actions, and need to determine how to distribute the costs amongst themselves in a fair manner. For example, a set of agents may agree to process their jobs together on a machine, and share the cost of optimally scheduling their jobs. This kind of situation can be modeled naturally as a *cooperative game*. A cooperative game is a pair (N, v) where $N = \{1, \ldots, n\}$ represents a set of agents, and $v(S)$ represents the cost to agents in $S \subseteq N$.

In this paper, we are concerned with cooperative games (N, v) where v is nonnegative, *supermodular*, and $v(\emptyset) = 0$. We call such games *supermodular cost cooperative games.* A set function $v : 2^N \mapsto \mathbb{R}$ is supermodular if

$$v(S \cup \{j\}) - v(S) \leq v(S \cup \{j, k\}) - v(S \cup \{k\}) \qquad \text{for all } S \subseteq N \setminus \{j, k\}. \tag{1.1}$$

In words, supermodularity captures the notion of increasing marginal costs. Our primary motivation behind studying these games is that many problems from combinatorial optimization—especially in machine scheduling—have optimal costs that are supermodular. Cooperative games whose costs are determined by combinatorial optimization problems have been considered previously: these include assignment games (Shapley and Shubik 1971), minimum-cost spanning tree games (Granot and Huberman 1981), traveling salesman games (Potters et al. 1991), facility location games (Goemans and Skutella 2004), scheduling-related games (Curiel et al. 1989, Maniquet 2003, Mishra and Rangarajan 2005), and many others.

The central concern in cooperative game theory is the fair allocation of costs amongst agents. The prominent solution concept for cooperative games is the *core* (Gillies 1959). The core of a cooperative game (N, v) consists of all cost allocations x that distribute $v(N)$—the cost incurred when all agents cooperate—in a way such that no subset of agents has incentive to forsake the rest of the agents and act on its own behalf. Formally,

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

(For notational convenience, for any vector x we define $x(S) = \sum_{i \in S} x_i$ for any $S \subseteq N$.) It is well known that cooperative games with submodular[1] costs always have nonempty cores (Shapley 1971). This result is

[1] A set function $v : 2^N \mapsto \mathbb{R}$ is *submodular* if $-v$ is supermodular.

quite intuitive. As a coalition grows, the cost of adding a particular agent to the coalition decreases, making the idea of sharing costs more appealing. On the other hand, a supermodular cost cooperative game (N, v) has an empty core (as long as v is not modular[2]). Similar intuition still holds: the cost of adding a particular agent to a coalition increases as the coalition grows, diminishing the appeal of sharing costs.

When a cooperative game has an empty core, one might wonder if it is possible to allocate costs so that no subset of agents has incentive to deviate, and a fraction α of the total cost $v(N)$ can be recovered. This notion is captured in the *approximate* or α*-core* solution concept. Formally, for any $\alpha \in (0, 1]$,

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

The α-core has been studied for a variety of games (Faigle et al. 1998, Faigle and Kern 1998, Pál and Tardos 2003, Immorlica et al. 2005). Unfortunately, in supermodular cost cooperative games, the largest fraction α one could hope to recover under a fair allocation is $\sum_{i \in N} v(\{i\})/v(N)$, which may be arbitrarily small.

Since the prospect for cooperation in sharing supermodular costs is bleak, we are led to ask: how much do we need to penalize a coalition for defecting in order to encourage cooperation amongst all agents? This notion is captured in the *least core value* of a cooperative game. The *least core* of a cooperative game (N, v) is the set of cost allocations x that are optimal solutions to the *least core optimization problem*

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

The optimal value z^* of (LC) is the least core value[3] of the game (N, v). By computing the least core value, we gain insight into the value a coalition of agents places on the ability to act on their own. The least core solution concept was introduced by Shapley and Shubik (1966), and later named by Maschler, Peleg, and Shapley (1979). Computing an element in the least core has been studied in several contexts. Faigle, Kern, and Paulusma (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 least

[2] A set function is *modular* if it is submodular and supermodular.

[3] Adding the inequalities $x_i \leq v(\{i\}) + z$ for all $i \in N$ and using the equality $x(N) = v(N)$, we can bound z^* below by $(v(N) - \sum_{i \in N} v(\{i\}))/|N|$. So as long as costs are finite, the least core value is well defined. Moreover, if v is supermodular and $v(\emptyset) = 0$, then $z^* \geq 0$.

core optimization problem for cardinality matching games. Properties of the least core value, on the other hand, seem to have been largely ignored.

