Cooperative Games in MultiAgent Systems

Second Workshop – CoopMAS 2011
Workshop 14 of AAMAS-2011
Tapei, Taiwan, May 3^{rd} 2011
Proceedings

Editors
Stéphane Airiau
Yoram Bachrach
Michael Wooldridge

Preface

The use of cooperative game theory to study how agents should cooperate and collaborate, along with the related topic of coalition formation, has received growing attention from the multiagent systems, game theory, and electronic commerce communities. The workshop is intended to focus on topics in cooperation in multi-agent systems, cooperative game theory and cooperative solution concepts, formation of coalitions, negotiation between agents, joint decision making, and voting. The topics of interest of the workshop include:

- Cooperative game theory
- Coalition formation
- Joint decision making and voting
- Representation issues
- Negotiation
- Collaborative filtering
- Market and economics based cooperation
- Interact with humans (negotiation / collaboration)

The workshop should be of interest to researchers in cooperative game theory and coalition formation, as well as to those who examine collaboration between agents, cooperation in multiagent systems and design and implement collaborating agents. We also welcome participants who are interested in applications of cooperative game theory, which include trading agents, sponsored search and recommender systems.

We would like to thank all the authors who submitted papers to the second edition of this workshop as well as all the program committee members for their usefull work. We thank the AAMAS conference for providing us a platform to hold this event. We are thankful to the Easychair website to manage the submissions and reviews. We look forward to a lively workshop with informative discussions and constructive exchange of ideas.

Stéphane Airiau,
Yoram Bachrach,
and Michael Wooldridge
March 2011

Organization

CoopMAS-2011 is co-located with AAMAS-2011 and took place Tapei, Taiwan, May 2nd, 2011.

Program Committee

Georgios Chalkiadakis (University of Southampton, UK)
Edith Elkind (Nanyang Technological University, Singapore)
Piotr Faliszewski (AGH University of Science & Technology, Poland)
Gianluigi Greco (University of Calabria, Italy)
Paul Harrenstein (Technische Universität München, Germany)
Wojtek Jamroga (University of Luxembourg)
Kate Larson (University of Waterloo, Canada)
Vangelis Markakis (Athens University of Economics and Business, Greece)
Tomasz Michalak (University of Southampton, UK)
Maria Polukarov (University of Southampton, UK)
Joel Uckelman (University of Amsterdam, NL)

Organizing Committee

Stéphane Airiau (ILLC – University of Amsterdam, NL)
Yoram Bachrach (Microsoft Research, Cambridge, UK)
Michael Wooldridge (University of Liverpool, UK)

Contents

Hierarchical Simple Games: Representations and Weightedness

Tatiana Gvozdeva[1], Ali Hameed[1] and Arkadii Slinko[1]

Department of Mathematics, The University of Auckland, Private Bag 92019, Auckland, New Zealand
{t.gvozdeva,a.hameed,a.slinko}@auckland.ac.nz

Abstract. In both human and artificial societies some activities are only allowed to coalitions that satisfy certain criteria, e.g., to sufficiently large coalitions or coalitions which involve players of sufficient seniority. Simmons (1988) formalised this idea in the context of secret sharing schemes by defining the concept of a (disjunctive) hierarchical access structure.
The mathematical concept which describe access structures of secret sharing schemes is that of a simple game. In this paper we aim to start a systematic study of hierarchical games, both disjunctive and conjunctive, and our results show that they deserve such a treatment. We prove the duality between disjunctive and conjunctive hierarchical games. We introduce a canonical representation theorem for both and characterise disjunctive hierarchical games as complete games with a unique shift-maximal losing coalition. We give a short combinatorial proof of the Beimel-Tassa-Weinreb characterisation theorem of weighted disjunctive hierarchical games. By duality we get similar theorems for conjunctive hierarchical games.

1 Introduction

In many situations cooperating agents have different status with respect to the activity. In the theory of simple games developed by (Neumann & Morgenstern, 1944) seniority of players is modeled by giving them different weights. Such situation, for example, arise in context of corporate voting when different shareholders have different number of shares. The access structure in secret sharing scheme (Simmons, 1990; Stinson, 1992) can also be modeled by a simple game but in this theory a different approach in defining seniority is often used. To this end (Simmons, 1990) introduced the concept of a hierarchical access structure. Such access structure stipulates that agents are partitioned into m levels and a sequence of thresholds $k_1 < k_2 < \ldots < k_m$ is set such that a coalition is authorised if it has either k_1 agents of the first level or k_2 agents of the first two levels or k_3 agents of the first three levels etc. Consider, for example, the situation of a money transfer from one bank to another. If the sum to be transferred is sufficiently large this transaction must be authorised by three senior tellers or two vice-presidents. However, two senior tellers and a vice-president

can also authorise the transaction. These hierarchical structures are called *disjunctive* since only one of the m conditions must be satisfied for a coalition to be authorised. If all conditions must be satisfied, the hierarchical access structure is called *conjunctive*.

It was shown that these two approaches are seldom equivalent since hierarchical access structures are seldom weighted. Both (Beimel, Tassa, & Weinreb, 2008) and (Farràs & Padró, 2010) characterised weighted disjunctive hierarchical access structures as a part of their characterisation of weighted ideal access structures. They showed that beyond two levels disjunctive hierarchical structures are normally non-weighted. This is extremely interesting from game-theoretic point of view since we now have a natural class of non-weighted access structures and hence simple games. However, the proof of this characterisation in both papers was indirect. They used the fact that hierarchical access structures are ideal (E. F. Brickell, 1990) and well-known relation between ideal secret sharing schemes and matroids (E. Brickell & Davenport, 1990). Conjunctive hierarchical access structures introduced in (Tassa, 2007) have got much less attention. We will use the game-theory methods and terminology and will be talking about hierarchical games, not access structures.

The progress in studying hierarchical games was hindered by absence of their canonical representation which is needed since different values of parameters can give us the same game. In this paper we introduce a canonical representation of hierarchical games and give a short combinatorial proof of the Beimel-Tassa-Weinreb characterisation theorem using the technique of trading transforms developed in (Taylor & Zwicker, 1999). Our statement is slightly more general as it allows for existence of dummy players. We also characterise disjunctive hierarchical games as complete games with a unique shift-maximal losing coalition. Then we prove the duality between disjunctive and conjunctive games. This allows us to characterise weighted conjunctive hierarchical games and obtain their structural characterisation as complete games with a unique shift-minimal winning coalition. The class of complete games with a unique shift-minimal winning coalition was studied on its own right in (Freixas & Puente, 2008) however, they did not notice that the games they study are hierarchical conjunctive games.

2 Preliminaries

The background material on simple games can be found in (Taylor & Zwicker, 1999).

Definition 1. *Let $P = [n] = \{1, 2, \ldots, n\}$ be a set of players and let $\emptyset \neq W \subseteq 2^P$ be a collection of subsets of P that satisfies the following property:*

$$\text{if } X \in W \text{ and } X \subseteq Y, \text{ then } Y \in W. \tag{1}$$

In such case the pair $G = (P, W)$ is called a simple game *and the set W is called the set of* winning coalitions *of G. Coalitions that are not in W are called* losing.

Due to the monotonic property (1) the subset W is completely determined by the set $W_{\min}$ of minimal winning coalitions of G. A player who does not belong to any minimal winning coalition is called a *dummy*. Such player can be removed from any winning coalition without making it losing.

Definition 2. *A simple game $G = (P, W)$ is called* weighted majority game *if there exist nonnegative weights $w_1, \ldots, w_n$ and a threshold q such that*

$$X \in W \Longleftrightarrow \sum_{i \in X} w_i \geq q. \tag{2}$$

In secret sharing weighted threshold access structures were introduced by (Shamir, 1979).

A distinctive feature of many games is that the set of players is partitioned into subsets and players in each of the subsets have equal status. We suggest analysing such games with the help of multisets. Given a simple game G we define a relation $\sim_G$ on P by setting $i \sim_G j$ if for every set $X \subseteq P$ not containing i and j

$$X \cup \{i\} \in W \Longleftrightarrow X \cup \{j\} \in W. \tag{3}$$

Lemma 1. *$\sim_G$ is an equivalence relation.*

Example 1. Suppose we have $P = \{a_1, a_2, b_1, c_1\}$ as the full set of players with weights as follows: a_1 and a_2 have weights 1, b_1 has weight 2 and c_1 has weight 3. Then the following is the set of minimal winning coalitions for the game with $q = 3$.

$$W_{min} = \{\{a_1, b_1\}, \{a_2, b_1\}, \{c_1\}\}.$$

This gives $a_1 \sim_G a_2$ and of course $a_2 \sim_G a_1$ as $\sim$ is symmetric. Since $\sim$ is reflexive, then $a_i \sim a_i$ for $i = 1, 2$, and also $b_1 \sim_G b_1$. Similarly $c_1 \sim_G c_1$. It follows that our equivalence classes are $\{a_1, a_2\}, \{b_1\}$ and $\{c_1\}$.

We need now the notion of a *multiset*.

Definition 3. *A multiset on the set $[m]$ is a mapping $\mu\colon [m] \to \mathbb{Z}_+$ of $[m]$ into the set of non-negative integers. It is often written in the form*

$$\mu = \{1^{k_1}, 2^{k_2}, \ldots, m^{k_m}\},$$

where $k_i = \mu(i)$ is called the multiplicity *of i in μ.*

A multiset $\nu = \{1^{j_1}, \ldots, m^{j_m}\}$ is a submultiset of a multiset $\mu = \{1^{k_1}, \ldots, m^{k_m}\}$, iff $j_i \leq k_i$ for all $i = 1, 2, \ldots, m$. This is denoted as $\nu \subseteq \mu$.

The existence of large equivalence classes relative to $\sim_G$ allows us to compress the information about the game. This is done by the following construction. Let now $G = (P, W)$ be a game and $\sim_G$ be its corresponding equivalence relation. Then P can be partitioned into a finite number of equivalence classes $P = P_1 \cup P_2 \cup \ldots \cup P_m$ relative to $\sim_G$ and suppose $|P_i| = n_i$. Then we put

in correspondence to the set of agents P a multiset $\bar{P} = \{1^{n_1}, 2^{n_2}, \ldots, m^{n_m}\}$. We take our base set P, identify those agents which are equivalent and do not distinguish between them any further. We carry over the game structure to $\bar{P}$ as well by defining the set of submultisets $\bar{W} \subseteq \bar{P}$ by assuming that a submultiset $Q = \{1^{\ell_1}, 2^{\ell_2}, \ldots, m^{\ell_m}\}$ is winning in $\bar{G}$ if a subset of P containing ℓ_i agents from P_i $(i = 1, 2, \ldots, m)$, is winning in G. This definition is correct since the sets P_i are defined in such a way that it does not matter which ℓ_i players from P_i are involved. We will call $\bar{G} = (\bar{P}, \bar{W})$ the *canonical representation* of G.

Definition 4. *A pair $\bar{G} = (\bar{P}, \bar{W})$ where $\bar{P} = \{1^{n_1}, 2^{n_2}, \ldots, m^{n_m}\}$ and $\bar{W}$ is a system of submultisets of the multiset $\bar{P}$ is said to be a* simple game *on $\bar{P}$ if $X \in \bar{W}$ and $X \subseteq Y$ implies $Y \in \bar{W}$.*

So the canonical representation of a simple game on a set of players P is a simple game on the multiset $\bar{P}$. We will omit bars when this does not invite confusion.

Given a game G on a set of players P we may also define a relation $\succeq_G$ on P by setting $i \succeq_G j$ if for every set $X \subseteq U$ not containing i and j

$$X \cup \{j\} \in W \Longrightarrow X \cup \{i\} \in W. \tag{4}$$

It is known as Isbel's desirability relation (Taylor & Zwicker, 1999). The game is called *complete* if $\succeq_G$ is a total (weak) order. We also define the relation $i \succ_G j$ as $i \succeq_G j$ but not $j \succeq_G i$.

Definition 5. *We say that $\bar{G} = (\bar{P}, \bar{W})$ is a weighted majority game if there exist non-negative weights $w_1, \ldots, w_m$ and $q \geq 0$ such that $Q = \{1^{\ell_1}, 2^{\ell_2}, \ldots, m^{\ell_m}\}$ is winning iff $\sum_{i=1}^{m} \ell_i w_i \geq q$.*

If G is weighted, then it is well-known (see, e.g., (Taylor & Zwicker, 1999), p.91) that equivalent players must have equal weights. Hence we obtain

Proposition 1. *A simple game $G = (P, W)$ is a weighted majority game if and only if the corresponding simple game $\bar{G} = (\bar{P}, \bar{W})$ is.*

One of the most interesting classes of complete games are hierarchical games. They can be of two types ((Beimel et al., 2008), (Tassa, 2007)) and will be considered in the next section.

If a game G is complete, then we define *shift-minimal* (δ-minimal in (Carreras & Freixas, 1996)) winning coalitions and shift-maximal losing coalitions. By a *shift* we mean a replacement of a player of a coalition by a less desirable player which did not belong to it. Formally, given a coalition X, player $p \in X$ and another player $q \notin X$ such that $q \prec_G p$ we say that the coalition $(X \setminus \{p\}) \cup \{q\}$ is obtained from X by a *shift*. A winning coalition X is *shift-minimal* if every coalition contained in it and every coalition obtained from it by a shift are losing. A losing coalition Y is said to be *shift-maximal* if every coalition that contains it is winning and there does not exist another losing coalition from which Y can be obtained by a shift.

The definition of a shift in the multiset context must be adapted as follows.

Definition 6. *Let G be a complete simple game on a multiset $P = \{1^{n_1}, \dots, m^{n_m}\}$, where $1 \succ_G 2 \succ_G \dots \succ_G m$. Suppose a submultiset*

$$A' = \{\dots, i^{\ell_i}, \dots, j^{\ell_j}, \dots\}$$

has $\ell_i \geq 1$ and $\ell_j < n_j$ for some $i < j$. Then we will say that the submultiset

$$A' = \{\dots, i^{\ell_i - 1}, \dots, j^{\ell_j + 1}, \dots\}$$

is obtained from A by a shift.

Shift-minimal winning and shift-maximal losing coalitions are then defined straightforwardly.

For $X \subset P$ we will denote its complement $P \setminus X$ by X^c.

Definition 7. *Let $G = (P, W)$ be a simple game and $A \subseteq P$. Let us define subsets*

$$W_{sg} = \{X \subseteq A^c \mid X \in W\},\ W_{rg} = \{X \subseteq A^c \mid X \cup A \in W\}.$$

Then the game $G_A = (A^c, W_{sg})$ is called a subgame *of G and $G^A = (A^c, W_{rg})$ is called a* reduced game *of G.*

Proposition 2. *Every subgame and every reduced game of a weighted majority game is also a weighted majority game.*

Let us discuss briefly duality in games. The dual game of a game $G = (P, W)$ is defined as $G^* = (P, L^c)$. Equivalently, the winning coalitions of the game G^* dual to G are exactly the complements of losing coalitions of G. We have $G = G^{**}$. We note also that, If $A \subseteq P$, then: $(G_A)^* = (G^*)^A$ and $(G^A)^* = (G^*)_A$. Moreover, the operation of taking the dual is known to preserve weightedness. We will also use the fact that Isbel's desirability relation is self-dual, that is $x \succeq_G y$ if and only if $x \succeq_{G^*} y$. All these concepts can be immediately reformulated for the games on multisets.

