Computing Shapley Value in Supermodular Coalitional Games*

David Liben-Nowell**, Alexa Sharp***, Tom Wexler†, and Kevin Woods‡

Abstract. Coalitional games allow subsets ("coalitions") of players to cooperate to receive a collective payoff. This payoff is then distributed "fairly" among the members of that coalition according to some division scheme. Various solution concepts have been proposed as reasonable schemes for generating fair allocations. The *Shapley value* is one classic solution concept: player i's share is precisely equal to i's expected marginal contribution if the players join the coalition one at a time, in a uniformly random order. In this paper, we consider the class of supermodular games (sometimes called "convex" games), define and survey computational results on other standard solution concepts, and contrast these results with new results regarding the Shapley value. In particular, we give a fully polynomial-time randomized approximation scheme (FPRAS) to compute the Shapley value to within a $(1 \pm \varepsilon)$ factor in monotone supermodular games. We show that this result is tight in several senses: no deterministic algorithm can approximate Shapley value as well, no randomized algorithm can do better, and both monotonicity and supermodularity are required for the existence of an efficient $(1 \pm \varepsilon)$-approximation algorithm. We also argue that, relative to supermodularity, monotonicity is a mild assumption, and we discuss how to transform supermodular games to be monotonic.

* This work was supported in part by NSF grant CCF-0728779 and by grants from Oberlin College and Carleton College. Thanks to Josh Davis for helpful discussions.

** Department of Computer Science, Carleton College. dlibenno@carleton.edu

*** Department of Computer Science, Oberlin College. asharp@oberlin.edu

† Department of Computer Science, Oberlin College. twexler@oberlin.edu

‡ Department of Mathematics, Oberlin College. kwoods@oberlin.edu

1 Introduction

Game theory is broadly defined as the study of self-interested players. These players may be restricted to make independent decisions, leading to *competitive* games; alternatively, players may be allowed to cooperate in order to achieve their goals, leading to *coalitional*, or *cooperative*, games. Both models provide rich computational and theoretical challenges, as demonstrated by a long history of research across the fields of economics, politics, and computer science.

In any game, we are interested in the outcomes that might reasonably be achieved as a result of players' self-interested behavior. For competitive games, the standard solution concept is the Nash equilibrium; for coalitional games, a number of reasonable solution concepts exist, each with its own merits. One class of solution concepts focuses on "fair" divisions of wealth. Assume that all players work together, to form the so-called "grand coalition" of all players. How can the total utility generated by the grand coalition be divided so that each player's portion is proportional to his or her influence or power? The *Shapley value* [37] is one such "fair allocation" scheme. A second class of solution concept focuses on "stable" divisions of wealth. Suppose again that all players cooperate to form the grand coalition. How can their total utility be divided so that no deviating coalition—a subset of deviating players who benefit in some way—can form? The *core* and the *kernel* are two such "stable allocation" schemes that differ in the character of the stability that they provide.

Generally, under these solution concepts for coalitional games, solutions are hard to compute. In some restricted domains, however, it may be possible to find exact or approximate solutions efficiently. In this paper, we consider *supermodular* games (also known as "convex" games in the economics literature [38]), a class of coalitional games in which incentives for joining a coalition increase as the coalition grows. We first survey some known efficient algorithms to test core membership and compute the kernel in supermodular games. The remainder of the paper is devoted to the efficient computation of Shapley value in supermodular games.

1.1 Related Work: Solution Concepts for Coalitional Games

The Shapley value is an established mechanism for "fair" wealth distribution in coalitional games, just as the core and kernel are established mechanisms for "stable" wealth distribution. Lloyd Shapley first introduced his eventually eponymic solution concept in 1953 [37]. Gillies first introduced the core as a tool to study "stable sets" [17], although Shapley and Shubik were the ones to develop it as a solution concept [39]. Davis and Maschler first introduced the kernel in [8]. These solution concepts have all been studied extensively, and they are described and surveyed in, e.g., [2, 4, 9, 10, 16, 26].

Finding the allocations described by these solution concepts is in general computationally intractable. Hence, much of the research in this area focuses on a variety of restricted domains in which one can hope to find either a reasonably efficient or reasonably approximate solution.

One such domain is "weighted majority" games, in which a coalition receives a payoff of 1 if its members constitute a majority of all players' weights, and 0 otherwise. Mann–Shapley [27] motivate this class of coalitional games and propose a Monte Carlo sampling algorithm to approximate the Shapley value. They apply their algorithm to the U.S. electoral college data, but they do not provide formal analysis. Deng–Papadimitriou [11] and Matsui–Matsui [31] show that it is NP-hard to determine whether a given player has nonzero Shapley value, and #P-hard to calculate it exactly. Matsui–Matsui [30] make the exact computation with a pseudo-polynomial dynamic programming algorithm, and find that Mann–Shapley's algorithm has error that goes to zero like $1/\sqrt{\#\text{samples}}$. Testing membership in the core for weighted voting games is coNP-complete [11]; Elkind et al. [12, 13] give a pseudo-polynomial algorithm to test core membership. In contrast, testing membership in the kernel can be done in polynomial time [2].