In this paper, we consider various theoretical aspects of computing the least core value of supermodular cost cooperative games. 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 problems and other combinatorial optimization 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 based on oracles that approximately determine maximally violated constraints. In Section 4, we apply our results to *schedule planning 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 schedule planning 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 a matroid basis. Using the framework established in Section 3 with the appropriate natural modifications, we show that the least core value 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 $u : 2^N \mapsto \mathbb{R}$ 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 consider 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 class of combinatorial optimization problems.

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

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

is supermodular on N.

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

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

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

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

Similarly, we have that

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

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

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,

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

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

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

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 $i = 1, \ldots, s-1$ and $d_s \geq 0$, implies that

$$
v(S) - v(S \setminus \{k\})
$$

$$= (d_k - d_{k+1})\Delta(S^{k-1}, k) + \sum_{i=k+1}^{s-1} (d_i - d_{i+1})\Delta(S^i \setminus \{k\}, k) + d_s \Delta(S^s \setminus \{k\}, k)$$

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

$$= v(S \setminus \{j\}) - v(S \setminus \{j,k\}).$$

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 problem*[4]

(a) $1 \,||\, \sum w_j C_j$,

(b) $Q \,|\, p_j = 1 \,|\, \sum w_j C_j$,

(c) $P \,|\, p_j = 1,\ r_j \text{ integral} \,|\, \sum w_j C_j$,

(d) $P \,||\, \sum C_j$,

(e) $1 \,|\, r_j,\ pmtn \,|\, \sum w_j M_j$,

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 $1 \,|\, r_j \,|\, \sum C_j$ and $1 \,|\, \text{prec} \,|\, \sum C_j$ 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 $u : 2^N \mapsto \mathbb{R}$ is submodular—has submodular optimal values.

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

[4]We describe these problems using the notation of Graham et al. (1979), in which the features of a scheduling problem is captured in the three-field abbreviation $\alpha \,|\, \beta \,|\, \gamma$. 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.

then the function $v : 2^N \mapsto \mathbb{R}$ *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.

The prominent example of maximizing a nonnegative linear function over a submodular polyhedron is finding a maximum weight basis of a matroid. Finding a maximum weight forest in an undirected graph is a special case of the maximum weight matroid basis problem (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 matroid basis problem.

3 Complexity and approximation

Now that we have some notion of what kind of combinatorial optimization problems have supermodular optimal costs, we turn our attention to the computational complexity and approximability of computing the least core value of (N, v), where v is supermodular. 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.*

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 $\kappa : 2^N \mapsto \mathbb{R}$ be the *cut function* of G; that is,

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

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

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

Clearly, η is nonnegative. Using the increasing marginal cost characterization of supermodularity (1.1), 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 $z^* = z = \max_{S \subseteq N, S \neq \emptyset, N} \kappa(S)$. In other words, finding the least core value of (N, v) is equivalent to finding the value of a maximum cut in $G = (N, E)$. □

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 algorithm[5] for computing the least core value of supermodular cost cooperative games, where $\rho < 1.0624$, unless P = NP.*

[5] 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.

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.

Suppose (N, v) is a cooperative game, with v supermodular. 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 optimization problem (LC). Since we are looking for the smallest value z such that $z \geq x(S) - v(S)$ for all $S \subseteq N$, $S \neq \emptyset, N$, we can determine z by maximizing $x(S) - v(S)$ over all subsets $S \subseteq N$, $S \neq \emptyset, N$. This observation motivates the following definitions. For any cooperative game (N, v) and cost allocation x such that $x(N) = v(N)$, define the function $f^x(S) = x(S) - v(S)$, and the following problem:

x-maximally violated constraint problem for cooperative game (N, v) (x-MVC).

For a given cost allocation x such that $x(N) = v(N)$, find a subset S^* such that

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

For any value of z, if $z \geq f^x(S^*)$, then (x, z) is feasible in (LC). If $z < f^x(S^*)$, then $x(S^*) \leq v(S^*) + z$ is a constraint that is maximally violated by (x, z). Intuitively, we want to find a value z that is as close to $f^x(S^*)$ as possible, but *larger* than $f^x(S^*)$, since (x, z) is feasible if and only if $z \geq f^x(S^*)$.