Let us remind the reader some more facts from the theory of simple games. The sequence of coalitions

$$\mathcal{T} = (X_1, \dots, X_j; Y_1, \dots, Y_j) \tag{5}$$

is called a *trading transform* if the coalitions $X_1, \dots, X_j$ can be converted into the coalitions $Y_1, \dots, Y_j$ by rearranging players. In other words, for any player p the cardinality of the set $\{i \mid p \in X_i\}$ is the same as the cardinality of the set $\{i \mid p \in Y_i\}$. We say that the trading transform $\mathcal{T}$ has length j.

Theorem 1 ((Taylor & Zwicker, 1999)). *A game $G = (P, W)$ is weighted majority game if for no j does there exist a trading transform (5) such that $X_1, \dots, X_j$ are winning and $Y_1, \dots, Y_j$ are losing.*

This theorem gives a combinatorial way to prove the existence of weights for a given game.

Definition 8. *Let $G = (P, W)$ be a simple game. A trading transform (5) where all $X_1, \ldots, X_j$ are winning in G and all $Y_1, \ldots, Y_j$ are losing in G is called* certificate of non-weightedness *for G.*

For complete games the criterion can be made easier to check due to the following result.

Theorem 2 ((Freixas & Molinero, 2009)). *A complete game is weighted majority game if and only if it does not have certificates of non-weightedness (5) such that $X_1, \ldots, X_j$ are shift-minimal winning coalitions and $Y_1, \ldots, Y_j$ are losing coalitions.*

3 Canonical Representations and Duality of Hierarchical Games

Definition 9 (Disjunctive Hierarchical Game). *Suppose the set of players P is partitioned into m disjoint subsets $P = \cup_{i=1}^{m} P_i$ and let $k_1 < k_2 < \ldots < k_m$ be a sequence of positive integers. Then we define the game $H = H_{\exists}(P, W)$ by setting*

$$W = \{X \in 2^P \mid \exists i \left(\left|X \cap \left(\cup_{j=1}^{i} P_i\right)\right| \geq k_i\right)\}.$$

From their definition it follows that any disjunctive hierarchical game H is complete, moreover for any $i \in [m]$ and $u, v \in P_i$ we have $u \sim_H v$. However, for arbitrary values of parameters we cannot guarantee that the canonical representation $\bar{H}$ of H will be defined on the multiset $\bar{P} = \{1^{n_1}, 2^{n_2}, \ldots, m^{n_m}\}$ since it is possible to have less than m equivalence classes. The next theorem shows when this does not happen.

Theorem 3. *Let H be a disjunctive hierarchical game defined on the set of players P partitioned into m disjoint subsets $P = \cup_{i=1}^{m} P_i$, where $n_i = |P_i|$, by a sequence of positive thresholds $k_1 < k_2 < \ldots < k_m$. Then the canonical representation $\bar{H}$ of H has m equivalence classes and, hence, is defined on $\bar{P} = \{1^{n_1}, 2^{n_2}, \ldots, m^{n_m}\}$ if and only if*

(a) $k_1 \leq n_1$, and
(b) $k_i < k_{i-1} + n_i$ for every $1 < i < m$.

When (a) and (b) hold the sequence $(k_1, \ldots, k_{m-1})$ is determined uniquely. Moreover, H does not have dummies if and only if $k_m < k_{m-1} + n_m$; in this case k_m is determined uniquely as well. If $k_m \geq k_{m-1} + n_m$ the last mth level consists entirely of dummies.

Proof. As we know players within each P_i are equivalent. We note that if $k_1 > n_1$, then $P_1 \sim_H P_2$. On the other hand, if $k_1 \leq n_1$, then any k_1 players from P_1 form a winning coalition M_1 which seize to be winning if we replace one of them with a player of P_2 yielding $P_1 \not\sim_H P_2$. Suppose that we know already that $P_{i-1} \not\sim P_i$ for some $i < m$ and that there is a minimal winning coalition

M_{i-1} contained in $\cup_{j=1}^{i-1} P_j$ which intersects P_{i-1} nontrivially and consists of k_{i-1} players. Then, if $k_i \geq k_{i-1} + n_i$, and a coalition $Q \subseteq \cup_{j=1}^{i} P_j$ is winning and has a non-zero intersection with P_i, then we also have $|Q \cap \left(\cup_{j=1}^{i-1} P_j\right)| \geq k_{i-1}$ and hence $Q \cap \left(\cup_{j=1}^{i-1} P_j\right)$ is also winning. Then any player of P_i in Q can be replaced with any player of P_{i+1} without Q becoming losing, i.e., $P_i \preceq_H P_{i+1}$. From the definition of hierarchical game we have $P_i \succeq_H P_{i+1}$, this implies $P_i \sim_H P_{i+1}$. On the other hand, if $k_i < k_{i-1} + n_i$, we see that a minimal winning coalition in $\cup_{j=1}^{i} P_i$ exists which intersects with P_i nontrivially and consists of k_i players. For constructing it we have to take k_i players of the ith level (if they are available) and, if their number is less than k_i add $k_i - n_i$ players from M_{i-1}. We note that the number of players needed to be added is less than k_{i-1} which makes M_i minimal. As above the existence of such coalition this implies $P_i \not\sim_H P_{i+1}$.

The uniqueness of $(k_1, \ldots, k_{m-1})$ (and also k_m in case $k_m < k_{m-1} + n_m$) follows from the fact that these numbers are exactly the cardinalities of minimal winning coalitions in $\bar{H}$.

By $H_\exists(\mathbf{n}, \mathbf{k})$ we will denote the m-level disjunctive hierarchical game canonically represented by $\mathbf{n} = (n_1, \ldots, n_m)$ and $\mathbf{k} = (k_1, \ldots, k_m)$ with $k_m = k_{m-1} + n_m$ in case the last level consists of dummies. Every new level, except maybe the last one adds a new class of minimal winning coalitions.

Corollary 1. *Let $G = H_\exists(\mathbf{n}, \mathbf{k})$ be an m-level disjunctive hierarchical game. Then we have $n_i > 1$ for every $1 < i < m$.*

Proof. If $n_i = 1$ for some $1 < i < m$, then (b) cannot hold.

We note that the first and the last mth level are special, if $k_1 = 1$, then every user of the first level is self-sufficient (passer) and its presence makes any coalition winning and if $k_m \geq k_{m-1} + n_m$, then the mth level consists entirely of dummies.

Definition 10 (Conjunctive Hierarchical Game). *Suppose the set of agents P is partitioned into m disjoint subsets $P = \cup_{i=1}^{m} P_i$ and let $k_1 < \ldots < k_{m-1} \leq k_m$ be a sequence of positive integers. Then we define the game $H_\forall(P, W)$ by setting*

$$W = \{X \in 2^P \mid \forall i \left(\left|X \cap \left(\cup_{j=1}^{i} P_i\right)\right| \geq k_i\right)\}.$$

Theorem 4. *Let $\mathbf{n} = (n_1, \ldots, n_m)$ and $\mathbf{k} = (k_1, \ldots, k_m)$. Then for an m-level hierarchical games $H_\exists(\mathbf{n}, \mathbf{k})^* = H_\forall(\mathbf{n}, \mathbf{k}^*)$ and $H_\forall(\mathbf{n}, \mathbf{k})^* = H_\exists(\mathbf{n}, \mathbf{k}^*)$, where*

$$\mathbf{k}^* = (n_1 - k_1 + 1, n_1 + n_2 - k_2 + 1, \ldots, \sum_{i \in [m]} n_i - k_m + 1).$$

Proof. We will prove only the first equality. As Isbel's desirability relation is self-dual, the canonical representation of $H_\exists(\mathbf{n}, \mathbf{k})^*$ will involve the same equivalence classes and hence it will be defined on the same multiset. Let $\mathbf{k}^* = (k_1^*, k_2^*, \ldots, k_m^*)$. It is easy to see that $k_i^* < k_{i+1}^*$ is equivalent to $k_{i+1} < k_i + n_{i+1}$

so we have $k_1^* < \ldots < k_{m-1}^* \leq k_m^*$ and $k_{m-1}^* = k_m^*$ if and only if $k_m = k_{m-1} + n_m$. So $\mathbf{k}^*$ is well-defined. Consider a losing in coalition $X = \{1^{\ell_1}, 2^{\ell_2}, \ldots, m^{\ell_m}\}$ in $H_\exists(\mathbf{n}, \mathbf{k})$. It satisfies $\sum_{j\in[i]} \ell_j < k_i$ for all $i \in [m]$. Then

$$\sum_{j\in[i]} (n_j - \ell_j) > \sum_{j\in[i]} n_j - k_i,$$

for all $i \in [m]$, and the coalition $X^c = \{1^{n_1-\ell_1}, 2^{n_2-\ell_2}, \ldots, m^{n_m-\ell_m}\}$ satisfies the condition $\sum_{j\in[i]}(n_j - \ell_j) \geq \sum_{j\in[i]} n_j - k_i + 1 = k_i^*$, for all $i \in [m]$. Therefore, X^c is winning in $H_\forall(\mathbf{n}, \mathbf{k}^*)$.

We need also to show that the complement of every winning in $H_\exists(\mathbf{n}, \mathbf{k})$ coalition is losing in $H_\forall(\mathbf{n}, \mathbf{k}^*)$. Consider a coalition $X = \{1^{\ell_1}, 2^{\ell_2}, \ldots, m^{\ell_m}\}$ which is winning in $H_\exists(\mathbf{n}, \mathbf{k})$. It means that there is an $i \in [m]$ such that $\sum_{j\in[i]} \ell_j \geq k_i$. But then the condition

$$\sum_{j\in[i]} (n_j - \ell_j) \leq \sum_{j\in[i]} n_j - k_i < \sum_{j\in[i]} n_j - k_i + 1 = k_i^*$$

holds. Thus, the complement $X^c = \{1^{n_1-\ell_1}, 2^{n_2-\ell_2}, \ldots, m^{n_m-\ell_m}\}$ is losing in $H_\forall(\mathbf{n}, \mathbf{k}^*)$.

We note a certain duality for the second parameter as $\mathbf{k}^{**} = \mathbf{k}$.

Theorem 5. *Let H be a conjunctive hierarchical game defined on the set of agents P partitioned into m disjoint subsets $P = \cup_{i=1}^m P_i$, where $n_i = |P_i|$, by a sequence of positive thresholds $k_1 < \ldots < k_{m-1} \leq k_m$. Then the canonical representation $\bar{H}$ of H has m equivalence classes and, hence, is defined on $\bar{P} = \{1^{n_1}, 2^{n_2}, \ldots, m^{n_m}\}$ if and only if*

(a) $k_1 \leq n_1$, and
(b) $k_i < k_{i-1} + n_i$ for every $1 < i \leq m$.

When (a) and (b) hold the sequence $(k_1, \ldots, k_m)$ is determined uniquely. The last mth level consists entirely of dummies if and only if $k_{m-1} = k_m$.

Proof. This is a direct consequence of duality and Theorem 3. Indeed we have $k_i^* < k_{i-1}^* + n_i$ if and only if $k_{i-1} < k_i$ and $k_1^* \leq n_1$ is equivalent to $k_1 > 0$, $k_1^* > 0$ is equivalent to $k_1 \leq n_1$ and $k_{i-1}^* < k_i^*$ is equivalent to $k_i < k_{i-1} + n_i$.

To prove the second statement we use duality and the fact that $\mathbf{k}^{**} = \mathbf{k}$.

We will need the following two propositions.

Proposition 3. *Let $\mathbf{n} = (n_1, \ldots, n_m)$, $\mathbf{k} = (k_1, \ldots, k_m)$ and $G = H_\exists(\mathbf{n}, \mathbf{k})$. If $\mathbf{n}' = (n_1, \ldots, n_{m-1})$, $\mathbf{k}' = (k_1, \ldots, k_{m-1})$, then $H(\mathbf{n}', \mathbf{k}')$ is a subgame G_A of G for $A = \{m^{n_m}\}$.*

Proposition 4. *Let $\mathbf{n} = (n_1, \ldots, n_m)$, $\mathbf{k} = (k_1, \ldots, k_m)$ and $G = H_\forall(\mathbf{n}, \mathbf{k})$. Suppose $k_1 = n_1$, $\mathbf{n}' = (n_1, \ldots, n_m)$, and $\mathbf{k}' = (k_2 - k_1, \ldots, k_m - k_1)$. Then $H_\forall(\mathbf{n}', \mathbf{k}')$ is a reduced game G^A, where $A = \{1^{n_1}\}$.*

4 Characterisations of Disjunctive Hierarchical Games

Firstly, we will obtain a structural characterisation of hierarchical disjunctive games.

Theorem 6. *The class of disjunctive hierarchical simple games are exactly the class of complete games with a unique shift-maximal losing coalition.*

Proof. Let $G = H_\exists(\mathbf{n}, \mathbf{k})$ be an m-level hierarchical game. If $k_m < k_{m-1} + n_m$, then the following coalition is shift-maximal losing one:

$$M = \{1^{k_1-1}, 2^{k_2-k_1}, \ldots, m^{k_m-k_{m-1}}\}. \tag{6}$$

Indeed, for every $i = 1, 2, \ldots, m$ it has $k_i - 1$ players from the first i levels so any replacement of a player with more influential one makes it winning. If $k_m \geq k_{m-1} + n_m$, then it has to be modified as

$$M = \{1^{k_1-1}, 2^{k_2-k_1}, \ldots, (m-1)^{k_{m-1}-k_{m-2}}, m^{n_m}\}. \tag{7}$$

Suppose now that G is complete, has canonical multiset representation on a multiset $P = \{1^{n_1}, 2^{n_2}, \ldots, m^{n_m}\}$ and has a unique shift-maximal losing coalition $M = \{1^{\ell_1}, 2^{\ell_2}, \ldots, m^{\ell_m}\}$. We claim that $\ell_i < n_i$ for all $1 \leq i < m$. Suppose not. We know there exist a multiset X such that $X \cup \{i\}$ is winning but $X \cup \{i+1\}$ is losing. We first take X to be of maximal possible cardinality first and then shift-maximal with this property. This will make $X \cup \{i+1\}$ shift-maximal losing coalition. Indeed, we cannot add any more elements to X and replacement any element of it with the more influential one makes it winning. Since $X \cup \{i+1\}$ is not equal to M (the multiplicity of i is not at full capacity) we get a contradiction. Hence $\ell_i < n_i$. Then $\{1^{\ell_1}, \ldots, (i-1)^{\ell_{i-1}}, i^{\ell_i+1}\}$ must be winning. Then every coalition with $k_i = \ell_1 + \ldots + \ell_i + 1$ player from the first i levels is winning. Now if $\ell_m = n_m$ we set $k_m = k_{m-1} + n_m$, alternatively we set $k_m = \ell_1 + \ldots + \ell_m + 1$. It is easy to see that G is in fact $H_\exists(\mathbf{n}, \mathbf{k})$.

(Beimel et al., 2008) characterised ideal weighted threshold secret sharing schemes. As part of this characterisation they characterised hierarchical weighted games. However, their proof is indirect and heavily relies upon the connection between ideal secret sharing schemes and matroids. Here we will prove the following theorem which is slightly more general than their Claim 6.5. In secret sharing dummies are not allowed to be present so they get maximum three levels, not four.