Another domain to receive attention is that of *simple* coalitional games, a generalization of weighted majority games in which every coalition has a payoff of 0 or 1. Bachrach et al. [3] apply the Mann–Shapley algorithm in this more general setting, and give an oracle-based sampling algorithm to approximate Shapley value in polynomial time, also with $1/\sqrt{\#\text{samples}}$ error. They also show that no approximation algorithm can do much better by giving lower bounds for both deterministic and randomized algorithms for these calculations.

A third domain with interesting results is that of *submodular* games [35, 36], in which incentives for joining a coalition *decrease* as the coalition grows. These games have an empty core, and one can instead examine the *least core*. Schulz and Uhan show that the least-core value is inapproximable to within a factor of 17/16 if $\mathsf{P} \neq \mathsf{NP}$ [36], though there is a $(3+\varepsilon)$-approximation [35]. They further show that for *scheduling* games, a special class of submodular games, the least-core value can be well-approximated [35].

1.2 Related Work: Supermodular Games

Supermodular coalitional games are another restricted class of coalitional games, and the class upon which we focus in this paper. These games, first introduced by Shapley [38], who called them *convex* games, capture the intuitive notion that incentives for joining a coalition increase as the coalition grows. In addition to many natural applications that result in supermodular coalitional games, these games also have pleasing theoretical properties: the core is nonempty [38], the Shapley value is in the core and is the center of mass of the core's vertices [38], the kernel is a single point corresponding to the *nucleolus* [29], and the *stable set* and *bargaining set* for the grand coalition coincide with the core [28]. Many specific and natural examples of supermodular games are studied in the literature; a sampling of these games are described in the remainder of this section.

The *multicast tree game* [1, 15, 19, 22] is used to model distribution networks such as waterways and telecommunication networks. Players receive a payoff for being connected to the source but must cover the cost of building the underlying (and fixed) tree. As more players join the network, the cost to a previously connected player cannot increase, and thus the game is supermodular.

The *edge synergy game* [11] takes place on an undirected graph, with the nodes as players. Each edge of a graph has an associated nonnegative benefit, and the value of a coalition is the sum of the benefits in its induced subgraph. This game possesses increasing returns of scale, as a player who joins a coalition adds value for each neighbor already in that coalition.

The *bankruptcy game*, first studied by O'Neill [33], is another natural supermodular game. In this setting, there is an estate to which each player has some claim, but not all claims can be satisfied. The value of a coalition S is what remains of the estate after satisfying the claims of the players *not* in S. Curiel et al. [7] show that bankruptcy games are supermodular.

These examples are meant to sample just some of the supermodular games in the literature; other specific examples have been recently studied algorithmically by, e.g., Jain–Vazirani [22] and Ieong–Shoham [20].

1.3 Our Results

In many of the specific games described in Section 1.2, computing the exact Shapley value is fairly easy; however, to the best of our knowledge, efficient computation of Shapley value for general supermodular games has not been addressed. (The core and kernel, however, are known to be computable in polynomial time [14, 18, 25, 41], as discussed in Section 3.)

NOTE: There was a bug in our proof of Theorem 4, which has (to the best of our knowledge) been resolved in subsequent work by Sasan Maleki. Please see p. 7.

Our main result (Theorem 4) resolves the open question regarding the computation of Shapley value in supermodular games: we give an efficient randomized algorithm to approximate arbitrarily well the Shapley value of any monotone supermodular coalitional game. Specifically, we show that the Mann–Shapley Monte Carlo sampling algorithm [27] is a fully polynomial-time randomized approximation scheme (FPRAS) for the Shapley value of a monotone supermodular game, assuming access to an oracle that returns the value of any given coalition. Furthermore, we show that this result is tight in the following senses: no fully polynomial deterministic algorithm can approximate Shapley value as well; no randomized algorithm can do better; and *both* monotonicity and supermodularity are required to approximate Shapley value within *any* multiplicative factor. Note that our definition of polynomial running time is slightly atypical, as we do not have the entire game as input, but rather only have oracle access to it. Also note that our negative results hold regardless of whether $\mathsf{P} \neq \mathsf{NP}$.

The remainder of this paper is structured as follows. Section 2 formally defines coalitional games and relevant solution concepts. Section 3 surveys efficient algorithms to test core membership and compute the kernel in supermodular games. We then turn to Shapley value: Section 4 presents and analyzes the FPRAS for Shapley value in monotone supermodular games, and Section 5 shows that this FPRAS is essentially the best result that one can hope to achieve.

Finally, Section 6 gives some reason to believe that the additional assumption of monotonicity in a supermodular game is reasonably natural. Specifically, we examine two ways to convert any supermodular game v into a monotone supermodular game. The *zero-normalization transform*, v_Z, translates the utility function so that $v_Z(\{i\}) = 0$ for each player i. This shift does not change

the strategies of the players, and, hence for any supermodular v we can approximate player i's gain over $v(\{i\})$ under the Shapley allocation. The *opt-out transform*, v_O, allows any players who contribute negative value to a coalition not to participate, thereby receiving zero utility. While v_O is a more substantive transform of v, in many settings it is a natural operation. For both transforms, we prove that if the original game is supermodular, then the transformed game is both supermodular and monotone, and furthermore we can compute the value of any coalition efficiently. Thus our results from Sections 4 and 5 apply to both v_O and v_Z.