How should we fix x? For any set function $v : 2^N \mapsto \mathbb{R}$, we define the polytope

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

For arbitrary set functions 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 for any cost allocation x in B_v, we can show that the optimal value of the x-maximally violated constraint problem is always within a factor of 2 of the least core value 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 $f^x(S^*)$ be the optimal value of the x-maximally violated constraint problem for (N, v), and let z^* be the least core value of (N, v). Then $f^x(S^*) \leq 2z^*$.*

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,

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

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

In some sense, Theorem 3.3 tells us that any cost allocation x in B_v is "almost" an element of the least core of (N, v). We use this observation, in conjunction with a ρ-approximation algorithm for the x-maximally violated constraint 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-maximally violated constraint 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-maximally violated constraint problem for (N, v), and let $z = \rho f^x(\bar{S})$. We show that (x, z) is a feasible solution to the optimization problem (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-maximally violated constraint problem for (N, v), it follows that $z = \rho f^x(\bar{S}) \geq f^x(S^*) \geq x(S) - v(S)$ 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 $z = \rho f^x(\bar{S}) \leq \rho f^x(S^*) \leq 2\rho z^*$. □

Note that the x-maximally violated constraint 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 f^x of the x-maximally violated constraint 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 optimization problem (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-maximally violated constraint problem for (N, v), for *every* x such that $x(N) = v(N)$.[6] 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*-maximally violated constraint 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 polynomial-time algorithm for computing a* ρ*-approximate element of the least core of* (N, v)*: that is, a cost allocation* x *such that*

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

(b) *there exists a* ρ*-approximation algorithm for computing the least core value of* (N, v).

As we noted in Section 3.2, the x-maximally violated constraint 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-maximally violated constraint 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 $f : 2^N \mapsto \mathbb{R}$ with $f(\emptyset) \geq 0$ and $f(N) \geq 0$, that has a performance guarantee of $(3+\epsilon)$ for any $\epsilon > 0$. Since $f^x(\emptyset) = f^x(N) = 0$ 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* $\epsilon > 0$*,*

[6]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 $\max_{S \subseteq N, S \neq \emptyset, N} f^x(S) \geq 0$. This ensures that the notion of a ρ-approximation algorithm for the x-maximally violated constraint problem is sensible, for any given cost allocation x such that $x(N) = v(N)$.

(a) *there exists a polynomial-time algorithm for computing a* $(3+\epsilon)$*-approximate element of the least core of* (N, v)*, and*

(b) *there exists a* $(3+\epsilon)$*-approximation algorithm for computing the least core value of* (N, v).

Before proving Theorem 3.6, we first need to establish some definitions and intermediate results. To simplify the exposition, for the remainder of this subsection, we assume that v is integer-valued.

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 such that 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 such that 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 3.8 (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* ν*, then* K *has facet complexity at most* $3n^2\nu$.

We define Q to be the feasible region of the optimization problem (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 **(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 (APP-NEMPT-Q_γ).

Either

(i) *find* $x \in Q_{\rho\gamma}$ *or*

(ii) *assert* Q_γ *is empty.*

Using techniques from Grötschel et al. (1988) and Jansen (2003), we can show the following theorem. We provide the proof in Appendix A.

Theorem 3.9. *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 3.9 and the fact that an approximation algorithm for the x-maximally violated constraint problem can be used to solve the approximate separation problem for x and Q_γ.

Lemma 3.10. *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*-maximally violated constraint 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-maximally violated constraint problem for (N, v), and it outputs $\bar{S}$. If $f^x(\bar{S}) \leq \gamma$, then for all $S \subseteq N$, $S \neq \emptyset, N$, we have that

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

and therefore $x \in Q_{\rho\gamma}$. Otherwise, $f^x(\bar{S}) > \gamma$, 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-maximally violated constraint 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 3.9, allows us to solve

APP-NEMPT-Q_γ in time polynomial in n and $\log v(N)$. □

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-maximally violated constraint problem for (N, v) exists for every cost allocation x such that $x(N) = v(N)$, by Lemma 3.10, 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 optimization problem (LC) for (N, v).

1. Set the following values:

$$m = 4(n+1)^2(2(n+1) + \lceil \log(v(N)+1) \rceil + 1), \tag{3.1a}$$

$$M = 2^m, \tag{3.1b}$$

$$\epsilon = (2M)^{-2}. \tag{3.1c}$$

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

$$1 \leq q \leq 2M \qquad \text{and} \qquad \left|\bar{\gamma} - \frac{p}{q}\right| < \frac{1}{2Mq}. \tag{3.2}$$

   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 (3.1a)-(3.1c), 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 (3.2), 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 (3.1a)-(3.1c).