Theorem 7. *Let $G = H_\exists(\mathbf{n}, \mathbf{k})$ be an m-level hierarchical simple game. Then G is a weighted majority game iff one of the following conditions is satisfied:*

(1) $m = 1$;
(2) $m = 2$ and $k_2 = k_1 + 1$;
(3) $m = 2$ and $n_2 = k_2 - k_1 + 1$;

(4) $m \in \{2,3\}$ *and* $k_1 = 1$. *When* $m = 3$, G *is weighted if and only if the subgame* $H_\exists(\mathbf{n}', \mathbf{k}')$, *where* $\mathbf{n}' = (n_2, n_3)$ *and* $\mathbf{k}' = (k_2, k_3)$ *falls under (2) or (3);*

(5) $m \in \{2,3,4\}$, $k_m \geq k_{m-1} + n_m$, *and the subgame* $H_\exists(\mathbf{n}', \mathbf{k}')$, *where* $\mathbf{n}' = (n_1, \ldots, n_{m-1})$ *and* $\mathbf{k}' = (k_1, \ldots, k_{m-1})$ *falls under one of the (1) – (4);*

Proof. We will prove this theorem using combinatorial technique of trading transforms. If $k_m \geq k_{m-1} + n_m$, then users of the last level are dummies and never participate in any minimal winning coalition. As a result if there exists a certificate of non-weightedness

$$\mathcal{T} = (X_1, \ldots, X_j; Y_1, \ldots, Y_j) \tag{8}$$

with minimal winning coalitions $X_1, \ldots, X_j$, which exist by Theorem 2, then no dummies may be found in any of the $X_1, \ldots, X_j$, hence they are not participating in this certificate. Hence G is weighted if and only if its subgame $H_\exists(\mathbf{n}', \mathbf{k}')$, where $\mathbf{n}' = (n_1, \ldots, n_{m-1})$ and $\mathbf{k}' = (k_1, \ldots, k_{m-1})$ is weighted. So we reduced our theorem to the case without dummies and in this case we have to prove that G falls under the one of the cases (1)-(4). Let us assume that $k_m < k_{m-1} + n_m$.

If $k_1 = 1$, then every user of the first level is self-sufficient (passer), that is, any coalition with participation of this agent is winning. If a certificate of non-weightedness (8) exists, then a 1 cannot be a member of any set $X_1, \ldots, X_j$ since then it will have to be also in one of the $Y_1, \ldots, Y_j$ and at least one of them will not be losing. Hence G is weighted if and only if its subgame $H_\exists(\mathbf{n}', \mathbf{k}')$, where $\mathbf{n}' = (n_2, \ldots, n_m)$ and $\mathbf{k}' = (k_2, \ldots, k_m)$ is weighted.

Now we assume $k_1 \geq 2$. The case $m = 1$ is trivial. Next we show that if we are restricted to two levels such that condition (5) is not met but any one of the two conditions (2) and (3) is met, then G is weighted. So we assume that $m = 2$ and $k_1 \geq 2$. If $k_2 \geq k_1 + n_2$ we have case (5); so suppose $k_1 \leq n_1$ and $k_2 < k_1 + n_2$. If $k_2 = k_1 + 1$ then this leads to weightedness. Indeed, suppose we have a certificate of non-weightedness (8) with $X_1, \ldots, X_j$ winning and $Y_1, \ldots, Y_j$ losing coalitions. We have then $|X_i| \geq k_1$ and $|Y_i| < k_2$ for all i. Thus we have $|X_i| = |Y_j| = k_1$ for all i, j. Since $|X_i| = k_1$ and winning, it must be $X_i = \{1^{k_1}\}$ for all i. But this will imply that $Y_i = \{1^{k_1}\}$ for all i which is an absurd as Y_i must be losing.

Now we show that $m = 2$ together with $n_2 = k_2 - k_1 + 1$ (we note that by Theorem 3 this is the smallest value that n_2 can take.) implies G is a weighted majority game. Assume towards a contradiction that G is not weighted, then there exists a certificate of non-weightedness (8). If $k_2 = k_1 + 1$, we know G is weighted. So assume that $k_2 \geq k_1 + 2$. The first shift-minimal winning coalition is $\{1^{k_1}\}$. As $k_1 > 1$, it follows that $n_2 < k_2$, which implies that $\{2^{k_2}\}$ is not a legitimate coalition. As $k_2 - n_2 = k_1 - 1$, the second shift-minimal winning coalition is therefore $\{1^{k_2 - n_2}, 2^{n_2}\} = \{1^{k_1 - 1}, 2^{n_2}\}$. There are no other.

In the certificate of non-weightedness (8) we may assume that $X_1, \ldots, X_j$ are shift-minimal, that is of the two types described earlier. It is obvious that no $\{1^{k_1}\}$ can be among $X_1, \ldots, X_j$. Hence $X_1 = \ldots = X_j = \{1^{k_1 - 1}, 2^{n_2}\}$. It is now clear that we cannot distribute all ones and twos among $Y_1, \ldots, Y_j$ so that they are all losing.

Conversely, we show that if all conditions (1)-(3) fail, then G is not weighted. If $m = 2$, this means that $k_2 \geq k_1 + 2$ and $n_2 \geq k_2 - k_1 + 2$. In this case the game possesses the following certificate of non-weightedness:

$$(\{1^{k_1}\}, \{1^{k_1-2}, 2^{k_2-k_1+2}\};$$
$$\{1^{k_1-1}, 2^{\lfloor \frac{k_2-k_1+2}{2} \rfloor}\}, \{1^{k_1-1}, 2^{\lceil \frac{k_2-k_1+2}{2} \rceil}\}).$$

Since $n_2 \geq k_2 - k_1 + 2$, all the coalitions are well-defined. Also, the restriction $k_2 \geq k_1 + 2$ secures that $\lceil \frac{k_2-k_1+2}{2} \rceil \leq k_2 - k_1$ and makes both multisets in the right-hand-side of the trading transform losing.

Now suppose $m \geq 3$, $k_1 \geq 2$ and the condition (5) is not applicable. We may also assume that $k_1 \leq n_1$, $k_2 < k_1 + n_2$ and $k_3 < k_2 + n_3$. Suppose first that $k_3 \leq n_3$. Then, since $k_3 \geq k_2 + 1 \geq k_1 + 2 \geq 4$, the following is a certificate of non-weightedness.

$$(\{1^{k_1}\}, \{3^{k_3}\}; \{1^{k_1-1}, 3^2\}, \{1, 3^{k_3-2}\}).$$

Suppose $k_3 > n_3$. If at the same time $k_3 \leq n_2 + n_3$, then since $k_3 - n_3 < k_2$ we have a legitimate certificate of non-weightedness

$$(\{1^{k_1}\}, \{2^{k_3-n_3}, 3^{n_3}\}; \{1^{k_1-1}, 2, 3\}, \{1, 2^{k_3-n_3-1}, 3^{n_3-1}\}).$$

Finally, if $k_3 > n_3$ and $k_3 > n_2 + n_3$, then the certificate of non-weightedness will be

$$(\{1^{k_1}\}, \{1^{k_3-n_2-n_3}, 2^{n_2}, 3^{n_3}\};$$
$$\{1^{k_1-1}, 2, 3\}, \{1^{k_3-n_2-n_3+1}, 2^{n_2-1}, 3^{n_3-1}\}).$$

All we have to check is that the second coalition of the losing part is indeed losing. To show this we note that $k_3 - n_3 < k_2$ and $k_3 - n_2 - n_3 + 1 < k_2 - n_2 + 1 \leq k_1$. This shows that the second coalition of the losing part is indeed losing and proves the theorem.

5 Characterisations of Conjunctive Hierarchical Games

First we obtain a structural characterization of conjunctive hierarchical games.

Theorem 8. *The class of conjuctive hierarchical simple games is exactly the class of complete games with a unique shift-minimal winning coalition.*

Proof. Let $H_\forall(\mathbf{n}, \mathbf{k})$ be a conjunctive hierarchical game. By Theorem 4, the dual game of $H_\forall(\mathbf{n}, \mathbf{k})$ is a disjunctive hierarchical game $H_\exists(\mathbf{n}, \mathbf{k}^*)$. If we can prove that the class of complete games with a unique shift-minimal winning coalition is dual to the class of complete games with a unique shift-maximal losing coalition, then by Theorem 6 this will be sufficient.

Let $G = (P, W)$ be a simple game with the unique shift-maximal losing coalition S. By definition, S^c is winning in G^*. Let us prove that it is shift-minimal winning coalition. Consider any other coalition X that can be obtained from S^c by a shift in G^*. It means there are players $i \in X$ and $j \notin X$ such that $j \prec_{G^*} i$ and $X = (S^c \setminus \{i\}) \cup \{j\}$. The complement of X is the set $X^c = (S \setminus \{j\}) \cup \{i\}$. Furthermore, $j \prec_G i$. The coalition X^c is winning in G, because there does not exist a losing coalition from which S can be obtained by a shift. Therefore, X is losing in G^* and S^c is shift-minimal. Consider now a subset X of S^c. The complement X^c of X is a superset of S. Hence, X^c is winning in G and X is losing in G^*. Thus, S^c is the shift-minimal winning coalition in G^*.

We claim that S^c is the unique shift-minimal winning coalition in G^*. Assume, to the contrary, there is another shift-minimal winning coalition X in G^*. As we have seen above $X^c S$ would be shift-maximal losing coalition in G and it is different from S, a contradiction.

It is interesting that the class of complete games with a unique shift-minimal winning coalition was studied before (Freixas & Puente, 2008) without noticing that this class is actually the class of conjunctive hierarchical games.

Theorem 9. *Let $G = H_\forall(\mathbf{n}, \mathbf{k})$ be an m-level conjunctive hierarchical simple game. Then G is a weighted majority game iff one of the following conditions is satisfied:*

(1) $m = 1$*;*
(2) $m = 2$ *and* $k_2 = k_1 + 1$*;*
(3) $m = 2$ *and* $n_2 = k_2 - k_1 + 1$*;*
(4) $m \in \{2, 3\}$ *and* $k_1 = n_1$*. When* $m = 3$*, G is weighted if and only if the reduced game* $H_\forall(\mathbf{n}, \mathbf{k})^{\{1^{n_1}\}} = H_\forall(\mathbf{n}', \mathbf{k}')$*, where* $\mathbf{n}' = (n_2, n_3)$ *and* $\mathbf{k}' = (k_2 - k_1, k_3 - k_1)$ *falls under (2) or (3);*
(5) $m \in \{2, 3, 4\}$*,* $k_m = k_{m-1}$*, and the reduced game* $H_\forall^{\{m^{n_m}\}}(\mathbf{n}, \mathbf{k}) = H_\forall(\mathbf{n}', \mathbf{k}')$*, where* $\mathbf{n}' = (n_1, \ldots, n_{m-1})$ *and* $\mathbf{k}' = (k_1, \ldots, k_{m-1})$ *falls under one of the (1) – (4);*

Proof. Theorem straightforwardly follows from Theorem 7, the duality between conjunctive hierarchical games and disjunctive hierarchical game and Proposition 4.

6 Further Research

An interesting question in relation to complete simple games is to find how quickly can dimension grow depending on the number of players (for general games this growth is exponential). Thus it will be of interest to find the dimension of disjunctive hierarchical games or get an upper bound for their dimension. It should be noted that the dimension of conjunctive hierarchical games, as it follows from results of (Freixas & Puente, 2008) and Theorem 8, is rather well-understood and has linear growth. There are a number of interesting algorithmic questions as well.

References

Beimel, A., Tassa, T., & Weinreb, E. (2008). Characterizing ideal weighted threshold secret sharing. *SIAM Journal on Discrete Mathematics*, *22*(1), 360-397.

Brickell, E., & Davenport, D. (1990). On the classification of ideal secret sharing schemes. In G. Brassard (Ed.), *Advances in cryptology crypto 89 proceedings* (Vol. 435, p. 278-285). Springer Berlin / Heidelberg.

Brickell, E. F. (1990). Some ideal secret sharing schemes. In *Proceedings of the workshop on the theory and application of cryptographic techniques on advances in cryptology* (pp. 468–475). New York, NY, USA: Springer-Verlag New York, Inc.

Carreras, F., & Freixas, J. (1996). Complete simple games. *Mathematical Social Sciences*, *32*(2), 139–155.

Farràs, O., & Padró, C. (2010). Ideal hierarchical secret sharing schemes. In D. Micciancio (Ed.), *Theory of cryptography* (Vol. 5978, p. 219-236). Springer Berlin / Heidelberg.

Freixas, J., & Molinero, X. (2009). Simple games and weighted games: A theoretical and computational viepoint. *Discrete Applied Mathematics*, *157*, 1496–1508.

Freixas, J., & Puente, M. A. (2008). Dimension of complete simple games with minimum. *European Journal of Operational Research*, *188*(2), 555-568.

Neumann, J. von, & Morgenstern, O. (1944). *Theory of games and economic behavior*. Princeton University Press.

Shamir, A. (1979). How to share a secret. *Commun. ACM*, *22*, 612–613.

Simmons, G. J. (1990). How to (really) share a secret. In *Proceedings of the 8th annual international cryptology conference on advances in cryptology* (pp. 390–448). London, UK: Springer-Verlag.

Stinson, D. R. (1992). An explication of secret sharing schemes. *Des. Codes Cryptography*, *2*, 357–390.

Tassa, T. (2007). Hierarchical threshold secret sharing. *J. Cryptol.*, *20*, 237–264.

Taylor, A., & Zwicker, W. (1999). *Simple games*. Princeton University Press.

Subsidies, Stability, and Restricted Cooperation in Coalitional Games

Reshef Meir[1], Jeffrey S. Rosenschein[1], and Enrico Malizia[2]

[1] Hebrew University, Jerusalem, Israel
{reshef24,jeff}@cs.huji.ac.il
[2] Università della Calabria, Rende, Italy
gollizilla@tiscali.it

Abstract. Cooperation among automated agents is becoming increasingly important in various artificial intelligence applications. Coalitional (i.e., cooperative) game theory supplies conceptual and mathematical tools useful in the analysis of such interactions, and in particular in the achievement of stable outcomes among self-interested agents. Here, we study the minimal external subsidy required to stabilize the core of a coalitional game. Following the *Cost of Stability* (CoS) model introduced by Bachrach et al. [3], we give tight bounds on the required subsidy under various restrictions on the social structure of the game. We then compare the extended core induced by subsidies with the least core of the game, proving tight bounds on the ratio between the minimal subsidy and the minimal demand relaxation that each lead to stability.

1 Introduction

Transferable utility (TU) coalitional games are commonly used to model interactions where groups of agents differ in the profits that they can guarantee to themselves. Given that a particular coalition is formed (and specifically, the *grand coalition* of all agents), a key question that arises is how to allocate payments.

Various solution concepts have been suggested in recent decades, specifying desired allocations according to criteria of stability and fairness. Due to the increasing ubiquity of automated agents, and in pursuit of cooperative behavior among self-interested entities, such solutions are being studied and applied in multiple areas of AI research (for several recent papers, see [6, 12, 7, 14]).

As a motivating example, consider three companies, A, B, and C, interested in a cooperative advertising campaign. Expected profit increases as more companies cooperate (e.g., due to exposure in multiple media). A joint effort by all three companies will result in a total profit of $\$12$ Million (the *value* of the coalition $\{A, B, C\}$). Alternatively, the campaign can be carried out by just A and B (with profit of $\$10$M), or each company can choose to advertise alone (with profit of $\$4$M). If companies are to cooperate, they must decide how to share the resulting profits.