2 Model and Definitions

A coalitional game v is defined by a set N of n players, and a function $v : \mathscr{P}(N) \to \mathbb{R}$, where $v(S)$ denotes the value generated by a coalition $S \subseteq N$. Without loss of generality, we assume throughout that $v(\emptyset) = 0$. We further assume that a game is represented by an oracle that, given $S \subseteq N$, returns $v(S)$.

A game is *monotone* if, for all $S \subseteq T \subseteq N$, we have $v(S) \leq v(T)$.

A game is *supermodular* if $v(S \cup T) + v(S \cap T) \geq v(S) + v(T)$ for any two sets $S, T \subseteq N$. Equivalently, a game is supermodular if $v(S \cup \{i\}) - v(S) \leq v(T \cup \{i\}) - v(T)$ whenever $S \subseteq T$ and $i \notin T$; that is, the marginal value that a player i adds to a coalition S is no greater than the marginal value i adds to a coalition $T \supseteq S$.

A weaker notion than supermodularity is *superadditivity*: a game is superadditive if for all *disjoint* sets $S, T \subseteq N$, we have $v(S \cup T) \geq v(S) + v(T)$. In a superadditive game, cooperation is always beneficial, and the "grand coalition" of all players will form. We will assume, at minimum, that a game is superadditive. The question, then, is how the players will divvy up $v(N)$, the value of the grand coalition. An allocation $x = \langle x_1, \ldots, x_n \rangle$ should certainly be *(economically) efficient* (no money is left on the table) and *individually rational* (no player makes less than he could make by acting alone): formally, we require $\sum_i x_i = v(N)$ and $x_i \geq v(\{i\})$ for all i. Solution concepts for coalitional games further refine these requirements.

Some solution concepts are based on the notion that the allocation for player i should be proportional to i's "power" in the game—that is, how much value i creates. The *Shapley value* [37] is one such solution concept:

Definition 1. *The* Shapley value *is the allocation where*

$$x_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(n-|S|-1)!}{n!} \big[v(S \cup \{i\}) - v(S)\big].$$

Given a permutation π ordering the arrival of the players, the marginal contribution of π_j is $v(\{\pi_1, \ldots, \pi_j\}) - v(\{\pi_1, \ldots, \pi_{j-1}\})$; the Shapley value for player i is the average, over all permutations, of the marginal contribution of i to the set of players who arrive before i.

Another type of solution concept for coalitional games is based on the notion that the allocation for player i should be such that i has no incentive to deviate

(by leaving the grand coalition with some subset of other players). The following solution concepts attempt to capture the "stability value" of each player:

Definition 2. *The* core *is the set of allocations x such that $v(S) \leq \sum_{i \in S} x_i$ for every $S \subseteq N$: no coalition is allocated less than the value it generates on its own.*

Definition 3. *The* best threat *of player i against player j is* $\text{bt}_{i,j} = \max_{S: i \in S, j \notin S} \{v(S) - \sum_{k \in S} x_k\}$. *The* kernel *is the set of allocations x such that, for any two players i and j, either* $\text{bt}_{i,j} \geq \text{bt}_{j,i}$ *or* $x_i = v(\{i\})$. *(See Section 3 for some intuition.)*

For supermodular games, the core is nonempty, and coincides with the *stable set* [38] and the *bargaining set* [29]. Furthermore, the kernel in a supermodular game is in fact a single point in the core, and the kernel coincides with the *prekernel* and *nucleolus* [29].

3 Testing Core Membership and Computing the Kernel

The results in this section are not new, but for completeness (and for contrast with the results of Sections 4 and 5), we briefly describe algorithmic results on the core and the kernel in supermodular games.

The core. Consider an allocation x, and suppose that we wish to determine whether x is in the core of the game. How might we do this? Define the function $\tau_x(S) := v(S) - \sum_{i \in S} x_i$. By definition, the allocation x is in the core of the game if and only if $v(S) \leq \sum_{i \in S} x_i$ for every $S \subseteq N$—in other words, if and only if $\max_{S \subseteq N} \tau_x(S) \leq 0$. Thus, to test whether x is in the core, it suffices to compute the maximum value of τ_x over all subsets of N.

It is easy to verify that if $v(S)$ is supermodular, then so too is $\tau_x(S)$. We can therefore use the efficient combinatorial algorithms for supermodular function maximization (equivalently, submodular function minimization) of Schrijver or Iwata–Fleischer–Fujishige [21, 34] to decide whether an allocation x is in the core of a supermodular game in polynomial time. (This result is not new—see [18, 41]—but it is pleasing that this computation can be done combinatorially.)

The kernel. Consider a pair of players i and j, and a set S with $i \in S$, $j \notin S$. One can think of S as a "threat" that i can make against j: player i could choose to abandon the grand coalition, and play with just S instead, generating $v(S)$ value for the deviating coalition S. To convince a player $k \in S$—who used to be getting x_k according to the allocation x from the grand coalition—to join this new coalition, player i would need to continue to pay him x_k. This threat's value to player i is exactly $\tau_x(S) = v(S) - \sum_{k \in S} x_k$—the generated value less the costs to pay the members of S their x-values. This value is the gain (or loss, if this quantity is negative) that i would experience when going with coalition S instead of sticking with her original allocation value x_i.

