Encouraging Cooperation in Sharing Supermodular Costs

Andreas S. Schulz* Nelson A. Uhan†

March 31, 2007

Extended Abstract

Consider a situation where a group of agents wishes to share the costs of their joint actions, and needs 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 optimal cost of scheduling these jobs. This kind of situation can be modelled naturally as a cooperative game.

In this work, we are concerned with cooperative games with supermodular costs. A set function $r : 2^N \mapsto \mathbb{R}$ is *supermodular* if

$$r(S \cup \{i\}) - r(S) \leq r(T \cup \{i\}) - r(T) \quad \forall S \subseteq T \subseteq N \backslash \{i\}.$$

Our primary motivation behind studying these games is that many problems from combinatorial optimization have optimal costs that are supermodular. In particular, we show the following theorem (for notational convenience, for any vector x we define $x(S) := \sum_{i \in S} x_i$):

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

$$v(S) = \min \left\{ \sum_{j \in S} d_j x_j : x(A) \geq r(A) \; \forall A \subseteq S \right\} \quad \forall S \subseteq N \tag{1}$$

is supermodular on N.

A variety of machine scheduling problems can be modeled as an optimization problem of the form in (1), including minimizing the sum of weighted completion times on a single machine. Some other combinatorial optimization problems that can be formulated as an optimization problem of the form in (1) include the minimum-cost spanning tree problem, and more generally, finding the minimum weight basis of a matroid.

It is well known that cooperative games with submodular, or decreasing marginal costs always have nonempty cores. This result is very intuitive. As the number of players increases, the average cost per player decreases, making the idea of sharing costs more appealing. On the other hand, cooperative games with supermodular, or increasing marginal costs always have empty cores. Similar intuition still holds: the average cost per player increases as the number of players increases, diminishing the appeal of sharing costs with other players.

Since the prospect for cooperation in supermodular cost cooperative games is bleak, we are led to ask, "how much do we need to penalize a coalition for defecting in order to achieve cooperation?" This notion is captured in the *least core* solution concept. The least core of a cooperative game (N, v) is the set of all optimal cost allocations x to the following linear program:

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

*Sloan School of Management, MIT, 77 Massachusetts Avenue, E53-361, Cambridge, MA 02139, e-mail: schulz@mit.edu
†Operations Research Center, MIT, 77 Massachusetts Avenue, E40-130, Cambridge, MA 02139, e-mail: uhan@mit.edu

We call z^*, the optimal value of (LC), the *least core value* of (N, v). The least core value of a cooperative game can be interpreted as the minimum penalty for defection necessary to achieve cooperation among all players.

The main focus of our work is the computational complexity and algorithmic aspects of computing the least core value of supermodular cost cooperative games. Along the way, we also uncover some structural properties of the least core of these games.

In terms of computational complexity, we show:

Theorem. *Computing the least core value of supermodular cost cooperative games is NP-hard.*

In our proof of the above theorem, we reduce any instance of finding the maximum cut of an undirected graph to our problem. As a result, we immediately obtain the following inapproximability result.

Corollary. *There is no ρ-approximation algorithm*[1] *for computing the least core value of supermodular cost cooperative games, where $\rho < 1.0624$, unless P=NP.*

The above negative results indicate that it is rather unlikely that we will be able to find exact polynomial time algorithms for computing the least core value of supermodular cost games. This motivates us to design methods with polynomial running time that approximate the value of z^*.

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 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 the maximization problem

$$z = \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \{x(S) - v(S)\}. \tag{2}$$

For any x, we call the maximization problem in the right hand side of (2) the *x-maximally violated constraint (x-MVC) problem* for (N, v). But how should we fix x? Inspired by the properties of vertices of supermodular polyhedra, we define the cost allocation $\bar{x}$ as

$$\bar{x}_i := \frac{1}{2}\Big(v\big(\{1, \ldots, i\}\big) - v\big(\{1, \ldots, i-1\}\big)\Big) + \frac{1}{2}\Big(v\big(\{i, \ldots, n\}\big) - v\big(\{i+1, \ldots, n\}\big)\Big) \tag{3}$$

for $i = 1, \ldots, n$. When we fix the cost allocation to $\bar{x}$, the smallest value of $\bar{z}$ such that $(\bar{x}, \bar{z})$ is feasible in (LC) is within a factor of 2 of the least core value. In other words, we are able to show that in some sense, $\bar{x}$ as defined above is "almost" an element of the least core:

Theorem. *Suppose (N, v) is a supermodular cost cooperative game. Let z^* be its least core value. Then,*

$$\max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \{\bar{x}(S) - v(S)\} \leq 2z^*.$$

This result allows us to translate the approximability of the $\bar{x}$-maximally violated constraint problem to the approximability of computing the least core value of supermodular cost cooperative games. In particular,

Theorem. *Suppose (N, v) is a supermodular cost cooperative game, and there exists a ρ-approximation algorithm for the $\bar{x}$-maximally violated constraint problem for (N, v). Then there exists a 2ρ-approximation algorithm for computing the least core value of (N, v).*

[1] 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 size of the input.

One might wonder if we can do better by computing a cost allocation on the fly, instead of fixing a cost allocation in advance. If we assume that for any x such that $x(N) = v(N)$, we can approximately solve the x-maximally violated constraint problem for (N, v), then with the help of the ellipsoid method and binary search, we can show that this is indeed the case:

Theorem. *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 any x such that $x(N) = v(N)$. Then there exists a ρ-approximation algorithm for computing the least core value of (N, v).*

As mentioned earlier, our primary motivation for studying cooperative games with supermodular costs is that many machine scheduling problems have supermodular optimal costs. We apply our results to a particular supermodular cost cooperative game arising from a classic scheduling problem. Consider a situation where agents each have a job that needs to processed on a machine, and any coalition of the players can potentially open their own machine. Suppose each agent $i \in N$ has a job whose processing time is $p_i \in \mathbb{N}$ and weight is $w_i \in \mathbb{N}$. 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 $v(S)$ is the minimum sum of weighted completion times of jobs in S. The least core value of schedule planning games can be interpreted as the amount we need to charge for opening a new machine in order to achieve cooperation.

For schedule planning games, we are able to make more explicit characterizations of properties of the least core. In particular,

Theorem.

1. *The cost allocation $\bar{x}$ defined in (3) is an element of the least core of schedule planning games.*
2. *The least core value of schedule planning games is*

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

Since the cost allocation $\bar{x}$ defined in (3) is in fact an element of the least core of schedule planning games, we are able to get tighter bounds when we translate the approximability of the $\bar{x}$-maximally violated constraint problem to the approximability of the least core value problem:

Theorem. *Suppose there exists a ρ-approximation algorithm for the $\bar{x}$-maximally violated constraint problem for schedule planning games. Then there exists a ρ-approximation algorithm for computing the least core value of schedule planning games.*

In addition, for schedule planning games, we design approximation algorithms for the $\bar{x}$-maximally violated constraint problem. We show that the $\bar{x}$-maximally violated constraint problem is in fact equivalent to finding the maximum cut in a complete undirected graph, with particular edge weights. This observation lets us use any approximation algorithm for the maximum cut problem for the $\bar{x}$-maximally violated constraint problem. In addition, we attack the $\bar{x}$-maximally violated constraint problem directly, and design a fully polynomial time approximation scheme[2]. As a consequence, we get the following result:

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

[2] 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 $1/\epsilon$.