The *core* is one of the earliest and most attractive solution concepts, and it directly addresses the issue of stability. The core contains all payment allocations (called imputations) that are stable, in the sense that no subgroup of agents could gain more by

"breaking away" from the grand coalition; that is, the payment allocated to the agents of every coalition is at least that coalition's value.

Unfortunately, in many TU games (including our example above) the core is empty, and the game is inherently unstable, as there is always a sub-coalition that is better off apart. Several relaxations of the core have been proposed in order to maintain stability in games with empty cores. One prominent approach is to assume that departing from the grand coalition incurs some cost to the deviating agents, i.e., that coalitions will be satisfied with a payoff that is slightly lower than their value. The *least core* aims to capture the minimal relaxation in coalitions' demands that will enable a stable imputation.

An alternative assumption is that certain coalitions are unlikely to form due to social reasons or other practical limitations (e.g., it may be difficult for a large coalition to coordinate its deviation). Such restrictions can take many forms, and generally make the game more stable, as fewer coalitions are likely to deviate from a proposed allocation. If companies A and B cannot cooperate without C, then the core in our previous example becomes non-empty (by allocating \$4M to each of the companies).

Whereas the two previous relaxations depend on the environment or on the behavior of the agents themselves, a different approach is to stabilize the game with an external monetary intervention. By subsidizing particular outcomes of the game, for example the formation of the grand coalition, an external authority can induce stability. While the injection of sufficiently large subsidies can always guarantee a non-empty core (e.g., if every agent gets more than the highest value in the game), one would naturally like to minimize the intervention. The minimal subsidy that stabilizes the coalitional game is known as its *Cost of Stability* (CoS) [3]. In our advertising example, a subsidy of \$2M allows us to allocate \$5M to A, \$5M to B, and \$4M to C, thereby achieving full cooperation with a stable allocation.

Our contribution. The value of the least core (i.e., the minimal demand relaxation) and the cost of stability can both serve as *measures* of the (in)stability of a given game. This paper answers certain natural questions regarding the conceptual and quantitative relationship between these measures. We prove tight bounds on the ratio between the minimal subsidy (the CoS) and the minimal demand relaxation that each stabilizes the game. In addition, we measure the amount by which several natural restrictions on coalitions reduce the cost of stability.

1.1 Related work

Subsidies have been proposed by several researchers, using different models and names. The reader is referred to the papers mentioned below for additional useful references and motivating examples.

Our work follows the model suggested by Bachrach et al. [3], which studied bounds and computational aspects of the CoS, focusing on the family of *weighted voting games*. That initial work has been extended by several other researchers [20, 15, 2], who addressed the computation of the CoS in various families of TU games, including Network Flow games, Graph games, Connectivity games, Anonymous games, and others.

A model for subsidies was independently suggested by Bejan and Gómez [5], who focused (as we do) on the relationship between subsidies and other solution concepts.

We adopt some of their notation, which is useful in our case as well. However, in their work the additional payment required to stabilize a game is gathered from the participating agents by means of a specific *taxation* system, rather than injected into the game by an external authority. We do not assume any form of taxation.

Particular attention has been devoted in Economics to *expense sharing* games, where agents share the *cost* of a project, rather than its profits (see, for example, [17, 13, 9]). In some of these papers there are additional requirements in addition to stability, whereas we impose none.

Restrictions on the cooperation structure have also been studied extensively, where the specific restriction may depend on the particular application (see [16, 10, 1, 19]). While the interaction with many solution concepts has been explicitly addressed, we are unaware of previous work that aims to quantify the affect of such restrictions on stability.

1.2 Paper structure

Section 2 provides some notation, and gives the formal definition of the Cost of Stability. In Section 3 we compute worst-case bounds on the CoS of TU games with restricted interactions. Our main results are in Section 4, where we study the relationship between the extended core and the least core. We prove tight bounds on the ratio between the minimal subsidy and the minimal relaxation that are each sufficient to stabilize the game, thereby improving on the results of Bachrach et al. [3]. In the final section, we discuss the relationship to some other solution concepts, and propose future directions for research.

This paper is currently under review for IJCAI-11.

2 Preliminaries

We briefly present the definitions required for our model. For more background, see for example [18]. A *transferable utility (TU) coalitional game* is defined by specifying the collective utility that can be achieved by every coalition of agents. Formally, $G = \langle N, v \rangle$, where N is a finite set of agents $N = \{1, \ldots, n\}$, and v is a function $v : 2^N \rightarrow \mathbb{R}$. For a singleton $i \in N$, we write $v(i)$ instead of $v(\{i\})$. The function v is called the *characteristic function* of the game. We assume by convention that $v(\emptyset) = 0$. Also, we restrict our attention in this paper to positive, monotone games unless explicitly stated otherwise. That is $v(S) \geq 0$ for all S, and $v(S) \geq v(S')$ for all $S' \subset S$.

A TU game is called *simple* if $v(S)$ always equals either 0 or 1. Coalitions with $v(S) = 1$ are called *winning* coalitions. A TU game is *superadditive* if for all $S, T \in 2^N$ s.t. $S \cap T = \emptyset$, $v(S \cup T) \geq v(S) + v(T)$.

A *payoff vector* $\mathbf{x} = (x_1, \ldots, x_n)$ (also called a preimputation) divides the gains of the grand coalition among its members, where $\sum_{i \in N} x_i = v(N)$. We call x_i the payoff of agent i, and denote the payoff of a coalition S as $x(S) = \sum_{i \in S} x_i$. We denote the set of all preimputations in G by $\mathbb{X}(G)$.

A preimputation $\mathbf{x} \in \mathbb{X}(G)$ is *individually rational* if no agent i can gain more than x_i by itself, i.e., if $x_i \geq v(i)$ for all $i \in N$. Individually rational preimputations are

called *imputations*. Similarly, a coalition $S \in 2^N$ *blocks* $\mathbf{x} \in \mathbb{X}(G)$, if $x(S) < v(S)$. The *core* of G, denoted $\mathrm{C}(G)$, consists of all imputations that are not blocked by any coalition.

2.1 The least core

Consider a game G with an empty core, and a value $\epsilon > 0$. We define the *weak ϵ-core* of G as

$$\mathrm{WC}_\epsilon(G) = \{\mathbf{x} \in \mathbb{X}(G) : \forall S \in 2^N, x(S) \geq v(S) - \epsilon|S|\}.$$

Clearly for a large enough ϵ, $\mathrm{WC}_\epsilon(G)$ in not empty. We denote by $\epsilon_{\mathbf{W}}(G)$ the smallest ϵ s.t. $\mathrm{WC}_\epsilon(G) \neq \emptyset$. The $\epsilon_{\mathbf{W}}$-core of G is referred to as the *weak least core*, and denoted by $\mathrm{WLC}(G)$.

The *strong ϵ-core* is defined as

$$\mathrm{SC}_\epsilon(G) = \{\mathbf{x} \in \mathbb{X}(G) : \forall S \in 2^N, x(S) \geq v(S) - \epsilon\},$$

and we define $\epsilon_S(G)$ and the *strong least core* (SLC) accordingly.

2.2 The cost of stability

Let $\Delta \geq 0$ be a payment that an external authority is willing to pay the grand coalition, in case such is formed. This induces a new game $G(\Delta) = \langle N, v_\Delta \rangle$, s.t. $v_\Delta(N) = v(N) + \Delta$. The value of all other coalitions remains unchanged. Clearly if Δ is large enough, then $G(\Delta)$ has a non-empty core (e.g., if $\Delta = n \cdot v(N)$). The *Cost of Stability* is defined as

$$\mathrm{CoS}(G) = \min\{\Delta \geq 0 \text{ s.t. } \mathrm{C}(G(\Delta)) \neq \emptyset\}.$$

The game induced by the minimal extra payment is denoted by $\overline{G} = G(\mathrm{CoS}(G))$ (which has a non-empty core).

A preimputation in $G(\Delta)$ is called a *superimputation* of G. A superimputation $\mathbf{x}'$ is an *extension* of the preimputation $\mathbf{x}$ (denoted $\mathbf{x}' \geq \mathbf{x}$), if $x'_i \geq x_i$ for all $i \in N$. The *extended core* consists of all preimputations that can be extended to stable payoff vectors with minimal subsidy. Formally,

$$\mathrm{EC}(G) = \{\mathbf{x} \in \mathbb{X}(G) \text{ s.t. } \exists \mathbf{x}' \geq \mathbf{x}, \mathbf{x}' \in \mathrm{C}(\overline{G})\}.$$

In general, we define for any $\mathbf{x} \in \mathbb{X}(G)$ its cost of stability, as the smallest payment required to extend $\mathbf{x}$ to a stable payoff vector, i.e.,

$$\mathrm{CoS}(\mathbf{x}, G) = \min\{\Delta \geq 0 \text{ s.t. } \exists \mathbf{x}' \geq \mathbf{x}, \mathbf{x}' \in \mathrm{C}(G(\Delta))\}.$$

Clearly $\mathrm{CoS}(\mathbf{x}, G) \geq \mathrm{CoS}(G)$, with equality iff $\mathbf{x} \in \mathrm{EC}(G)$. The extended core in our initial example contains $\mathbf{p} = (5, 5, 2)$, which can be extended to the (minimal) stable superimputation $(5, 5, 4)$. In contrast, $\mathbf{p}' = (3, 4, 5) \notin \mathrm{EC}(G)$, as $\mathrm{CoS}(\mathbf{p}', G) = 3 > 2 = \mathrm{CoS}(G)$.

2.3 Balanced collections and linear programs

The CoS can also be formulated in a closed form, using the Bondareva-Shapley characterization of the core. We use a variant of the theorem that will be used later in Section 4.

Definition 1. *Let D be a collection of coalitions, and denote by $\delta_S \in \mathbb{R}_+$ the coefficient of coalition S. We say that D is a* balanced collection *if there are $\{\delta_S\}_{S\in D}$, such that for every agent i, $\sum_{S\in D: i\in S} \delta_S = 1$.*

A balanced collection D is called minimal, *if there is no $D' \subsetneq D$ s.t. D' is balanced.*

Theorem 1 (Bondareva-Shapley Theorem). *The core of G is non-empty iff all [minimal][3] balanced collections hold $\sum_{S\in D} \delta_S v(S) \leq v(N)$.*

By a simple continuity argument, it follows that in $\overline{G}$ there is at least one [minimal] collection, for which the above holds with an equality. Such collections are called *solutions* of $\overline{G}$. It is easy to verify that for any game G with empty core,

$$\text{CoS}(G) = \max_{\text{balanced } D} \sum_{S\in D} \delta_S v(S) - v(N). \tag{1}$$

Another way to define the game $\overline{G}$ is by a linear program, where $\sum_{i\in N} x_i$ should be minimized, and every constraint corresponds to a coalition (see [3]). The solutions of the dual linear program (whose variables correspond to coefficients of coalitions), coincide with the solutions of $\overline{G}$. See [11] for a detailed discussion on balanced collections, and a proof of Theorem 1.

2.4 The relative CoS

It is sometimes convenient to treat the external payment as a relative fraction of $v(N)$ (as we do in Section 3). We therefore define the *Relative Cost of Stability* as

$$\text{RCoS}(G) = \min\left\{ \frac{v(N)+\Delta}{v(N)} \geq 0 \text{ s.t. } \text{C}(G(\Delta)) \neq \emptyset \right\}.$$

Note that the transformation is straightforward, as $\text{RCoS}(G) = \frac{v(N)+\text{CoS}(G)}{v(N)}$. Trivial bounds on the RCoS are $1 \leq \text{RCoS}(G) \leq n$, and these are tight.

3 Games with Restricted Coalitions

Suppose that there is some given subset of coalitions $\mathcal{T} \subseteq 2^N$ that can deviate (we assume $\mathcal{T}$ contains all singletons). Given a game $G = \langle N, v\rangle$ and a restriction $\mathcal{T}$, we define the restricted game $G|_{\mathcal{T}} = \langle N, v|_{\mathcal{T}}\rangle$, where $v|_{\mathcal{T}}(S) = v(S)$ if $S \in \mathcal{T}$, and 0 otherwise. We emphasize that the restrictions are given exogenously to the game, and do not depend on the value function or the structure of the game.

Clearly, the more we restrict allowed coalitions, the fewer the constraints on allowed imputations, and therefore the core can only expand. This means that such restrictions can only decrease the CoS of the game. We now consider how some natural restrictions affect the (relative) CoS of the game.

[3] There are versions with and without the minimality requirement.

1. Only coalitions of size at most k are allowed, i.e., $\mathcal{T} = \{S \in 2^N : |S| \leq k\}$.
2. N is divided according to some fixed partition $P = \{C_1, C_2, \ldots, C_k\}$, and $\mathcal{T} = \{S \in 2^N : S \subseteq C_j\}$.
3. Relations between agents are described by a (non-directed) communication graph (N, E). A coalition S is allowed only if the subgraph $(S, E|_S)$ is connected.

The third restriction was proposed by Myerson [16], motivated by the approach that members of a coalition in a society are not allowed to communicate through non-members. Note that the second restriction is a special case of the third, where the graph is a block graph.

Without further assumptions on the game, restricting the coalition structure does not give a better bound on the CoS (in the worst case): consider a simple game $G = \langle N, v\rangle$ where all nonempty coalitions win; then RCoS$(G|_{\mathcal{T}}) = n$ even if only singletons are allowed in $\mathcal{T}$. We therefore consider only superadditive games (i.e., the original value function v is superadditive). Superadditivity is known to induce more stability. For example, it has been shown by Demange that if a game is superadditive *and* its set of coalitions $\mathcal{T}$ is restricted to an uncyclic communication graph, then its core is non-empty [8] (i.e., it has RCoS of 1). Further, the following is known.

Theorem 2 (Bachrach et al., full version [4]). *Let G be a superadditive TU game (even without restrictions); then CoS$(G) \leq (\sqrt{n} - 1)\, v(N)$. Equivalently,*

$$\textit{RCoS}(G) \leq \sqrt{n}, \tag{2}$$

and this bound is tight (up to a small additive constant).

Proposition 1. *For any superadditive TU game G:*

1. *If $\mathcal{T} = \{S \in 2^N : |S| \leq k\}$, then RCoS$(G|_{\mathcal{T}}) \leq \min\{k, \sqrt{n}\}$. Also, for $k=2$ a stable superimputation with cost 2 can be found using a greedy algorithm.*
2. *If $\mathcal{T}$ is restricted to subsets of a partition P, then RCoS$(G|_{\mathcal{T}}) \leq \max_{C \in P} \sqrt{|C|}$.*
3. *If $\mathcal{T}$ is restricted to a communication graph which has a single cycle, then RCoS$(G|_{\mathcal{T}}) \leq 2$.*

Moreover, all bounds are tight (up to a small additive constant in 1. and 2.).

Proof. We prove each case separately.

Bounded coalition size, $k = 2$. We construct a superimputation $\mathbf{p}$ using the following algorithm.