The best threat that i can make against j is $\text{bt}_{i,j} = \max_{S: i \in S, j \notin S} \tau_x(S)$. We say that i is *susceptible to threats* if $x_i > v(\{i\})$ (otherwise, i can easily

make $v(\{i\})$ by "going it alone"). For an allocation x to be in the kernel, all threats must be in equilibrium: if i is susceptible to threats, then no player can "out-threaten" him. Formally, the allocation x is in the kernel if the following condition holds: for any two players i and j, if $x_i > v(\{i\})$, then $\mathrm{bt}_{i,j} \geq \mathrm{bt}_{j,i}$.

We can use the above observations to describe an efficient ellipsoid-based algorithm to compute the kernel (a single point in supermodular games), based on the results of Faigle, Kern, and Kuipers [14]. (The original polynomial-time algorithm is due to Kuipers [25].) Intuitively, given an allocation x that is not in the kernel, we can find two players i and j such that $x_i > v(\{i\})$ and such that j's best threat exceeds that of i; just as in the case of the core, we can find i's best threat against j by maximizing the supermodular function τ_x efficiently. This capability is enough to compute a separating hyperplane that allows us to narrow in on the kernel, using the ellipsoid algorithm.

4 Algorithms to Approximate the Shapley Value

One intuitive explanation of why the kernel of a supermodular game is efficiently computable is that the kernel is fundamentally a robust solution concept. If we take a supermodular game and make an ε change to a single coalition's value, this change would be unlikely to affect the kernel, in that the kernel would change *only if* that altered coalition were a best threat for some player against another. (And there are only $\Theta(n^2)$ best threat coalitions among all 2^n possible subsets.) In contrast, the Shapley value is less robust: a small change in the value of *any single* coalition necessarily alters the Shapley value of *every* player. This difference in robustness suggests that exact computation of the Shapley value may be more difficult, and that we may not be able to hope for more than an approximation to the Shapley value.

We will say that a vector $\overline{s}$ is a (multiplicative) *ε-approximation to the vector s* if $|\overline{s}_i - s_i| \leq \varepsilon s_i$ for all indices i. In this section, we show that there is an oracle-based fully polynomial-time randomized approximation scheme (FPRAS) for the Shapley value, as long as the game is *both* supermodular and monotone. That is, we give a $\mathrm{poly}(n, 1/\varepsilon)$-time randomized algorithm to ε-approximate the Shapley value in monotone supermodular games with high probability. In Section 5, we show that this result is essentially the best possible. Our approach is based on sampling: we compute the marginal value of each player in a random permutation, and average over many permutations. (This type of sampling is also used by Mann–Shapley [27] and Bachrach et al. [3]; however, Mann–Shapley provides no theoretical analysis, and Bachrach et al.'s analysis does not apply to non-simple coalitional games. The comparative difficulty here is that our game may have payoffs besides 0 and 1, and so there is not an immediate bound on the variance of the sampling.)

Algorithm SV-Sample [27]. Given an n-player game v and $\varepsilon > 0$:
Generate $m = 4n(n-1)/\varepsilon^2$ random permutations of the players $\{1, \ldots, n\}$. For each player, define $\overline{s}_i$ to be the average marginal contribution of player i over these m permutations. Return the vector $\overline{s}$.

Theorem 4. *There is an FPRAS for the Shapley value of any game v that is both supermodular and monotone. In particular, Algorithm* SV-Sample(v, ε) *produces an ε-approximation to the Shapley value with probability at least* $3/4$.

Proof. Let X_i denote the marginal contribution of player i in a random permutation. Because v is both supermodular and monotone, we have $X_i \geq 0$. By definition, the Shapley value for player i is $s_i := E[X_i]$. By supermodularity, the maximum possible value achieved by X_i occurs when i is the last player in the permutation, which happens in a $1/n$ fraction of permutations. Thus X_i achieves its maximum value with probability at least $1/n$, and so X_i is at most $n \cdot s_i$.

To upper bound the variance of X_i, we first define a new random variable Y_i that is $n \cdot s_i$ with probability $1/n$ and 0 otherwise. Note that the variances of X_i and Y_i satisfy $\sigma^2_{X_i} \leq \sigma^2_{Y_i}$, because X_i and Y_i have the same expectation, and we have simply pushed individual values to the extremes as much as possible in Y_i. Therefore we have

$$\sigma^2_{X_i} \quad \leq \quad \sigma^2_{Y_i} \quad = \quad E[Y_i^2] - E[Y_i]^2 \quad = \quad \frac{1}{n}(n \cdot s_i)^2 - s_i^2 \quad = \quad (n-1) \cdot s_i^2.$$

Compute the sample mean $\overline{s}_i = \frac{1}{m} \sum_{j=1}^{m} X_i^{(j)}$, where each $X_i^{(j)}$ is an independent trial as above. Now

$$\sigma^2_{\overline{s}_i} = \sigma^2_{X_i}/m \leq (n-1)s_i^2/m \qquad \text{and} \qquad E[\overline{s}_i] = s_i.$$