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 (3.1a)-(3.1c) 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 (3.2) 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 optimization problem (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 $z \leq \rho z^*$. □

4 A special case from single-machine scheduling

In this section, we study a particular supermodular cost cooperative game. Consider a situation 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 $p_i > 0$ and weight is $w_i \geq 0$. Jobs are independent, and are scheduled non-preemptively on a single machine, which can process at most one job at a time. A *schedule planning 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. The least core value of schedule planning games has a natural interpretation: it is the amount we need to charge any coalition for opening a

new machine in order to achieve 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. Schedule planning 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 schedule planning 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 schedule planning games. Before doing so, however, we establish some useful and interesting properties of the least core of schedule planning games. These properties will in turn help us determine the computational complexity of this special case, and 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 $\bar{x}$-maximally violated constraint problem for schedule planning games, as well as show a stronger translation between the approximability of the $\bar{x}$-maximally violated constraint problem and the approximability of the least core value of these games.

4.1 Key properties of the least core of schedule planning games

The structure of the cost function for schedule planning games allows us to explicitly express an element of the least core of schedule planning games and recast the least core optimization problem 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 one machine. To simplify the analysis, for the remainder of this paper we assume without loss of generality that

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

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

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

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

for $i = 1, \ldots, n$, where $S^i = \{1, \ldots, i\}$ 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 schedule planning 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 schedule planning game. Then, the cost allocation $\bar{x}$ as defined in* (4.2) *satisfies*

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

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 $S \subseteq N$,

$$v(S) = \sum_{i \in S} \sum_{\substack{j=1 \\ j \in S}}^{i} w_i p_j.$$

Therefore,

$$\begin{aligned}
2\big(\bar{x}(S) - v(S)\big) &= \sum_{i \in S} \sum_{j=1}^{i} w_i p_j + \sum_{i \in S} \sum_{j=i}^{n} p_i w_j - 2 \sum_{i \in S} \sum_{\substack{j=1 \\ j \in S}}^{i} w_i p_j \\
&= \sum_{i \in S} \sum_{j=1}^{i} w_i p_j + \sum_{i \in S} \sum_{j=i}^{n} p_i w_j - \sum_{i \in S} \sum_{\substack{j=1 \\ j \in S}}^{i} w_i p_j - \sum_{i \in S} \sum_{\substack{j=i \\ j \in S}}^{n} p_i w_j \\
&= \sum_{i \in S} \sum_{\substack{j=1 \\ j \in N \setminus S}}^{i} w_i p_j + \sum_{i \in S} \sum_{\substack{j=i \\ j \in N \setminus S}}^{n} p_i w_j \\
&= \sum_{i \in S} \sum_{\substack{j=1 \\ j \in N \setminus S}}^{i} w_i p_j + \sum_{i \in N \setminus S} \sum_{\substack{j=1 \\ j \in S}}^{i} w_i p_j \\
&= \sum_{i \in N} \sum_{j=1}^{i} w_i p_j - \sum_{i \in S} \sum_{j=1}^{i} w_i p_j - \sum_{i \in N \setminus S} \sum_{j=1}^{i} w_i p_j
\end{aligned} \tag{4.3}$$

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

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

Theorem 4.2. *Suppose* (N, v) *is a schedule planning 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*

$$
z^* = \frac{1}{2} \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \big\{ v(N) - v(S) - v(N \setminus S) \big\}. \tag{4.4}
$$

Proof. Let $\bar{z}$ be the value of the right-hand side of (4.4). 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,

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

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 value of schedule planning games (Mishra and Rangarajan 2005). This is quite special: in the cooperative game (N, v) defined in Example 4.3, the cost function v is supermodular, but the Shapley value

is not necessarily an element of the least core of (N, v).

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 $\sigma : N \mapsto N$ where $\sigma(i)$ denotes the position of player $i \in N$, we define the cost allocation $\bar{x}^\sigma$ as follows:

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

for $i = 1, \ldots, n$, where $S^i = \{\sigma^{-1}(1), \ldots, \sigma^{-1}(i)\}$, and $S^0 = \emptyset$. The cooperative game (N, v) defined in Example 4.3 is an instance of a cooperative game with v nonnegative and supermodular (in particular, of the form (2.1)), for which the cost allocation $\bar{x}^\sigma$ 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 schedule planning 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 $||\sum C_j$, 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-maximally violated constraint problem for this game is $\max_{S \subseteq N, S \neq \emptyset, N} f^\phi(S) = 5/3$. However, the least core value of this game is $3/2$. It is also straightforward to check that $\max_{S \subseteq N, S \neq \emptyset, N} f^{\bar{x}^\sigma}(S) = 2$ 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 schedule planning games. In the previous subsection, we showed that we can efficiently compute an element of the least core of schedule planning games. In fact, we have an explicit formula for a least core element. Computing the least core value of schedule planning games, however, remains NP-hard.