Let $S_1 = \{a_1, a_2\}$ be the most expensive coalition in $\mathcal{T}$.
Set $p(a_1) = v(S_1)$.
for $t = 2, 3, \ldots, n$ **do**
  find the most expensive coalition in $\mathcal{T}$ containing x_t, i.e., $S_t = \{a_t, b\}$
  Set $p(a_t) = v(S_t)$.
  Set $a_{t+1} \leftarrow b$.
end for

First observe that $\mathbf{p}$ is a stable superimputation. Let $S = \{a, b\}$ be any coalition. If S was selected in some iteration, then either a or b gets the value of S and would therefore not participate. If S was not selected, then there is some S_t with $v(S_t) \geq v(S)$, and S_t contains one of a, b. Thus one of them is paid $v(S_t)$ and would not participate in S.

It is left to prove that $p(N) \leq 2v(N)$. Clearly $p(N) = \sum_{i \in N} p(i) = \sum_{t=1}^{n} v(S_t)$. Think of $\{S_t\}_{t=1}^{n}$ as nodes in a graph, where an edge connects two coalitions if they intersect. Since S_t is only connected to S_{t-1} and S_{t+1} (when they exist) we get a bipartite graph (L, R), where L contains all coalitions S_t with odd t, and R with even t. Coalitions inside L and R are pairwise disjoint. From superadditivity we have that $p(N)$ holds

$$\sum_{t=1}^{n} v(S_t) = \sum_{A} v(S_t) + \sum_{B} v(S_t) \leq v\left(\bigcup_{A} S_t\right) + v\left(\bigcup_{B} S_t\right),$$

i.e., at most $2v(N)$.

While the greedy algorithm supplies us with a stable superimputation whose value is at most $2v(N)$, it is possible to do better (see next paragraph).

Bounded coalition size, $k > 2$. If $k \geq \sqrt{n}$ then by Theorem 2 we are done. Assume therefore $k < \sqrt{n}$. Consider a balanced collection D which is a solution of $\overline{G}$.

Lemma 1 (Bachrach et al. [3]). *If G is superadditive, then there is a solution D in which any two sets $S, S' \in D$ with nonzero coefficients intersect.*

Thus take any coalition $S \in D$ of size at most k with a nonzero coefficient δ_S. There must be such a set, otherwise all coefficients are 0 (which means we can find a better solution to the dual program).

$$\begin{aligned} p(N) &= \sum_{j \in N} p_j = \sum_{S \in D} \delta_S v(S) && \text{(by duality)} \\ &\leq \sum_{i \in S} \sum_{S : i \in S} \delta_S v(S) \leq v(N) \sum_{i \in S} \sum_{S : i \in S} \delta_S && \text{(Lemma 1)} \\ &= v(N) \sum_{i \in S} 1 = v(N)|S| \leq k v(N). \end{aligned}$$

For tightness, let $q = k - 1$ and $n_q = k^2 - 1 = q^2 + q + 1$. Take a game $G_q = \langle N_q, v_q \rangle$ s.t. $|N_q| = n_q$, and $\text{CoS}(G_q) > \sqrt{n_q} - 1$ (such a game exists by the tightness example in Theorem 2). We now embed G_q in a game $G = \langle N, v \rangle$, where $N = N_q \cup \{k^2, \ldots, n\}$. Set $v(N) = v_q(N_q)$, $v(S) = v_q(S)$ if $S \subseteq N_q$, and $v(S) = 0$ otherwise. We thus have $\text{CoS}(G) = \text{CoS}(G_q) > \sqrt{n_q} - 1 > k - 1$.

Using Lemma 1, it can be shown that when $k = 2$, $\text{CoS}(G|_{\mathcal{T}}) \leq 1.5$ (which is tight).

Partitions. Take any $C \in P$, and the linear constraints induced by its subcoalitions. From Theorem 2 we can satisfy these constraints by paying at most $p(C) \leq \sqrt{|C|} v(C)$.

We set the payoffs of each set C independently in the same manner. As there are no further constraints, $\mathbf{p}$ is stable. Also

$$p(N) = \sum_{C \in P} p(C) \leq \sum_{C \in P} \sqrt{|C|} v(C) \leq \max_{C \in P} \sqrt{|C|} v(N),$$

where the last inequality is due to superadditivity of v.

A single cycle. We construct a stable superimputation p', by paying $v(N)$ to an arbitrary node in the circle, and solve the remaining game as a tree (using Demange's algorithm [8]). While this solution is quite simple, for the worst case it is asymptotically tight:

Consider a simple anonymous game where a coalition wins iff its size is at least $\lceil (n+1)/2 \rceil$, and a communication graph with n nodes connected in a circle. Since the game is symmetric, we have $p_i = p_j = p$, and for smallest winning coalitions S, $v(N) = v(S) \leq p(S) = |S| p = \lceil (n+1)/2 \rceil \, p$.
that is, $p(N) = np$, which equals either $(2 - \frac{2}{n+2}) v(N)$ (for odd n), or $(2 - \frac{1}{n+1}) v(N)$ (for even n). ■

The lower bound example not only has coalitions of size k, but can also be embedded in a communication graph of degree k. We conjecture that this always holds, i.e., that $\text{RCoS}(G|_{\mathcal{T}}) \leq d(\mathcal{T})$, where d is the degree of the communication graph of $\mathcal{T}$.

4 CoS and the Least Core

We use the following lemma (for a proof, see Gilles [11]).

Lemma 2. *Any minimal balanced collection has a size of at most n, and a unique set of balancing coefficients.*

As an immediate corollary we get the following result, which has been independently shown by Malizia et al. [14] using the geometric properties of the game.

Corollary 1. *If the core of G is empty, then there is a set of coalitions of size at most n that are sufficient to determine the emptiness of the core.*

4.1 The strong least core

It trivially holds (see Bachrach et al. [3]) that

$$\epsilon_{\mathbf{S}}(G) \leq \text{CoS}(G) \leq n \cdot \epsilon_{\mathbf{S}}(G). \tag{3}$$

While the upper bound is tight (consider a game where $v(S) = 1$ for all $S \neq \emptyset$), it can be improved when the game is superadditive, as we will see next.

For the results in this section, we use the following construction. Given a game with an empty core G and $\epsilon = \epsilon_{\mathbf{S}}(G)$, define a new game $G_\epsilon = \langle N, v_\epsilon \rangle$, where $v_\epsilon(S) = v(S) - \epsilon$ for all $S \subsetneq N$, and $v_\epsilon(N) = v(N)$. Clearly $\text{C}(G_\epsilon) = \text{SC}_\epsilon(G) = \text{SLC}(G)$.

Theorem 3. *For any superadditive game G, CoS$(G) \leq \sqrt{n} \cdot \epsilon_{\mathbf{S}}(G)$, and this is tight.*

Our proof uses techniques similar to those used in [3]. Moreover, we can derive the $\sqrt{n}$ bound that appears in Theorem 2 (with a small additive factor), since $\epsilon_{\mathbf{S}}(G) \leq v(N)$.

Proof of Theorem 3. From Lemma 1, there is a balanced collection $\langle D, \{\delta_S\}_{S \in D} \rangle$ in which any two sets S and S' with $\delta_S \neq 0$ and $\delta_{S'} \neq 0$ intersect, and $v(N) + \text{CoS}(G) = \sum_{S \in D} \delta_S v(S)$.

Since D is balanced, it must hold by Theorem 1 that

$$\sum_{S \in D} \delta_S (v(S) - \epsilon) = \sum_{S \in D} \delta_S v_\epsilon(S) \leq v(N),$$

and by combining the last equation and (1),

$$\text{CoS}(G) = \sum_{S \in D} \delta_S v(S) - v(N) \leq \epsilon \sum_{S \in D} \delta_S. \tag{4}$$

Lemma 3. $\sum_{S \in D} \delta_S \leq \sqrt{n}$.

Proof. Suppose first that there is a set $T \in D$ with $|T| \leq \sqrt{n}$, $\delta_T > 0$. Any set $S \in D$ with $\delta_S > 0$ contains one of the elements in T. Thus, we have

$$\sum_{S \in D} \delta_S \leq \sum_{i \in T} \sum_{S \in D : i \in S} \delta_S = \sum_{i \in T} 1 = |T| \leq \sqrt{n}.$$

On the other hand, if for any $S \in D$ with $\delta_S > 0$ it holds that $|S| > \sqrt{n}$, we have

$$\begin{aligned} \sqrt{n} \sum_{S \in D} \delta_S &< \sum_{S \in D} |S| \delta_S \leq \sum_{S \in D} \sum_{i \in S} \delta_S \\ &= \sum_{i \in N} \sum_{S \in D : i \in S} \delta_S = \sum_{i \in N} 1 = n, \end{aligned}$$

which also means $\sum_{S \in D} \delta_S \leq \sqrt{n}$. □

From (4) and the lemma,

$$CoS(G) \leq \sum_{S \in D} \delta_S \epsilon \leq \sqrt{n} \epsilon = \sqrt{n} \epsilon_{\mathbf{S}}(G).$$

The tightness follows from the tightness of Theorem 2. I.e., there is a game G in which CoS$(G) \geq (\sqrt{n} - O(1))\, v(N) \geq (\sqrt{n} - O(1))\, \epsilon_{\mathbf{S}}(G)$. ■

Our main result is showing that the lower bound can be improved in the general case. We begin with a simple example. Consider the case of $n = 2$, and suppose there is an empty core. This simply means $v(1) + v(2) > v(1, 2)$. If we define $z = v(1, 2) - (v(1) + v(2))$, then we can easily see that CoS$(G) = z = 2\epsilon_{\mathbf{W}}(G) = 2\epsilon_{\mathbf{S}}(G)$. With more agents, this ratio is generalized as follows.

Theorem 4. *Let G be a game with an empty core. CoS$(G) \geq \frac{n}{n-1} \epsilon_{\mathbf{S}}(G)$, and this bound is tight.*

Proof. For tightness, it is sufficient to consider a simple game where all coalitions of size at least $n-1$ win.

Let G be a game with an empty core. Consider the strong least core of G, i.e., $\text{SLC}(G)$. Let $\epsilon = \epsilon_{\mathbf{S}}(G)$.

Recall the game G_ϵ. Similarly to the argument used in Section 2.3, there is a solution of G_ϵ (a minimal balanced collection) $\langle K, \{\delta_S\}_{S\in K}\rangle$ s.t.

$$\sum_{S\in K} \delta_S v_\epsilon(S) = v(N). \tag{5}$$

W.l.o.g. $N \notin K$. Assume otherwise; then either $K = \{N\}$ or $\{N\} = K' \subsetneq K$ in contradiction to the minimality of K. However, if $K = \{N\}$ is the only balanced collection with equality, then G can be stabilized with $\epsilon' = 0 < \epsilon$, which is a contradiction to the minimality of $\epsilon = \epsilon_{\mathbf{S}}(G)$.

For all $i \in N$, $1 = \sum_{S\in K: i\in S} \delta_S$. Summing over $i \in N$,

$$\begin{aligned} n =& \sum_{i\in N} \sum_{S\in K: i\in S} \delta_S = \sum_{S\in K} \sum_{i\in S} \delta_S = \sum_{S\in K} |S|\delta_S \leq (n-1)\sum_{S\in K} \delta_S, \\ \text{thus } & \sum_{S\in K} \delta_S \geq \frac{n}{n-1}. \end{aligned} \tag{6}$$

By definition, $v_\epsilon(S) = v(S) - \epsilon$, thus

$$\begin{aligned} v(N) &= \sum_{S\in K} \delta_S v_\epsilon(S) = \sum_{S\in K} \delta_S (v(S) - \epsilon) \\ &= \sum_{S\in K} \delta_S v(S) - \epsilon \sum_{S\in K} \delta_S \leq \sum_{S\in K} \delta_S v(S) - \epsilon \frac{n}{n-1}. \end{aligned}$$

By Equations (1) and (6),

$$\text{CoS}(G) \geq \sum_{S\in K} \delta_S v(S) - v(N) \geq \epsilon \frac{n}{n-1}.$$

■

Theorem 4 establish a quantitative relationship between the CoS and the strong least core. However, the relationship could be deeper.

Conjecture 1. For any game G, $\text{SLC}(G) \subseteq \text{EC}(G)$.

In other words, we conjecture that preimputations in the least core are the easiest to stabilize: for any $\mathbf{x} \in \text{SLC}(G)$, $\text{CoS}(\mathbf{x}, G) = \text{CoS}(G)$.

For small games, the conjecture indeed holds.

Proposition 2. *If $n \leq 3$, then SLC$(G) \subseteq$ EC(G).*

We have already seen that when $n = 2$, $\text{SLC}(G), \text{WLC}(G)$ coincide, and are contained in $\text{EC}(G)$. For $n = 3$ there is only a small number of minimal balanced collections, and we can simply go over all the possibilities. We omit the full proof.

4.2 The weak least core

In the weak ϵ-core, every agent in every coalition agrees to lower her demand by ϵ. Instead, we can increase the payoff of each agent by the same amount (see Bejan and Gómez [5]); thus $\text{CoS}(G) = n \cdot \epsilon_{\mathbf{W}}(G)$.

Clearly, this means that $\text{WLC}(G) \subseteq \text{EC}(G)$, as any preimputation $\mathbf{z} \in \text{WLC}(G)$ can be extended to a stable superimputation by adding ϵ to every coordinate.

Moreover, this tight relation allows us to conclude the following bounds from Theorems 3 and 4.

Corollary 2. $(n-1)\epsilon_{\mathbf{W}}(G) \geq \epsilon_{\mathbf{S}}(G)$.

Corollary 3. *For any superadditive game,* $\epsilon_{\mathbf{S}}(G) \geq \sqrt{n} \cdot \epsilon_{\mathbf{W}}(G)$.

5 Discussion

We showed that various restrictions on the interaction of agents can significantly reduce the cost of stability in (superadditive) TU games. While we focused on profit games, we note that similar results hold when we impose restrictions on *expense sharing* games (as in Meir et al. [15]).

We established a tight lower bound for the CoS, in terms of the minimal relaxation that defines the least core. The upper bound is also improved, but only under conditions of superadditivity. Indeed, superadditive games have many attractive properties related to stability and to its computational aspects (see [6, 3, 8]).

5.1 The nucleolus

One difficulty with solution concepts such as the core and its variations is that even when they exist, they usually do not specify a unique imputation.

A unique solution that is highly motivated by the notion of stability is the *nucleolus* (Schmeidler [21]) and its variations. Informally, the (pre)nucleolus is the preimputation that minimizes the dissatisfaction of all coalitions, sorted according to a certain lexicographic order (see [5] for definitions of some variations). Like any other preimputation, we can always stabilize the nucleolus by extending it with sufficient subsidies to a stable superimputation. Aziz et al. [2] offered an alternative way to achieve a stable nucleolus: first extend the core, then compute the nucleolus in the extended game. For the per-capita nucleolus, both solutions coincide, i.e., the per-capita nucleolus of the extended game $\overline{G}$ is an extension of the per-capita nucleolus of G. This arises simply by adding $\epsilon_{\mathbf{W}}(G)$ to every coordinate of the per-capita nucleolus, which is contained in the WLC.

It is an open problem whether the (standard) nucleolus $\text{N}(G)$ has similar properties. Indeed, since it is contained in the SLC, we have that $\text{CoS}(\text{N}(G), G) = \text{CoS}(G)$ in every game for which Conjecture 1 holds. We further conjecture that $\text{N}(\overline{G})$ is a minimal extension of $\text{N}(G)$, which is not entailed by the previous conjecture.