Using Chebyshev's inequality [32], we have

$$\Pr[|\overline{s}_i - s_i| \geq \varepsilon \cdot s_i] \quad \leq \quad \frac{\sigma^2_{\overline{s}_i}}{s_i^2 \varepsilon^2} \quad \leq \quad \frac{(n-1)s_i^2/m}{s_i^2 \varepsilon^2} \quad = \quad \frac{n-1}{m \cdot \varepsilon^2}.$$

Taking a union bound, we have that

$$\Pr[\exists i : |\overline{s}_i - s_i| \geq \varepsilon \cdot s_i] \quad \leq \quad \frac{n(n-1)}{m\varepsilon^2}.$$

Because we defined $m = 4n(n-1)/\varepsilon^2$, this upper bound on the failure probability is $1/4$. Thus $\overline{s}$ is an ε-approximation to s with probability at least $3/4$. □

The choice of $3/4$ as the success probability in Theorem 4 was arbitrary. By rerunning Algorithm SV-Sample $\Theta(\log(1/\delta))$ times, taking the coordinate-wise median value for each player, and rescaling the resulting vector to preserve economic efficiency (i.e., ensuring that $\sum_{i=1}^{n} \overline{s}_i$ and $v(N)$ are equal), we get an ε-approximation to s with failure probability at most δ.

5 Lower Bounds for Approximating Shapley Value

In this section, we prove that the randomized approximation scheme from Section 4 is the best possible, in several senses: no deterministic algorithm can do

NOTE:
It is our understanding that the theorem as we originally stated it is essentially correct, but a step that we claimed in the original proof of our theorem was erroneous, and our claimed proof was therefore incorrect. A correct proof of a version of this theorem appears in Sasan Maleki's 2015 Ph.D. thesis. We apologize for the error.

Maleki, Sasan (2015) "Addressing The Computational Issues of the Shapley Value With Applications in The Smart Grid," University of Southampton, Faculty of Physical Sciences and Engineering Electronics and Computer Science, PhD Thesis.

as well, a randomized algorithm can do no better, and *both* the monotonicity and supermodularity conditions are required to achieve this approximation.

We will use the following class of n-player supermodular games for several of the lower bounds, for an even number n. Let $\mathcal{C}$ be a collection of subsets of $\{1, \ldots, n\}$, each of cardinality $n/2$. Define the game $v_{\mathcal{C}}$ as follows:

$$v_{\mathcal{C}}(A) = \begin{cases} 2|A| - n & \text{if } |A| > n/2 \\ 1 & \text{if } |A| = n/2 \text{ and } A \in \mathcal{C} \\ 0 & \text{otherwise.} \end{cases}$$

In other words, no coalition of fewer than half the players can receive any value, only some coalitions of size exactly $n/2$ (those in $\mathcal{C}$) receive some value, and larger coalitions receive linearly increasing value as the size grows (regardless of membership).

By examining each player's marginal contributions, we can see that $v_{\mathcal{C}}$ is supermodular. The Shapley value of each player in the game $v_\emptyset$ is 1, and the games $v_\emptyset$ and $v_{\mathcal{C}}$ differ only on the sets in $\mathcal{C}$. Thus, writing $\mathcal{A}_i = \{A : |A| = n/2 \text{ and } i \in A\}$ and $\overline{\mathcal{A}}_i = \{A : |A| = n/2 \text{ and } i \notin A\}$, we have

$$\text{the Shapley value for player } i \text{ in } v_{\mathcal{C}} \quad = \quad 1 + \frac{|\mathcal{C} \cap \mathcal{A}_i|}{\binom{n-1}{n/2-1} \cdot n} - \frac{|\mathcal{C} \cap \overline{\mathcal{A}}_i|}{\binom{n-1}{n/2} \cdot n}.$$

A fully polynomial-time deterministic approximation scheme (FPTAS) is the deterministic analog to an FPRAS. Our next result says that the randomization used in Theorem 4 is in fact necessary: there is no FPTAS for Shapley value in monotone supermodular games. (That is, there is no poly$(n, 1/\varepsilon)$-time deterministic algorithm to ε-approximate Shapley value in n-player monotone supermodular games.)

Theorem 5. *There is no FPTAS for the Shapley value, even for games that are both supermodular and monotone.*

Proof. Assume that such an algorithm exists. For any n, we take $\varepsilon = 1/2n$. The algorithm must ε-approximate a player i's Shapley value with only poly(n) oracle calls. Define $\mathcal{A}_i = \{A : |A| = n/2 \text{ and } i \in A\}$. Assume that the oracle responds to all queries as if the game is $v_\emptyset$, and let $\mathcal{Q}_i \subseteq \mathcal{A}_i$ be the collection of sets among $\mathcal{A}_i$ queried by the algorithm. Then these queries cannot distinguish between the games $v_\emptyset$ and $v_{\mathcal{A}_i \setminus \mathcal{Q}_i}$. The Shapley values for player i in these two games are

$$1 \qquad \text{and} \qquad 1 + \frac{1}{n} - \frac{|\mathcal{Q}_i|}{\binom{n-1}{n/2-1} n},$$

respectively. Because $|\mathcal{Q}_i|$ is polynomial in n, and $\binom{n-1}{n/2-1}$ grows faster than any polynomial, we may take n large enough so that $|\mathcal{Q}_i| / \binom{n-1}{n/2-1} < 1/2$. In this case, the purported algorithm cannot distinguish between two games whose Shapley values differ by a multiplicative factor of $\varepsilon = 1/2n$, as was required. Therefore, such an algorithm cannot exist. □

The randomized sampling algorithm from Section 4 requires $\text{poly}(n, 1/\varepsilon)$ time to ε-approximate Shapley values. In other words, for any polynomial $q(m)$, we can get a $1/q(m)$ approximation in $\text{poly}(n, m)$ time. One might hope for a better algorithm—for example, a $1/2^m$ approximation in $\text{poly}(n, m)$ steps. We now show that no such algorithm exists.