Theorem 4.4. *Computing the least core value of scheduling planning 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 schedule planning games is

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

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 $j \in N$. □

The above result is in stark contrast to the underlying problem defining the costs in schedule planning 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-maximally violated constraint problem implies a 2ρ-approximation algorithm for computing the least core value of (N, v). It is reasonable to believe, though, that for schedule planning 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

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

This is exactly the $\bar{x}$-maximally violated constraint problem for schedule planning games! Therefore, we obtain the following strengthening of Theorem 3.4.

Theorem 4.5. *Suppose there exists a ρ-approximation algorithm for the $\bar{x}$-maximally violated constraint problem for schedule planning 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 schedule planning games.*

In the following section, we show that we can design such ρ-approximation algorithms for the $\bar{x}$-maximally violated constraint problem for schedule planning games with small values of ρ.

4.4 Approximately solving $\bar{x}$-MVC for schedule planning games

Some proofs from the previous subsections give us insight into how to design oracles that approximately solve $\bar{x}$-MVC for schedule planning games. By carefully looking at the proof of Lemma 4.1, we can show that $\bar{x}$-MVC for schedule planning games is actually a special case of finding a maximum weighted cut in a complete undirected graph. In the proof of Theorem 4.4, we see that $\bar{x}$-MVC for schedule planning games is actually equivalent (with respect to optimization) to the scheduling problem P2 $||\sum w_j C_j$, implying that we might be able to use or modify existing algorithms to approximately solve $\bar{x}$-MVC.

4.4.1 A maximum-cut based approximate oracle

By (4.3), we have that

$$f^{\bar{x}}(S^*) = \frac{1}{2} \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \left\{ \sum_{j \in S} \sum_{i \in N \setminus S} \mu_{ij} \right\} \quad \text{where} \quad \mu_{ij} = \begin{cases} w_j p_i & \text{if } i < j \\ w_i p_j & \text{if } i > j \end{cases}$$

for all $i \neq j$. Observe that $\mu_{ij} = \mu_{ji}$. So $f^{\bar{x}}(S^*)$ is proportional to the value of the maximum cut on a complete undirected graph with node set N and capacity μ_{ij} for arc $\{i, j\}$. Therefore, if we have a ρ-approximation algorithm for the maximum cut problem, then by Theorem 4.5, we have a ρ-approximation algorithm for finding the least core value of schedule planning games. For example, using the approximation algorithm of Goemans and Williamson (1995) based on a semidefinite relaxation of the maximum cut problem yields a 1.1382-approximation algorithm for finding the least core value of schedule planning games. However, using the algorithm of Goemans and Williamson does not exploit the special structure of this particular maximum cut problem, since their method applies for maximum cut problems on general undirected graphs.

4.4.2 A fully polynomial-time approximation scheme based on two-machine scheduling

In this subsection, we provide a fully polynomial time approximation scheme[7] (FPTAS) for the $\bar{x}$-maximally violated constraint problem for schedule planning games. By Theorem 4.2, we know that $\bar{x}$-MVC is in fact

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

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

As mentioned earlier, maximizing $g(S)$ is equivalent to minimizing the sum of weighted completion times of jobs in N on two identical parallel machines. This problem is denoted by $\mathrm{P2}\,||\,\sum w_jC_j$ in the notation of Graham et al. (1979). $\mathrm{P2}\,||\,\sum w_jC_j$ is NP-complete (Bruno et al. 1974), and has an 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 $\bar{x}$-MVC exactly for schedule planning games in psuedopolynomial time, and then convert this dynamic program into an FPTAS. This development is inspired by the FPTAS for $\mathrm{P2}\,||\,\sum w_jC_j$ by Sahni (1976). The analysis is similar to the analysis of the FPTAS for $\mathrm{P2}\,||\,C_{\max}$ by Schuurman and Woeginger.

We think of determining the maximizer S^* 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 $w_k(a+b+p_k)-w_k(a+p_k) = w_kb$. If job k is scheduled on machine 2, then the running objective value increases by $w_k(a+b+p_k)-w_k(b+p_k) =$

[7] A *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 $1/\epsilon$.

$w_k a$. This suggests the following dynamic programming algorithm.

Algorithm 4.6. (Exact dynamic program)

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

Output: the optimal value of $\bar{x}$-MVC for schedule planning games, $f^{\bar{x}}(S^*)$.