5.2 Future directions

While our results indicate that there is a tight connection between the extended core and other solution concepts, there are many open questions for future research. Beyond the conjectures that we explicitly stated, it would be interesting to explore these relationships in specific families of TU games, such as those that were mentioned in the introduction.

While in the general case (non-superadditive) restricted cooperation cannot guarantee improved stability, it may dramatically reduce required subsidies in certain limited families of TU games. Such combinations are worth studying.

Finally, the relation between the CoS and similar solution concepts in non-TU games (such as the strong price of anarchy) should be studied.

References

1. E. Algaba, J. Bilbao, and J. López. A unified approach to restricted games. *Theory and Decision*, 50:330–345, 2001.
2. H. Aziz, F. Brandt, and P. Harrenstein. Monotone cooperative games and their threshold versions. In *AAMAS-10*, pages 1117–1024, 2010.
3. Y. Bachrach, E. Elkind, R. Meir, D. Pasechnik, M. Zuckerman, J. Rothe, and J. Rosenschein. The cost of stability in coalitional games. In *SAGT-09*, pages 122–134, 2009.
4. Y. Bachrach, E. Elkind, R. Meir, D. Pasechnik, M. Zuckerman, J. Rothe, and J. Rosenschein. The cost of stability in coalitional games. Technical report, arXiv:0907.4385 [cs.GT], ACM Comp. Research Repository, 2009.
5. C. Bejan and J. C. Gómez. Core extensions for non-balanced TU-games. *Int. journal of game theory*, 38(1):3–16, 2009.
6. V. Conitzer and T. Sandholm. Complexity of constructing solutions in the core based on synergies among coalitions. *Journal of Artificial Intelligence*, 170(6):607–619, 2006.
7. V. Dang, R. Dash, A. Rogers, and N. Jennings. Overlapping coalition formation for efficient data fusion in multi-sensor networks. In *AAAI-06*, pages 635–640, July 2006.
8. G. Demange. On group stability in hierarchies and networks. *Journal of Political Economy*, 112:754–778, 2004.
9. N. R. Devanur, M. Mihail, and V. V. Vazirani. Strategyproof cost-sharing mechanisms for set cover and facility location games. *Decision Support Systems*, 39:11–22, 2005.
10. U. Faigle. Cores of games with restricted cooperation. *ZOR - Methods and. Models of Operations Research*, 33:405–422, 1989.
11. R. P. Gilles. *the cooperative game theory of networks and Hierarchies*. Springer-Verlag, 2010.
12. S. Ieong and Y. Shoham. Multi-attribute coalitional games. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 170–179, 2006.
13. N. Immorlica, M. Mahdian, and V. S. Mirrokni. Limitations of cross-monotonic cost sharing schemes. In *SODA-05*, pages 602–611, 2005.
14. E. Malizia, L. Palopoli, and F. Scarcello. Infeasibility certificates and the complexity of the core in coalitional games. In *IJCAI-07*, pages 1402–1407, 2007.
15. R. Meir, Y. Bachrach, and J. S. Rosenschein. Minimal subsidies in expense sharing games. In *SAGT-10*, pages 347–358, 2010.
16. R. B. Myerson. Graphs and cooperation in games. *Mathematics of operations research*, 2(3):225–229, 1977.

17. M. Pál and E. Tardos. Group strategy proof mechanisms via primal-dual algorithms. In *FOCS-03*, pages 584–593, 2003.
18. B. Peleg and P. Sudhölter. *Introduction to the Theory of Cooperative Games*. Kluwer Publishers, 2003.
19. M. A. Pulido and J. Sanchez-Soriano. Characterization of the core in games with restricted cooperation. *European Journal of Operational Research*, 175(2):860–869, 2006.
20. E. Resnick, Y. Bachrach, R. Meir, and J. Rosenschein. The cost of stability in network flow games. In *MFCS-09*, pages 636–650. Springer, 2009.
21. D. Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics*, 17(6):1163–1170, 1969.

Concise Characteristic Function Representations in Coalitional Games Based on Agent Types

Suguru Ueda, Makoto Kitaki, Atsushi Iwasaki, and Makoto Yokoo

Kyushu University,
744, Motooka, Nishi-ku,
Fukuoka 819-0395, Japan,
{ueda, kitaki}@agent.is.kyushu-u.ac.jp, {iwasaki, yokoo}@is.kyushu-u.ac.jp

Abstract. Forming effective coalitions is a major research challenge in AI and multi-agent systems (MAS). Thus, coalitional games, including Coalition Structure Generation (CSG), have been attracting considerable attention from the AI research community. Traditionally, the input of a coalitional game is a black-box function called a characteristic function. A range of previous studies have found that many problems in coalitional games tend to be computationally intractable when the input is a black-box function. Recently, several concise representation schemes for a characteristic function have been proposed. Although these schemes are effective for reducing the representation size, most problems remain computationally intractable.

In this paper, we develop a new concise representation scheme based on the idea of *agent types*. Intuitively, a type represents a set of agents, which are recognized as having the same contribution. This representation can be exponentially more concise than existing concise representation schemes. Furthermore, this idea can be used in conjunction with existing schemes to further reduce the representation size. Moreover, we show that most of the problems in coalitional games, including CSG, can be solved in polynomial time in the number of agents, assuming the number of possible types is fixed.

1 Introduction

Forming effective coalitions is a major research challenge in AI and multi-agent systems (MAS). A coalition of agents can sometimes accomplish things that individual agents cannot or can do things more efficiently. There are two major research topics in coalitional games. The first topic involves partitioning a set of agents into coalitions so that the sum of the rewards of all coalitions is maximized. This problem is called the Coalition Structure Generation problem (CSG) [7, 8]. The second topic involves how to divide the value of the coalition among agents. The theory of coalitional games provides a number of solution concepts, such as the core, the Shapley value, and the nucleolus.

A range of previous studies have found that many problems in coalitional games, including CSG, tend to be computationally intractable. Traditionally,

Table 1. Computational complexities of coalition formation problems and CSG using conventional representations

	Representation schemes		
	Characteristic function	SCG	MC-nets
Core non-empty	exponential	NP-complete [3]	co-NP-hard [4]
Core membership	exponential	linear[1] [3]	co-NP-complete [4]
The Shapley value	exponential	$O(2^{2^n})$[2]	linear[1] [4]
CSG	$O(3^n)$ [7]	NP-hard [6]	NP-hard [6]

Table 2. Computational complexities of coalition formation problems and CSG using type-based representations

	Type-based representation schemes		
	Characteristic function	SCG	MC-nets
Core non-empty	polynomial (Thm. 2)	polynomial (Thm. 8)	polynomial (Thm. 13)
Core membership	$O(n^t)$ (Thm. 3)	$O(n^{2t})$ (Thm. 9)	$O(n^{2t})$ (Thm. 13)
The Shapley value	$O(n^t)$ (Thm. 4)	$O(n^{2t})$ (Thm. 10)	$O(\|R\| \cdot n^{2t})$ (Thm. 14)
CSG	$O(n^{2t})$ (Thm. 5)	$O(n^{2t})$ (Thm. 11)	$O(n^{2t})$ (Thm. 15)

the input of a coalitional game is a black-box function called a characteristic function, which takes a coalition as an input and returns the value of the coalition (or a coalition structure as a whole). Recently, several concise representation schemes for a characteristic function have been proposed, e.g., synergy coalition group (SCG) [3] and marginal contribution nets (MC-nets) [4]. These schemes represent a characteristic function as a set of rules rather than as a single black-box function and can effectively reduce the representation size. However, most problems are still computationally intractable (Table 1).

In this paper, we develop a new concise representation scheme for a characteristic function, which is based on the idea of *agent types*. Intuitively, a type represents a set of agents, which are recognized as having the same contribution. Most of the hardness results in Table 1 are obtained by assuming that all agents are different types. In practice, however, in many MAS application problems, while the number of agents grows, the number of different types of agents remains small. This type-based representation can be exponentially more concise than existing concise representation schemes. Furthermore, this idea can be used in conjunction with existing schemes, i.e., SCG and MC-nets, for further reducing the representation size. We show that most of the problems in coalitional games, including CSG, can be solved in polynomial time in the number of participating agents, assuming the number of possible types t is fixed (Table 2).

[1] These problems can be solved in linear time in the input size.
[2] This bound is not tight. Examining a tight bound is an open problem.

Our idea of using agent types is inspired by the recent innovative work of Shrot *et al.* [9]. They assume that a game is already represented in some concise representation, e.g., SCG. The goal of their work is first to identify agent types and then to efficiently solve problems in coalitional games by utilizing the knowledge of agent types. This approach becomes infeasible when a standard characteristic function representation is used, since there exists no efficient way for identifying agent types.

In contrast to their study, we assume that agent types are explicitly used for describing a characteristic function in the first place. Also, we consider a wider range of problems including CSG. As a result, the overlap between our work and [9] is very small. In Table 2, only two entries, i.e., Core non-empty and the Shapley value for SCG, might be considered as somewhat overlapping, while other topics are not discussed in [9].

A similar representation to our work is introduced in [2]. However, they consider only *simple games*, where the value of a characteristic function is either 0 or 1, while our work considers general representation schemes that can represent any characteristic function. Also, Bachrach and Rosenschein [1] introduce *coalitional skill games*, where the capability of an agent is characterized by its skills. We can consider such skills correspond to agent types. However, they do not assume that the possible types/skills an agent can have are fixed (even if the number of skills is fixed, the combinations of skills are exponential). Thus, their algorithms and complexity results are quite different from ours.

The rest of this paper is organized as follows. First, we define the model of a coalitional game including CSG and describe the notion of agent types introduced by Shrot *et al.* (Section 2). Next, we introduce a new notion of agent types, i.e., recognizable types, and type-based representation and examine the computational complexity of the coalition formation problem (Section 3) and CSG (Section 4). Then, we describe the extensions of the idea by simultaneously using concise representation schemes (Section 5) and show experimental evaluation results (Section 6). Finally, we conclude this paper and describe future works (Section 7).

2 Model

2.1 Coalitional Games and Coalition Structure Generation

Let $A = \{1, 2, \ldots, n\}$ be a set of all agents. The value of a coalition S is given by a characteristic function v. A characteristic function $v : 2^A \to \mathbb{R}$ assigns a value to each set of agents (coalition) $S \subseteq A$. We assume that each coalition's value is non-negative.

Let $x = (x_1, x_2, \ldots, x_n)$ be a payoff vector. A *solution concept* assigns to each coalitional game a set of reasonable payoff vectors. Two of the best-known ones are the core and the Shapley value.

Definition 1. *The core is the set of all payoff vectors x, which satisfy the feasibility condition: $\sum_{i \in A} x_i = v(A)$, and the non-blocking condition: $\forall S \subseteq A, \sum_{i \in S} x_i \geq v(S)$.*

If there exists a blocking coalition S such that $\sum_{i \in S} x_i < v(S)$ holds, then the agents in S have an incentive to collectively deviate from the grand coalition and divide $v(S)$ themselves. In general, the core can be empty or contain a large set of payoff vectors.

Definition 2. *The Shapley value of agent i, ϕ_i, is defined as:*

$$\phi_i = \sum_{S \subseteq A \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!}(v(S \cup \{i\}) - v(S)).$$

One intuitive interpretation of the Shapley value is that it averages an agent's marginal contribution over all possible orders in which the agent may join the coalition.

A coalition structure CS is a partition of A, into disjoint, exhaustive coalitions. More precisely, $CS = \{S_1, S_2, \dots\}$ satisfies the following conditions:

$$\forall i, j \ (i \neq j), \ S_i \cap S_j = \phi, \quad \bigcup_{S_i \in CS} S_i = A.$$

In other words, in CS, each agent belongs to exactly one coalition, and some agents may be alone in their coalitions.

For example, if there exist three agents a, b, and c, then there are seven possible coalitions: $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$, $\{b, c\}$, $\{a, c\}$, $\{a, b, c\}$, and five possible coalition structures: $\{\{a\}, \{b\}, \{c\}\}$, $\{\{a, b\}, \{c\}\}$, $\{\{a\}, \{b, c\}\}$, $\{\{b\}, \{a, c\}\}$, $\{\{a, b, c\}\}$.

The value of a coalition structure CS, denoted as $V(CS)$, is given by:

$$V(CS) = \sum_{S_i \in CS} v(S_i).$$

An optimal coalition structure CS^* is a coalition structure that satisfies the following condition:

$$\forall CS, V(CS^*) \geq V(CS).$$

We say a characteristic function is super-additive, if for any disjoint sets S_i, S_j, $v(S_i \cup S_j) \geq v(S_i) + v(S_j)$ holds. If the characteristic function is super-additive, solving CSG becomes trivial, i.e., the grand coalition is optimal. In this paper, we assume a characteristic function can be non-super-additive.

Example 1. Let there be four agents a, b, c, and d. The characteristic function is given as follows:

$$\begin{array}{lll} v(\{a\}) = 3, & v(\{b\}) = 3, & v(\{c\}) = 2, \\ v(\{d\}) = 2, & v(\{a, b\}) = 6, & v(\{a, c\}) = 5, \\ v(\{a, d\}) = 5, & v(\{b, c\}) = 5, & v(\{b, d\}) = 5, \\ v(\{c, d\}) = 2, & v(\{a, b, c\}) = 8, & v(\{a, b, d\}) = 8, \\ v(\{a, c, d\}) = 5, & v(\{b, c, d\}) = 5, & v(\{a, b, c, d\}) = 5. \end{array}$$

In this case, there exist multiple optimal CSs. For example, $\{\{a, b, c\}, \{d\}\}$ and $\{\{a, b, d\}, \{c\}\}$ are optimal CSs, and the value of these CSs is 10.

2.2 Agent Types

Shrot *et al.* [9] introduced the idea of using *agent types* to reduce the computational complexity of coalition formation problems. If two agents have the same type, their marginal contributions are the same. They introduced two different notions of agent types, i.e., *strategic types* and *representational types*. The former defines types based on the strategic power of the agents, and the latter defines them based on the representation of the game.

Strategic types are defined based on the strategic power (marginal contribution) of each agent, i.e., if two agents are strategically equivalent, they belong to the same (strategic) type.

Definition 3 (Definition 2.1 in [9]). *Agents $i, j \in A$ are* strategically equivalent *if for any coalition S, such that $i, j \notin S : v(S \cup \{i\}) = v(S \cup \{j\})$.*

The notion called *representational type* is introduced to check the equivalence of agents more conveniently based on a concise representation. Agents are representationally equivalent if they only differ in their identifier. If two agents are representationally equivalent, they are also strategically equivalent, but not vice versa.

Shrot *et al.* examined the computational complexity for determining strategic or representational types for several concise representations. They further showed that if the number of types is fixed in these representations, most intractable problems in coalitional games become polynomial.

3 Type-based Characteristic Function Representation

We assume the person who is describing a game has some prior information about the equivalence of agents. Then the person will describe the game by explicitly using the information of the agent types of which he/she is aware. We need another notion of agent types. This is because (i) the information of the person can be partial and he/she is not necessarily aware of all strategic equivalence, and (ii) the equivalence that he/she is aware of is representation-independent. Therefore, we introduce another notion called *recognizable types*.

Definition 4. *Agents $i, j \in A$ are recognizably equivalent if the person who is describing the game (either by a characteristic function or by a concise representation) knows that for any coalition S, such that $i, j \notin S : v(S \cup \{i\}) = v(S \cup \{j\})$.*

From this definition, if two agents are recognizably equivalent, they are also strategically equivalent, but not vice versa. Furthermore, assuming appropriate representation is chosen, if two agents are recognizably equivalent, they are very likely to be representationally equivalent.