Theorem 6. *Suppose $\varepsilon(m)$ is a function that converges to zero faster than $1/q(m)$ for any polynomial $q(m)$. Then no randomized algorithm can $\varepsilon(m)$-approximate the Shapley value of an n-player monotone supermodular game in $\text{poly}(n, m)$ time.*

Proof. Yao's Minimax Principle [42] states that it suffices to prove that no deterministic polynomial-time algorithm can give an $\varepsilon(m)$ approximation on any particular probability distribution of games. We define the distribution as follows. Let i be a particular player, let $\mathcal{A}_i = \{A : |A| = n/2 \text{ and } i \in A\}$, and let $k = \varepsilon(n)\binom{n-1}{n/2-1}n$. (Assume that n is large enough that $\varepsilon(n) < 1/n$.) With probability $1/2$, we choose the game $v_\emptyset$, and with probability $1/2$ we choose uniformly at random a subcollection $\mathcal{Q}_i$ of $\mathcal{A}_i$ of exactly k sets. The respective Shapley values for player i in $v_\emptyset$ and $v_{\mathcal{Q}_i}$ are

$$1 \qquad \text{and} \qquad 1 + \frac{k}{\binom{n-1}{n/2-1}n} = 1 + \varepsilon(n).$$

Thus the algorithm must be able to distinguish $v_\emptyset$ and $v_{\mathcal{Q}_i}$ with probability $3/4$. But the only way to differentiate is to query a set that is in $\mathcal{Q}_i$. The probability of querying a set in $\mathcal{Q}_i$ in one query is $k/\binom{n-1}{n/2-1} = n\varepsilon(n)$, and the probability of querying a set in $\mathcal{Q}_i$ in $p(n)$ queries is at most $p(n) \cdot n\varepsilon(n)$. As $1/\varepsilon(n)$ eventually exceeds any polynomial and the number of queries $p(n)$ is polynomial in n, this probability approaches zero, contradicting the requirement that the algorithm distinguish $v_\emptyset$ and $v_{\mathcal{Q}_i}$ with probability $3/4$. □

Finally, we prove that both the supermodularity and monotonicity conditions are required, in a very strong sense: with only one of the two properties, no polynomial-time algorithm can distinguish a zero from a nonzero Shapley value (either deterministically or probabilistically). Therefore, there is no ε-approximation algorithm that runs in polynomial time, for any $\varepsilon > 0$.

Theorem 7. *No polynomial-time (deterministic or randomized) algorithm can determine whether the Shapley value of a supermodular game is nonzero.*

Proof. First we prove the deterministic version. For a given collection $\mathcal{C}$ of subsets of size $n/2$, define a new game $v'_\mathcal{C}$ by $v'_\mathcal{C}(A) = v_\mathcal{C}(A) - |A|$ for all subsets A. Each player's Shapley value is decreased by 1 under this transformation. Such a game is still supermodular, but $v'_\mathcal{C}$ is not monotone, because $v'_\mathcal{C}(\emptyset) = 0 > -1 = v'_\mathcal{C}(\{i\})$ for any player i. Suppose that the oracle answers queries as if the game is $v'_\emptyset$, in which every player has Shapley value 0. Because the algorithm can make only $\text{poly}(n)$ oracle calls and there are $\binom{n}{n/2}$ sets of size $n/2$, one of these sets, A, has

not been queried (for sufficiently large n). Then in the game $v'_{\{A\}}$ each player has nonzero Shapley value—the players in A have positive Shapley value, those not in A have negative Shapley value—but the algorithm cannot distinguish $v'_{\{A\}}$ from $v'_{\emptyset}$ based on its oracle calls.

For the randomized version, we use Yao's Minimax Principle, as in Theorem 6. For the random distribution, with probability $1/2$ we take $v'_{\emptyset}$, and otherwise we take $v'_{\{A\}}$ for a set A of size $n/2$ chosen uniformly at random. □

Theorem 8. *No polynomial-time (deterministic or randomized) algorithm can determine whether the Shapley value of a monotone game is nonzero, even assuming superadditivity.*

Proof. Fix a player i. Suppose the oracle answers queries as if we have the following monotone, superadditive game: if $|A| > n/2$, or if $|A| = n/2$ and $i \notin A$, then $v(A) = 1$; otherwise $v(A) = 0$. Player i has Shapley value 0 in this game. A polynomial number of oracle calls cannot differentiate this game from a monotone, superadditive game v' where one set B, of size $n/2$ and with $i \notin B$, is changed from value 1 to value 0 (which gives player i nonzero Shapley value in v'). The randomized version follows as before. □

Note that, while Theorem 5 shows that there is no FPTAS for computing the Shapley value of a supermodular game, it is an open question whether there is a PTAS—i.e., a deterministic ε-approximation algorithm that runs in time poly(n) for any *fixed* $\varepsilon > 0$.