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

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

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 $k = 1, \ldots, n$, 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 $(\delta^{-r}, \delta^{-s}, \delta^{-t})$, $r = 1, \ldots, L$, $s = 1, \ldots, L$, $t = 1, \ldots, M$. We divide each of the state sets E_k, $k = 1, \ldots, n$, into the boxes formed by the grid:

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

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

$$\delta a_1 \leq a_2 \leq \frac{a_1}{\delta}, \qquad \delta b_1 \leq b_2 \leq \frac{b_1}{\delta}, \qquad \delta c_1 \leq c_2 \leq \frac{c_1}{\delta}. \tag{4.5}$$

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 4.7. (Dynamic program with "trimmed" state space)

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

Output: an approximation to the optimal value of $\bar{x}$-MVC for schedule planning games, $f^{\bar{x}}(\bar{S})$.

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

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

For $k = 2, \ldots, n$

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 $f^{\bar{x}}(\bar{S}) = \frac{1}{2} g(\bar{S}) = \frac{1}{2}\bar{c}$

The key property of the "trimmed" state space used in Algorithm 4.7 is that every element in the original state space has an element in the "trimmed" state space that is relatively close. In particular,

Lemma 4.8. *For every $(a, b, c) \in E_k$, there exists a vector $(a', b', c') \in E_k^\delta$ such that*

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

Proof. By induction. The base case $k = 1$ holds by (4.5). 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 $\alpha' \geq \delta^{k-1}\alpha$, $\beta' \geq \delta^{k-1}\beta$, and $\gamma' \geq \delta^{k-1}\gamma$. 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

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

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. □

Finally, we analyze the performance and running time of the trimmed dynamic programming algorithm.

Theorem 4.9. *Algorithm 4.7 is a fully polynomial time approximation scheme for the $\bar{x}$-maximally violated constraint problem for schedule planning games.*

Proof. 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:

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

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

Now we analyze the performance of this algorithm. Let $c^* = g(S^*)$, the optimal value of g. Note that there exists a vector $(a^*, b^*, c^*) \in E_n$. By Lemma 4.8, 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 $c' \geq (1 + \epsilon/(2n))^{-n} c^* \geq (1 + \epsilon)^{-1} c^*$. □

Combining Theorem 4.5 and Theorem 4.9, gives us the following result.

Theorem 4.10. *There exists a fully polynomial time approximation scheme for computing the least core value of schedule planning games.*

5 Submodular profits and matroid optimization

Up to this point, we have only considered cooperative games in which coalitions 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 fairness of a profit allocation; for example, the core for a profit cooperative game (N, v) is defined as

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

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

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

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 in order to encourage cooperation amongst all agents.

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, we define the function $f^x : 2^N \mapsto \mathbb{R}$ as $f^x(S) = v(S) - x(S)$ for any given profit allocation x such that $x(N) = v(N)$, and for all $S \subseteq N$. 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-maximally violated constraint problem for a cooperative game (N, v) with v representing profits is still to find a subset S^* such that $f^x(S^*) = \max_{S \subseteq N, S \neq \emptyset, N} f^x(S)$.

In this section, we study the following game. Let $(N, \mathcal{I})$ be a matroid with weights $w_i \geq 0$ for each $i \in N$, and let $\mathcal{I}|S = \{T \in \mathcal{I} : T \subseteq S\}$ for any subset $S \subseteq N$. We define $v(S)$ as the maximum w-weight of a basis in $(S, \mathcal{I}|S)$, for every subset $S \subseteq N$. Then (N, v) defines a cooperative game where the profits 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.

By Theorem 2.3, matroid profit games are submodular profit cooperative games. It turns out that the x-maximally violated constraint 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-maximally violated constraint 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-maximally violated constraint problem for (N, v):

Input: matroid profit game (N, v) with matroid $(N, \mathcal{I})$ and weights $w_i \geq 0$ for all $i \in N$.

Output: an optimal solution $\bar{S}$ to x-MVC for (N, v).

1. Compute a maximum $\bar{w}$-weight basis 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 $\bar{T} = T^* \cup (A \setminus T^*)$.
   - If $\bar{T} \neq \emptyset, N$, output $\bar{S} = \bar{T}$.
   - Otherwise, output $\bar{S} = \arg\max\{f^x(S) : S \in \{T, N \setminus T\}\}$ for an arbitrary $T \subseteq N$, $T \neq \emptyset, N$.