Let $T = \{1, 2, \ldots, t\}$ be the set of all recognizable types and n_A^i be the number of agents of type $i \in T$ in the set of all agents A. Also, $n_A = \langle n_A^1, n_A^2, \ldots, n_A^t \rangle$ denotes a vector, where each element represents the number of agents of each type in A.

We represent a characteristic function as follows:

Definition 5. *For a coalition S, the coalition type of S is a vector $n_S = \langle n_S^1, n_S^2, \ldots, n_S^t \rangle$, where each n_S^i is the number of type i agents in S. We denote the set of all possible coalition types as $A^t = \{\langle n^1, n^2, \ldots, n^t \rangle \mid 0 \leq n^i \leq n_A^i\}$. A type-based characteristic function is defined as $v_t : A^t \to \mathbb{R}$.*

From the definition of recognizable equivalence, $\forall S$ and its type n_S, $v(S) = v_t(n_S)$ holds.

Theorem 1. *A type-based characteristic function requires $O(n^t)$ space.*

Proof. It is clear from the fact that $|A^t| = (n_A^1 + 1) \times \ldots \times (n_A^t + 1) < n^t$. □

Example 2. Let agents a, b be type 1 and agents c, d be type 2 in Example 1. A type-based characteristic function representation for Example 1 is given as follows:

$$v_t(\langle 1, 0 \rangle) = 3,\ v_t(\langle 0, 1 \rangle) = 2,\ v_t(\langle 1, 1 \rangle) = 5,$$
$$v_t(\langle 2, 0 \rangle) = 6,\ v_t(\langle 0, 2 \rangle) = 2,\ v_t(\langle 2, 1 \rangle) = 8,$$
$$v_t(\langle 1, 2 \rangle) = 5,\ v_t(\langle 2, 2 \rangle) = 5.$$

For example, the type of coalition $S = \{a, b, c\}$ is $\langle 2, 1 \rangle$ because S contains two agents of type 1 and one agent of type 2. Thus, $v(S) = 8$. Here, the type-based representation defines the value for each of eight possible coalition types, while the standard representation needs to specify the value for each of fifteen possible coalitions. In general, a type-based representation is exponentially more concise than a standard representation.

We say a payoff vector is *symmetric* if all agents with the same type receive an identical amount. We can restrict our attention to *symmetric* payoff vectors without loss of generality (Lemma 3.2 in [9]). A symmetric payoff vector is represented as $\langle x_1, \ldots, x_t \rangle$, where all type i agents receive x_i. We examine the computational complexity of coalition formation problems in this restriction and show that these problems can be solved in polynomial time in the number of agents.

Theorem 2. *If the number of agent types t is fixed, by using a type-based characteristic function, determining whether the core is non-empty can be done in polynomial time in the number of agents n.*

Proof. To check whether the core is non-empty, it is sufficient to confirm whether there exists a symmetric payoff vector that is feasible and not blocked by any coalition. To this end, we construct the following linear programming formula and check whether it has a solution. The variables of the linear program are elements of a symmetric payoff vector $x = \langle x_1, \ldots, x_t \rangle$. The program is as follows:

$$\sum_{1 \leq i \leq t} n_A^i \cdot x_i = v_t(n_A);\ \forall n_S \in A^t,\ \sum_{1 \leq i \leq t} n_S^i \cdot x_i \geq v_t(n_S)$$

From Theorem 1, if the number of agent types t is fixed, the number of constraints in this linear programming formula is polynomial in the number of agents n. Thus, this problem can be solved in polynomial time. □

Theorem 3. *By using a type-based characteristic function representation, determining whether a symmetric payoff vector x is in the core can be done in $O(n^t)$ time.*

Proof. For a given symmetric payoff vector $x = \langle x_1, \ldots, x_t \rangle$, we need to check whether x is feasible, i.e., $\sum_{1 \leq i \leq t} n_A^i \cdot x_i \geq v_t(n_A)$ holds, and $\forall n_S \in A^t$, x is not blocked by S, i.e., $\sum_{1 \leq i \leq t} n_S^i \cdot x_i \geq v_t(n_S)$ holds. Since $|A^t| = O(n^t)$, determining whether a symmetric payoff vector x is in the core can be done in $O(n^t)$ time. □

Theorem 4. *By using a type-based characteristic function representation, computing the Shapley value of any agent can be done in $O(n^t)$ time.*

Proof. The Shapley value averages an agent's marginal contributions over all possible orders in which the agents may join the coalition. Computing the Shapley value for an agent requires the agent's marginal contributions over all possible coalitions that the agent may join. However, the Shapley value of agents with the same type must be the same, since the Shapley value is symmetric. Also, if two coalitions are the same type, the marginal contribution of an agent when joining these coalitions is the same. Thus, we can obtain the Shapley value from the marginal contribution of each type for each $n_S \in A^t$, which can be done in $O(n^t)$ time. □

4 Coalition Structure Generation with Agent Types

In this section, we develop an algorithm for the CSG problem based on knapsack problems [5]. A multidimensional unbounded knapsack problem (MUKP) is the knapsack problem, where the knapsack has multidimensional constraint and multiple copies exist for each item. For each item j, we denote the profit as p_j, the weight of the i-th constraint as w_{ij}, and the number of copies packed in the knapsack as q_j. A MUKP with m items and t constraints of knapsack $c_1, \ldots, c_t$ is formalized as follows:

$$\begin{aligned} &\text{maximize } \textstyle\sum_j p_j q_j \\ &\text{subject to } \textstyle\sum_j w_{ij} q_j \leq c_i,\ i = 1, \ldots, t \\ &\qquad\qquad q_j \geq 0,\ j = 1, \ldots, m \end{aligned}$$

Theorem 5. *By using a type-based characteristic function representation, finding an optimal coalition structure can be done in $O(n^{2t})$ time.*

Proof. We show that a CSG problem with $m = |A^t|$ coalition types and t possible agent types can be formalized as a MUKP with m items and t constraints. Let us assume that one possible coalition type $n_{S_j} \in A^t$ corresponds to item j, where its value p_j is equal to $v_t(n_{S_j})$ and its weight for the i-th constraint is equal to $n_{S_j}^i$. The capacity constraint of knapsack c_i is determined by n_A^i.

Let $z_j[d_1] \ldots [d_t]$ be the optimal solution value for the knapsack problem (CSG) with j coalition types $\{n_{S_1}, \ldots, n_{S_j}\}$ and a capacity constraint of knapsack $c_i = d_i, \forall i \in T$. If $z_{j-1}[d_1] \ldots [d_t]$ is known for all capacity values $0 \leq d_i \leq$

$n_A^i, \forall i \in T$, then we can include another coalition type n_{S_j} and compute the corresponding solutions $z_j[d_1]\dots[d_t]$ using the following recursive formula:

$$z_j[d_1][d_2]\dots[d_t] = \max \begin{cases} z_j[d_1 - n^1_{S_j}][d_2 - n^2_{S_j}]\dots[d_t - n^t_{S_j}] + v_t(n_{S_j}) \\ (\text{if } \forall i \in T, d_i \geq n^i_{S_j}) \\ z_{j-1}[d_1][d_2]\dots[d_t] \end{cases}$$

We can construct a dynamic programming based algorithm from this recursive formula, which takes $O(n^t \times |A^t|) = O(n^{2t})$ steps (see Section 9.3.2 in [5]). Thus, for any fixed t, finding an optimal coalition structure can be done in $O(n^{2t})$ time. □

A similar argument has been done for the winner determination problem in combinatorial auctions with a fixed number of types of items [10]. In fact, a CSG problem can be mapped into that problem by assuming each coalition type corresponds to a bid and that each type of agent corresponds to a type of item.

5 Combining with Concise Representation Schemes

5.1 Type-based SCG

We first show the original definition of SCG [3].

Definition 6. *An SCG consists of a set of pairs of the form:* $(S, v(S))$. *For any coalition* S*, the value of the characteristic function is:* $v(S) = \max\{\sum_{S_i \in p_S} v(S_i)\}$, *where* p_S *is a partition of* S*, i.e., all* S_i *are disjoint and* $\cup_{S_i \in p_S} S_i = S$*, and for all the* S_i*,* $(S_i, v(S_i)) \in SCG$*. To avoid senseless cases that have no feasible partitions, we require that* $(\{a\}, 0) \in SCG$ *whenever* $\{a\}$ *does not receive a value elsewhere in SCG.*

Using this original definition, we can represent only super-additive characteristic functions. To allow for characteristic functions that are not super-additive, Ohta *et al.* [6] slightly modify the definition, i.e., they add the following requirement for partition p_S: $\forall p'_S \subseteq p_S$, where $|p'_S| \geq 2, (\cup_{S_i \in p'_S} S_i, v(\cup_{S_i \in p'_S} S_i))$ is not an element of SCG. We refer to this modified definition as a standard SCG.

Next, we introduce the definition of a type-based SCG.

Definition 7. *A type-based SCG consists of a set of pairs of the form:* $(n_S, v_t(n_S))$. *For any coalition type* n_S*, the value of the characteristic function is defined in a similar way as a standard SCG.*

Theorem 6. *A type-based SCG can represent any characteristic function represented in a standard SCG using at most the same amount of space.*

Proof. The worst case is the situation where recognizable types of all agents are different, i.e., only agent i belongs to the i-th type. In such a case, for each

element of a standard SCG $(S, v(S))$, we create an element of a type-based SCG $(n_S, v_t(n_S))$, where the i-th element of n_S, i.e., n_S^i, is 1 if $i \in S$, otherwise n_S^i is 0. Clearly, this type-based SCG takes exactly the same amount of space as the standard SCG. □

Theorem 7. *A type-based characteristic function is exponentially more concise than a standard SCG for certain games.*

Proof. Let us consider a coalitional game with n agents, where all agents have an identical type and n is even. Let us assume the type-based SCG consists of only one element $(\langle n/2 \rangle, \alpha)$. In the coalitional game, $v(S) = \alpha$ if $n/2 \leq |S| < n$, and $v(S) = 2\alpha$ if $|S| = n$, otherwise, $v(S) = 0$. Let us represent this characteristic function using a standard SCG. In this case, we need to include each coalition S such that $|S| = n/2$ in an SCG. The number of all coalitions with a size of $n/2$ is given as $\binom{n}{\frac{n}{2}} = \frac{n!}{(\frac{n}{2}!)^2}$. Using Stirling's approximation, i.e., $n! \approx \sqrt{2\pi n}\frac{n^n}{e^n}$, we obtain $\binom{n}{\frac{n}{2}} \approx \frac{\sqrt{2\pi n}}{\pi n} 2^n$. Thus, the standard SCG representation requires $O(2^n)$ space to specify this characteristic function. □

Example 3. A type-based SCG for Example 1 is given as follows:

$$(\langle 1, 0 \rangle, 3), (\langle 0, 1 \rangle, 2), (\langle 0, 2 \rangle, 2), (\langle 2, 2 \rangle, 5).$$

In this case, $v_t(\langle 2, 1 \rangle) = v_t(\langle 1, 0 \rangle) + v_t(\langle 1, 0 \rangle) + v_t(\langle 0, 1 \rangle) = 8$. Here, the type-based SCG defines the value of four coalition types, while the type-based characteristic function representation needs to describe eight possible coalition types.

Let us examine the complexity of coalition formation problems when we use the type-based SCG representation. As discussed in [3], core-related coalition formation problems remain hard in a standard SCG. However, this is due to the fact that determining the value of the grand coalition is hard. If the value of the grand coalition is given explicitly, these problems become tractable. We first prove the following lemma.

Lemma 1. *Translating a type-based SCG representation to a type-based characteristic function representation (i.e., obtaining the values of characteristic function for all coalition types that are not explicitly described in SCG) can be done in $O(n^{2t})$ time.*

Proof. Let us consider obtaining $v_t(n_A)$, i.e., the value of the grand coalition. It can be obtained using a method similar to the DP-based algorithm described in Theorem 5. More precisely, we consider each coalition type described in the type-based SCG as an item of the knapsack problem, where the capacity constraint of knapsack $c_i = n_A^i$ for all $i \in T$. By running the DP-based algorithm, for each possible coalition type n_S, we obtain $v_t(n_S)$, which is represented as $z_m[n_S^1] \dots [n_S^t]$. One slight difference with the DP-based algorithm described in Theorem 5 is that, if $v_t(n_S)$ is already described in the type-based SCG explicitly, we fix the value of $z_j[n_S^1] \dots [n_S^t]$ to $v(n_S)$ and do not update. The algorithm can be done in $O(n^{2t})$ time. □

Now, since the value of the grand coalition can be obtained in polynomial time, it is straightforward to show that core-related coalition formation problems are tractable.

Theorem 8. *If the number of agent types t is fixed, by using a type-based SCG representation, determining whether the core is non-empty can be done in polynomial time in the number of agents n.*

Proof. When confirming that a symmetric payoff vector x is not blocked by any coalition type, it is sufficient to check against coalition types that are explicitly described in the type-based SCG. Therefore, we construct the following linear programming formula with the constraint of coalition types described in the type-based SCG. The program is as follows:

$$\sum_{1\leq i\leq t} n_A^i \cdot x_i = v_t(n_A);\ \forall (n_S, v_t) \in SCG,\ \sum_{1\leq i\leq t} n_S^i \cdot x_i \geq v_t(n_S)$$

After we obtain $v_t(n_A)$ (by Lemma 1, this can be done in polynomial time), the above program can be solved in polynomial time in the number of agents n. □

Theorem 9. *By using a type-based SCG representation, determining whether a symmetric payoff vector x is in the core can be done in $O(n^{2t})$ time.* □

Proof. We first obtain $v_t(n_A)$. By Lemma 1, this can be done in $O(n^{2t})$ time. Next, for a symmetric payoff vector $x = \langle x_1, \ldots, x_t \rangle$, we check whether $\sum_{i\in T} n_A^i \cdot x_i = v_t(n_A)$ holds. Then, we confirm that this symmetric payoff vector is not blocked by any coalition. It is sufficient to check against coalition types explicitly described in the type-based SCG. Thus, determining whether a symmetric payoff vector x is in the core can be done in $O(n^{2t})$ time. □

Unfortunately, as far as the authors are aware, there is no efficient way to compute Shapley values using SCG-based representations. However, we can use a naive translation approach, which can be done in polynomial time.

Theorem 10. *If the number of agent types t is fixed, by using a type-based SCG representation, computing the Shapley value can be done in $O(n^{2t})$ time.*

Proof. From Lemma 1, we can compute the values of all coalition types in $O(n^{2t})$ time. Also, Theorem 4 shows that we can compute the Shapley value in $O(n^t)$ time if we know the value of all coalitions. Thus, computing the Shapley value can be solved in $O(n^{2t})$ time. □

Theorem 11. *By using a type-based SCG representation, finding an optimal coalition structure can be done in $O(n^{2t})$ time.*

Proof. Ohta *et al.* showed that there exists a coalition structure CS such that $V(CS) = V(CS^*)$ and $\forall S \in CS, (S, v(S)) \in SCG$ (see Theorem 3 in [6]). Using a similar argument, we can show that when searching CS^*, we need to consider only the coalition types that are explicitly provided in the type-based SCG. We

can find an optimal coalition structure using the DP-based algorithm provided in Theorem 5, where possible coalition types (or items of the knapsack problem) are restricted to the elements appearing in the type-based SCG. This algorithm also takes $O(n^{2t})$ steps. Thus, finding an optimal coalition structure can be done in $O(n^{2t})$ time. □

5.2 Type-based MC-nets

We first show the original definition of MC-nets [4].