6 Ensuring Monotonicity in Supermodular Games

Section 5 shows that computing the Shapley value of a supermodular game, even approximately, is difficult when the game is not monotone. There are, however, two natural transforms that add monotonicity to any supermodular game, while maintaining supermodularity.

Zero-Normalization Transform. Given a coalitional game v, define a new game v_Z where

$$v_Z(A) = v(A) - \sum_{i \in A} v(\{i\}).$$

The zero-normalization transform offsets the value of any coalition A by the value that each member of A would gather alone, be that amount positive or negative. Thus the value of any singleton coalition is normalized to 0. This change does not affect the strategic character of the game, as only the relative utility of a player's options are important. Note that player i's Shapley values in v and in v_Z differ by exactly $v(\{i\})$; that is, the value in v_Z is the share of the *gains* due to cooperation that are allocated to a player. If v is supermodular, then we will show shortly that v_Z is both supermodular and monotone, meaning these Shapley values can be approximated efficiently, using Theorem 4.

Opt-Out Transform. Given a coalitional game v, define a new game v_O where

$$v_O(A) = \max_{S \subseteq A} v(S).$$

The opt-out transform essentially allows players to "opt out" of any coalition and receive zero utility. Thus, whenever $v(\{i\})$ is negative, we can think of $v(\{i\})$ as the cost for player i to participate in the game, and he will do so only if this cost is offset by the benefits of cooperating.

If v is supermodular, then the following lemma shows that the game v_O (like the game v_Z) is both supermodular and monotone, and $v_O(A)$ can be efficiently computed. Thus Shapley values can be efficiently approximated here as well.

Lemma 9. *If v is supermodular, then both v_Z and v_O are supermodular and monotone. Furthermore, we can compute $v_Z(A)$ and $v_O(A)$ in polynomial time.*

Proof. Supermodularity of v_Z follows from the definition. Because the marginal contribution of a player to the empty set is now zero, his marginal contribution to any set is nonnegative (by supermodularity), so v_Z must be monotone. Computation of $v_Z(A)$ is straightforward.

Monotonicity of v_O is immediate by definition. For supermodularity, we use a simpler version of a result of Topkis [40]. For all sets B_1 and B_2 and all subsets $A_1 \subseteq B_1$ and $A_2 \subseteq B_2$, we have

$$\begin{aligned} v_O(B_1 \cup B_2) + v_O(B_1 \cap B_2) &\geq v(A_1 \cup A_2) + v(A_1 \cap A_2) \quad \text{(by definition of } v_O\text{)} \\ &\geq v(A_1) + v(A_2) \qquad\quad \text{(by supermodularity of } v\text{)} \end{aligned}$$

Maximizing the right-hand side over all $A_1 \subseteq B_1$ and $A_2 \subseteq B_2$ gives us

$$v_O(B_1 \cup B_2) + v_O(B_1 \cap B_2) \geq v_O(B_1) + v_O(B_2)$$

as desired. The ability to compute $v_O(A)$ in polynomial time follows from our ability to maximize the supermodular function v [21, 34]. □

To illustrate these two transforms, consider the following coalitional game v. Let G be an undirected graph. The players of v are the vertices of G, and the value of a coalition A is as follows. Every vertex in A pays an activation *cost* c, and gains a *benefit* $b \geq 0$ for each neighbor in A. That is,

$$v(A) = 2b \cdot |E_A| - c \cdot |A|,$$

where E_A is the set of edges induced by the vertex set A. One can verify that the game v is supermodular, even when generalized to weighted costs and benefits.

Notice that v_Z is equivalent to v but with $c = 0$; thus, v_Z is precisely the edge synergy game studied by Deng and Papadimitriou [11], who show that the Shapley value of player i is exactly $\deg(i) \cdot b$. Thus, in v, the Shapley value for player i is $\deg(i) \cdot b - c$.

The game v_O provides another interesting variant of v. In particular, we can think of v_O as a version of v in which we allow players to opt out of a given

coalition A if their participation would incur a net loss. This game appears to be markedly different from v_Z. We can efficiently compute the exact Shapley value of each player when $b \geq 3c/4$. In this case, player i's value is a function of both $\deg(i)$ and the degrees of the nodes within a small radius of i in G. For smaller b, we can approximate Shapley values using the algorithm SV-Sample. Broadly speaking, a player's Shapley value in v_O increases as her degree increases, and it also increases when nodes near her in the network become more valuable. One potentially intriguing way of interpreting Shapley value in v_O is as a new measure of influence of nodes in a network—as in [5, 6, 23, 24], among others.