First, note that any optimal solution of the following relaxation of the x-maximally violated constraint problem

$$\max_{S \subseteq N} f^x(S) = \max_{S \subseteq N} \big\{v(S) - x(S)\big\} \tag{5.1}$$

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

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

We show that the basis T^* computed in Step 1 of the above algorithm is an optimal solution to (5.2). Let S^* be an optimal solution to (5.2), 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 basis of $(S^*, \mathcal{I}|S^*)$. If $x_i \geq 0$, then $i \in S^* \setminus A$, and can be removed without decreasing the objective value of (5.2); if $x_i < 0$, then $i \in A$, and removing it does not affect the objective value of (5.2). Therefore,

$$\bar{w}(S^*) = w(S^*) - x(S^* \setminus A)$$

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

which contradicts the assumption that T^* is a maximum $\bar{w}$-weight basis of $(N, \mathcal{I})$. So T^* is an optimal solution to (5.2).

Using this fact, we show that the output of the above algorithm is correct. Note that

$$
\begin{aligned}
\max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} f^x(S) &\leq \max_{S \subseteq N} f^x(S) && \text{(5.3a)} \\
&= \max_{S \subseteq N} \left\{ \upsilon(S) - x(S \setminus A) \right\} - x(A) && \text{(5.3b)} \\
&= \upsilon(T^*) - x(T^* \setminus A) - x(A) && \text{(5.3c)} \\
&\leq \upsilon(\bar{T}) - x(\bar{T}) && \text{(5.3d)} \\
&= f^x(\bar{T}). && \text{(5.3e)}
\end{aligned}
$$

We consider two cases.

1. $\bar{T} \neq \emptyset, N$. By (5.3a)-(5.3e), it follows that $\bar{T}$ is an optimal solution to x-MVC for (N, υ).
2. $\bar{T} = \emptyset$ or $\bar{T} = N$. In this case, $f^x(\bar{T}) = 0$. Fix some $T \subseteq N$, $T \neq \emptyset, N$. Since υ is submodular and $\upsilon(\emptyset) = 0$, we have that

$$
f^x(T) + f^x(N \setminus T) = \upsilon(T) + \upsilon(N \setminus T) - \upsilon(N) \geq 0.
$$

Therefore, we must have $f^x(T) \geq 0$, or $f^x(N \setminus T) \geq 0$, or both. Without loss of generality, suppose $f^x(T) \geq 0$. It follows from (5.3a)-(5.3e) that

$$
\max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} f^x(S) \leq 0 \leq f^x(T).
$$

Therefore, $\arg\max\{f^x(S) : S \in \{T, N \setminus T\}\}$ for any given $T \subseteq N$, $T \neq \emptyset, N$ is an optimal solution to x-MVC for (N, υ).

Since the maximum weight basis of a matroid can be found in polynomial time (Rado 1957, Edmonds 1971), it follows that the above algorithm solves the x-maximally violated constraint problem for a matroid profit game (N, v) in polynomial time. □

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-maximally violated constraint 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 the x-maximally violated constraint problem as a 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 an element of the least core of (N, v), and*

(b) *computing the least core value of (N, v).*

6 Conclusion

In a cooperative game with supermodular costs, cooperation amongst agents is unlikely: as a coalition grows, the cost of adding a particular agent increases, making the idea of sharing costs less appealing. As we showed, these situations arise when sharing the optimal costs of a variety of combinatorial optimization problems, especially in machine scheduling. In this paper, we considered one way of encouraging cooperation in these situations: the least core value of a cooperative game, or the minimum penalty we need to charge a coalition for defecting to ensure cooperation amongst all agents. 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. This framework, with the appropriate natural modifications, can also be used to approximate the least core value of submodular profit cooperative games. Using this framework with the approximation algorithms for submodular function maximization of Feige et al. (2007), we obtained a $(3 + \epsilon)$-approximation algorithm for computing the least core value of both supermodular cost cooperative games and submodular profit cooperative games. In addition, we used this framework to design a fully polynomial-time approximation scheme for computing the least core value of schedule planning games, as well as an exact polynomial-time algorithm for computing the least core

value of matroid profit games.

Acknowledgments

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

A Proof of Theorem 3.9

In this appendix, we establish a result that generalizes Theorem 3.9: 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 ideas here closely follow the analyses found in Grötschel et al. (1988) and Jansen (2003).

A.1 Preliminaries

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 $\varphi + n$.

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

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

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

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

A.2 Approximate separation and non-emptiness

For this subsection, we 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 (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.1 (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 $\mathrm{vol}(E(A, a)) \leq \epsilon$.