References

1. A. Archer, J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker. Approximation and collusion in multicast cost sharing. *Games and Economic Behavior*, 47:36–71, 2004.
2. H. Aziz. *Algorithmic and complexity aspects of simple coalitional games*. PhD thesis, University of Warwick, 2009.
3. Y. Bachrach, E. Markakis, E. Resnick, A. D. Procaccia, J. S. Rosenschein, and A. Saberi. Approximating power indices: theoretical and empirical analysis. *Autonomous Agents and Multi-Agent Systems*, 20:105–122, 2010.
4. R. Barua, S. R. Chakravarty, and S. Roy. Measuring power in weighted majority games. Technical report, Department of Economics, Iowa State University, 2007.
5. P. Bonacich. Power and centrality: A family of measures. *American J. of Sociology*, 1987.
6. S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
7. I. J. Curiel, M. Maschler, and S. Tijs. Bankruptcy games. *Zeitschrift für Operations Research*, 31:143 – 159, 1987.
8. M. Davis and M. Maschler. The kernel of a cooperative game. *Naval Research Logistics Quarterly*, 12:223–259, 1965.
9. J. Deegan and E. W. Packel. A new index of power for simple n-person games. *International Journal of Game Theory*, 7(2):113–123, 1978.
10. X. Deng and Q. Fang. Algorithmic cooperative game theory. *Pareto Optimality, Game Theory and Equilibria*, 17(1):159–185, 2008.
11. X. Deng and C. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19(2):257–266, 1994.
12. E. Elkind, L. A. Goldberg, P. W. Goldberg, and M. Wooldridge. On the computational complexity of weighted voting games. *Annals of Mathematics and Artificial Intelligence*, 56(2):109–131, 2009.
13. E. Elkind and D. Pasechnik. Computing the nucleolus of weighted voting games. In *Proc. 20th Symposium on Discrete Algorithms*, pages 327–335, 2009.
14. U. Faigle, W. Kern, and J. Kuipers. On the computation of the nucleolus of a cooperative game. *International J. of Game Theory*, 30(1):79–98, 2001.
15. J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *J. Comput. Syst. Sci.*, 63:21–41, August 2001.
16. D. Felsenthal and M. Machover. The measurement of voting power: Theory and practice, problems and paradoxes. *Public Choice*, 102(3-4):373–376, 2000.
17. D. B. Gillies. Solutions to general non-zero-sum games. *Contributions to the Theory of Games*, 4:47–85, 1959.
18. M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.
19. S. Herzog, S. Shenker, and D. Estrin. Sharing the "cost" of multicast trees: an axiomatic analysis. *IEEE/ACM Trans. Netw.*, 5:847–860, December 1997.
20. S. Ieong and Y. Shoham. Marginal contribution nets: a compact representation scheme for coalitional games. In *Proc. 6th ACM Conference on Electronic Commerce*, 2005.
21. S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. In *Proc. 32nd Symposium on Theory of Computing*, pages 97–106, 2000.

22. K. Jain and V. Vazirani. Applications of approximation algorithms to cooperative games. In *Proc. 33rd ACM Symposium on Theory of Computing*, 2001.
23. L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, March 1953.
24. J. Kleinberg and E. Tardos. Balanced outcomes in social exchange networks. In *Proc. 40th Symposium on Theory of Computing*, 2008.
25. J. Kuipers. A polynomial time algorithm for computing the nucleolus of convex games. Technical Report M 96-12, Maastricht University, 1996.
26. D. Leech. An empirical comparison of the performance of classical power indices. *Political Studies*, 50(1):1–22, 2002.
27. I. Mann and L. S. Shapley. Values of large games, IV: Evaluating the electoral college by Monte-Carlo techniques. Technical report, The Rand Corporation, Santa Monica, CA, 1960.
28. M. Maschler, B. Peleg, and L. S. Shapley. The kernel and bargaining set for convex games. *International Journal of Game Theory*, 1:73–93, 1971.
29. M. Maschler, B. Peleg, and L. S. Shapley. Geometric properties of the kernel, nucleolus, and related solution concepts. *Mathematics of Operations Research*, 4(4):303–338, November 1979.
30. Y. Matsui and T. Matsui. A survey of algorithms for calculating power indices of weighted majority games. *Journal of the Operations Research Society of Japan*, 43(1):71–86, 2000.
31. Y. Matsui and T. Matsui. NP-completeness for calculating power indices of weighted majority games. *Theoretical Computer Science*, 263(1-2):305–310, 2001.
32. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, 1995.
33. B. O'Neill. A problem of rights arbitration from the Talmud. *Mathematical Social Sciences*, 2(4):345 – 371, 1982.
34. A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory B*, 80:346–355, 2000.
35. A. S. Schulz and N. A. Uhan. Approximating the least core value and least core of cooperative games with supermodular costs. Working paper, 2010.
36. A. S. Schulz and N. A. Uhan. Sharing supermodular costs. *Operations Research*, 58(4):1051–1056, 2010.
37. L. S. Shapley. A value for n-person games. In H. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, 1953.
38. L. S. Shapley. Cores of convex games. *International J. of Game Theory*, 1:11–26, 1971.
39. L. S. Shapley and M. Shubik. On the core of an economic system with externalities. *American Economic Review*, 59:678–684, 1969.
40. D. M. Topkis. Minimizing a submodular function on a lattice. *Operations Research*, 26(2):305–321, March–April 1978.
41. D. M. Topkis. *Supermodularity and Complementarity*. Princeton University Press, 1998.
42. A. C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proc. 18th Symposium on Foundations of Computer Science*, pages 222–227, 1977.