1. Set the following values:

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

$$p = 8N \tag{A.2}$$

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:

$$a_0 = 0 \tag{A.3}$$

$$A_0 = R^2 I \tag{A.4}$$

   - 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

$$\|c_k\|_\infty = 1 \tag{A.5}$$

$$c_k^\mathsf{T} a_k > c_k^\mathsf{T} x \quad \text{for all } x \in K \tag{A.6}$$

   then compute

$$a_{k+1} \approx a_k - \frac{1}{n+1}\frac{A_k c_k}{\sqrt{c_k^\mathsf{T} A_k c_k}} \tag{A.7}$$

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

   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.2 (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 ($k = 0, 1, \ldots, N$) are defined as in* (A.3)-(A.4) *and* (A.7)-(A.8)*, and c_k ($k = 0, 1, \ldots, N$) satisfy* (A.5)-(A.6)*. Then, the following statements hold for $k = 0, 1, \ldots, N$:*

(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.3. *Algorithm A.1 (APP-ELL) is correct.*

Proof. Lemma A.2 immediately implies that A_k and a_k ($k = 0, 1, \ldots, N$) 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.2 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. □

Now consider the following problem:

Approximate non-emptiness problem (APP-NEMPT).

Either

(i) *find a vector* $y \in \bar{K}$ *or*

(ii) *assert* K *is empty.*

We can use APP-ELL (Algorithm A.1) in conjunction with an oracle for S-APP-SEP to solve APP-NEMPT. To show this, we need the following lemma.

Lemma A.4 (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* $K \subseteq \{x \in \mathbb{R}^n : f^\mathsf{T} x = g\}$. *Moreover,* f *and* g *can be found in time polynomial in* n, φ, *and the encoding length of* A^{-1}.

Finally, we are ready to show the main result of this appendix.

Theorem A.5. *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 3.8, K has vertex complexity at most $4n^2\varphi$. Apply APP-ELL (Algorithm A.1) 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.4, 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

$$K \subseteq \left\{x \in \mathbb{R}^n : \begin{pmatrix} F_1 & F_2 \end{pmatrix} x = g\right\}$$

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

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

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

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

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 $\left(\begin{smallmatrix} z^* \\ u^* \end{smallmatrix}\right)$ 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
$$y = \begin{pmatrix} F_1^{-1} g - F_1^{-1} F_2 w \\ w \end{pmatrix}$$
2. 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

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

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 $\text{vol}(E^k) \leq \epsilon$. Therefore, by Lemma A.4 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 $K \subseteq \{x \in \mathbb{R}^n : (f^{k+1})^\mathsf{T} x = g^{k+1}\}$. By the induction hypothesis,

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

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 $\{x \in \mathbb{R}^n : (f^1)^\intercal x = g^1, \ldots, (f^n)^\intercal x = g^n\}$, 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.2(b) and Lemma A.4, 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. □

Note that Theorem A.5 implies Theorem 3.9, since the facet complexity of Q_γ is polynomially bounded by n and the encoding length of $v(N) + \gamma$.

References

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.

I. Curiel, G. Pederzoli, and S. Tijs. Sequencing games. *European Journal of Operational Research*, 40: 344–351, 1989.

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.

J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1:127–136, 1971.

U. Faigle and W. Kern. Approximate core allocation for binpacking games. *SIAM Journal on Discrete Mathematics*, 11:387–399, 1998.

U. Faigle, S. P. Fekete, W. Hochstättler, and W. Kern. On approximately fair cost allocation for Euclidean TSP games. *OR Spektrum*, 20:29–37, 1998.

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. Feige, V. Mirrokni, and J. Vondrák. Maximizing non-monotone submodular functions. To appear in *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, 2007.

M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

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 and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

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.

J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48:798–859, 2001.

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

K. Jansen. Approximate strong separation with application in fractional graph coloring and preemptive scheduling. *Theoretical Computer Science*, 302:239–256, 2003.

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.

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.

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.

M. Pál and É. Tardos. Group strategyproof mechanisms via primal-dual algorithms. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 584–593, 2003.

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 *Proceedings of the 4th International Conference on Integer Programming and Combinatorial Optimization*, volume 920 of *Lecture Notes in Computer Science*, pages 307–320, 1995.

R. Rado. Note on independence functions. In *Proceedings of the London Mathematical Society*, volume 7, pages 300–320, 1957.

S. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23:116–127, 1976.

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. 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.

W. E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.

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.