Definition 8. *An MC-net consists of a set of rules R. Each rule $r \in R$ is of the form: $(P_r, N_r) \to v_r$, where $P_r \subseteq A, N_r \subseteq A, P_r \cap N_r = \emptyset, v_r \in \mathbb{R}$. v_r can be either positive or negative. We say that rule r is applicable to coalition S if $P_r \subseteq S$ and $N_r \cap S = \emptyset$, i.e., S contains all agents in P_r (positive literals) but no agent in N_r (negative literals). For a coalition S, $v(S)$ is given as $\sum_{r \in R_S} v_r$, where R_S is the set of rules applicable to S.*

Next, we introduce the definition of type-based MC-nets.

Definition 9. *A type-based MC-net consists of a set of rules R. Each rule $r \in R$ is of the form: $(L_r, U_r) \to v_r$, where $L_r = \langle l_r^1, l_r^2, \ldots, l_r^t \rangle$ and $U_r = \langle u_r^1, u_r^2, \ldots, u_r^t \rangle$. Each l_r^i (and u_r^i) represents the lower (upper) bound of the number of i-th type agents in a coalition so that this rule becomes effective. We say that rule r is applicable to coalition S if $\forall i \in T, l_r^i \leq n_S^i \leq u_r^i$. For a coalition S, $v(S)$ is given as $\sum_{r \in R_S} v_r$, where R_S is the set of rules applicable to S.*

Theorem 12. *A type-based MC-net can represent any characteristic function represented in a standard MC-net using at most the same amount of space and is exponentially more concise than a standard MC-net for certain games.*

For space reasons, we omit the proof. We can prove the theorem using a similar argument described in Theorems 6 and 7.

Example 4. A type-based MC-net for Example 1 is given as follows:

$$r_1 : (\langle 1, 0 \rangle, \langle 1, 2 \rangle) \to 3,\ r_2 : (\langle 2, 0 \rangle, \langle 2, 1 \rangle) \to 6,$$
$$r_3 : (\langle 0, 1 \rangle, \langle 2, 2 \rangle) \to 2,\ r_4 : (\langle 2, 2 \rangle, \langle 2, 2 \rangle) \to 3.$$

In this case, r_2 and r_3 are applicable to coalition type $\langle 2, 1 \rangle$, but r_1 and r_4 are not. Thus, $v_t(\langle 2, 1 \rangle)$ is equal to $6 + 2 = 8$. Here, the type-based MC-net consists of four rules, while the type-based characteristic function representation needs to specify the value of eight possible coalition types.

Unfortunately, as far as the authors are aware, there is no efficient way to solve core-related coalition formation problems using MC-net-based representations. However, we can use a naive translation approach, which can be done in polynomial time. We first prove the following lemma.

Lemma 2. *Translating a type-based MC-net representation to a type-based characteristic function representation (i.e., obtaining the values of characteristic function for all coalition types) can be done in $O(n^{2t})$ time.*

Proof. We can safely assume that the number of rules of a type-based MC-net is $O(n^t)$. Otherwise, it's better to use the type-based characteristic function representation in the first place. For each $n_S \in A^t$, we initialize $v_t(n_S)$ to 0. Then, for each rule $r \in R$, and for each coalition type n_S, if the rule is applicable to n_S, we increment $v_t(n_S)$ by the value of r. Since each rule is applicable to at most n^t coalition types, this procedure can be done in $O(n^{2t})$ time. □

Theorem 13. *If the number of agent types t is fixed, by using a type-based MC-net representation, determining whether the core is non-empty can be done in polynomial time in the number of agents n. Also, determining whether a symmetric payoff vector x is in the core can be done in $O(n^{2t})$ time.*

Proof. This is clear since by Lemma 2 we can transform a type-based MC-net representation into a type-based characteristic function representation in $O(n^{2t})$ time. Then, from Theorem 2, we can determine whether the core is non-empty in polynomial time. Also, from Theorem 3, whether a symmetric payoff vector x is in the core can be checked in $O(n^t)$ time. Thus, the total required time is polynomial in the number of agents n (in particular, $O(n^{2t})$ for checking whether x is in the core). □

In contrast to core-related coalition formation problems, the standard MC-net representation is suitable for computing Shapley values. This is also true for our type-based MC-net representation, i.e., the following theorem holds.

Theorem 14. *If the number of agent types t is fixed, by using a type-based MC-net representation, computing the Shapley value of any agent can be done in $O(|R| \cdot n^{2t})$ time.*

Proof. To compute the Shapley value of an agent, we can compute its Shapley value for each rule and use the summation of these values (see Proposition 5 in [4]). Furthermore, we can decompose a rule into multiple rules, where each decomposed rule has a form: $(\langle y^1, \ldots, y^t \rangle, \langle y^1, \ldots, y^t \rangle) \rightarrow v$, i.e., the rule is applicable to exactly one coalition type. The Shapley value of type i agent for rule r (denoted as $\phi_{i,r}$) is computed by the following procedure:

$$\phi_{i,r} = \frac{v}{n!}(f_i^+(y_1, \ldots, y_t) - f_i^-(y_1, \ldots, y_t)),$$

$$f_i^+(y_1, \ldots, y_t) =$$

$$\begin{cases} 0 & \text{if } y_i = 0, \\ \Pi_{j \neq i} \, {}_{n_A^j}C_{y_j} \cdot {}_{n_A^i - 1}C_{y_i - 1} \cdot (s_y - 1)!(n - s_y)! & \text{otherwise.} \end{cases}$$

$$f_i^-(y_1,\ldots,y_t) = \begin{cases} 0 & \text{if } y_i = n_A^i, \\ \Pi_{j\neq i}\ {}_{n_A^j}C_{y_j} \cdot {}_{n_A^i-1}C_{y_i} \cdot (s_y)!(n-s_y-1)! & \text{otherwise.} \end{cases}$$

where $s_y = \sum_{j\in T} y_j,\ n = \sum_{j\in T} n_A^j$.

Here, $f_i^+(y_1,\ldots,y_t)$ represents the number of orderings where the marginal contribution of one type i agent is v, and $f_i^-(y_1,\ldots,y_t)$ represents the number of orderings where the marginal contribution of one type i agent is $-v$. Thus, $\frac{v}{n!}(f_i^+(y_1,\ldots,y_t) - f_i^-(y_1,\ldots,y_t))$ represents the Shapley value of type i agent for this rule. Using this procedure, the required time for computing the Shapley value of an agent becomes $O(|R| \cdot n^{2t})$. □

Although Ohta *et al.* proposed an efficient method for solving CSG problems based on the standard MC-net [6], we cannot apply this method straightforwardly. Nevertheless, we can still rely on a naive approach that translates a type-based MC-net representation into the corresponding type-based characteristic function representation.

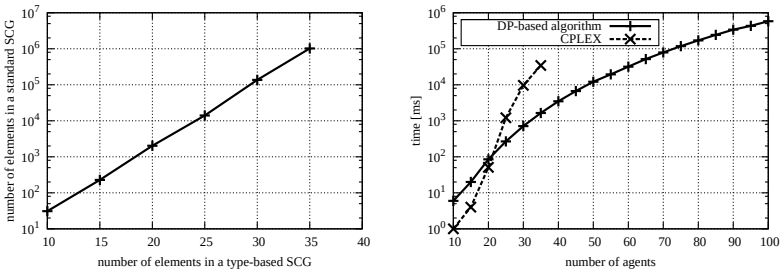
Theorem 15. *If the number of agent types t is fixed, by using a type-based MC-net representation, finding an optimal coalition structure can be done in $O(n^{2t})$ time.*

Proof. This is clear since by Lemma 2 we can transform a type-based MC-net representation into a type-based characteristic function representation in $O(n^{2t})$ time. Then, from Theorem 5, we can find an optimal coalition structure in $O(n^{2t})$ time. Thus, the total required time is $O(n^{2t})$. □

6 Experimental Evaluations

In this section, we experimentally evaluate the performance of our proposed methods. We concentrate on methods for type-based SCG, since we can control the input size of a problem instance. In addition, the DP-based algorithm is also used in other representations. All tests were run on a Core 2 Quad Q9650 3GHz processor with 16GB RAM. The test machine runs Windows 7 Enterprise x64 Edition.

Let us consider a type-based SCG problem instance, where n agents have one of five different types ($t = 5$). We vary n from 10 to 100 and set the number of elements in a type-based SCG to n, (i.e., equal to the number of agents). We generate each element using a decay distribution as follows. Initially, the required number of agents in each type is set to zero. First, we randomly choose one type and increment the required number of agents in the type by one. Then, we repeatedly choose a type randomly and increment its required number of agents with probability α, until a type is not chosen or the required number of agents exceeds the limit. We choose the value of that coalition between 1 and



(a) Number of elements in a standard SCG

(b) Computation time

Fig. 1. Experimental Results

$10 \times n$ uniformly at random and use $\alpha = 0.55$. In this way, we generated 50 problem instances for each n.

We translate each generated problem instance represented by type-based SCG into an equivalent problem instance represented by standard SCG. In Figure 1(a), the x-axis shows the number of elements in the type-based SCG representation, and the y-axis shows the number of elements in the standard SCG representation. Each data point shows the average of 50 problem instances. The number of elements in the standard SCG grows exponentially compared to that in the type-based SCG When the number of elements in the type-based SCG representation exceeds 40, we cannot translate the problem instances due to insufficient memory. This result illustrates that the type-based representation is exponentially more concise than the standard representation.

We investigate the computation time of our DP-based algorithm. For comparison, we show the results of the MIP formulation in Ohta *et al.* [6], which uses a standard SCG representation. To obtain this result, we used CPLEX version 12.1, a general-purpose mixed integer programming package.

Figure 1(b) illustrates the average computation times for solving the generated problem instances by our DP-based algorithm using the type-based SCG (DP) and by CPLEX in the MIP formulation using the standard SCG (CPLEX). The x-axis indicates the number of agents, and the y-axis shows the average computation times. When $n \leq 20$, CPLEX is faster than DP, while DP eventually outperforms CPLEX for $n > 20$. CPLEX can reduce the search space efficiently when the input size is relatively small. However, the input size for CPLEX grows exponentially. Thus, its computation time increases very rapidly. When $n > 40$, even generating problem instances becomes infeasible. On the other hand, the computation time for DP grows more slowly than the exponential rate. This result corresponds to the theoretical complexity presented in Theorem 11, i.e., finding an optimal coalition structure can be done in $O(n^{2t})$ time. As shown in

this result, the type-based SCG enables us to solve a CSG problem instance with up to 100 agents in a reasonable amount of time.

7 Conclusion

In this paper, we developed a new concise representation scheme for a characteristic function, which is based on the idea of *agent types*. The type-based representation can be exponentially more concise than existing concise representation schemes. Furthermore, this idea can be used in conjunction with existing schemes, i.e., MC-nets and SCG, for further reducing the representation size. We showed that most problems in coalitional games, including CSG, can be solved in polynomial time in the number of agents, assuming the number of types t is fixed. We also experimentally showed that a type-based SCG enables us to solve a CSG problem instance with up to 100 agents in a reasonable amount of time. Our idea of using agent types is inspired by the recent work of Shrot *et al.* [9]. However, in contrast to their study, our work introduced the idea of describing a characteristic function explicitly using agent types in the first place, and considered a wider range of problems in coalitional games including CSG.

Our future works include examining the complexity of solving other problems in coalitional games, e.g., finding the nucleolus, and combining the idea of agent types with other concise representation schemes such as [11].

References

1. Bachrach, Y., Rosenschein, J.S.: Coalitional skill games. In: AAMAS. pp. 1023–1030 (2008)
2. Chalkiadakis, G., Elkind, E., Jennings, N.R.: Simple coalitional games with beliefs. In: IJCAI. pp. 85–90 (2009)
3. Conitzer, V., Sandholm, T.: Complexity of constructing solutions in the core based on synergies among coalitions. Artificial Intelligence 170(6), 607–619 (2006)
4. Ieong, S., Shoham, Y.: Marginal contribution nets: a compact representation scheme for coalitional games. In: EC. pp. 193–202 (2005)
5. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer (2004)
6. Ohta, N., Conitzer, V., Ichimura, R., Sakurai, Y., Iwasaki, A., Yokoo, M.: Coalition structure generation utilizing compact characteristic function representations. In: CP. pp. 623–638 (2009)
7. Rahwan, T., Jennings, N.R.: An improved dynamic programming algorithm for coalition structure generation. In: AAMAS. pp. 1417–1420 (2008)
8. Sandholm, T., Larson, K., Andersson, M., Shehory, O., Tohmé, F.: Coalition structure generation with worst case guarantees. Artificial Intelligence 111(1-2), 209–238 (1999)
9. Shrot, T., Aumann, Y., Kraus, S.: On agent types in coalition formation problems. In: AAMAS. pp. 757–764 (2010)
10. Tennenholtz, M.: Some tractable combinatorial auctions. In: AAAI. pp. 98–103 (2000)
11. Ueda, S., Iwasaki, A., Yokoo, M., Silaghi, M.C., Hirayama, K., Matsui, T.: Coalition structure generation based on distributed constraint optimization. In: AAAI. pp. 197–203 (2010)

False-Name-Proofness in Social Networks

Vincent Conitzer[1], Nicole Immorlica[2], Joshua Letchford[1], Kamesh Munagala[1], and Liad Wagman[3]

[1] Duke University
{conitzer,jcl,kamesh}@cs.duke.edu
[2] Northwestern University
nickle@eecs.northwestern.edu
[3] Illinios Institute of Technology
lwagman@stuart.iit.edu

Abstract. In mechanism design, the goal is to create rules for making a decision based on the preferences of multiple parties (agents), while taking into account that agents may behave strategically. An emerging phenomenon is to run such mechanisms on a social network; for example, Facebook recently allowed its users to vote on its future terms of use. One significant complication for such mechanisms is that it may be possible for a user to participate multiple times by creating multiple identities. Prior work has investigated the design of *false-name-proof* mechanisms, which guarantee that there is no incentive to use additional identifiers. Arguably, this work has produced mostly negative results. In this paper, we show that it is in fact possible to create good mechanisms that are robust to false-name-manipulation, by taking the social network structure into account. The basic idea is to exclude agents that are separated from trusted nodes by small vertex cuts. We provide key results on the correctness, optimality, and computational tractability of this approach.

1 Introduction

Recently, Facebook, Inc. decided to allow its users to vote on its future terms of use [19]. While the result was not binding,[4] this vote represents a new phenomenon that is likely to become more prominent in the future: agents participating in an election or other mechanism through a social networking site. Holding an election among the users of a social networking site introduces some issues that do not appear in regular elections. Perhaps the foremost such issue, and the one that we will focus on, is that it is generally easy for a user to create additional accounts/identities, allowing her to vote multiple times. This can compromise the legitimacy of the election and result in a suboptimal alternative being chosen.

The topic of designing elections or other mechanisms for settings where it is easy to create multiple identities and participate multiple times has already received some attention. The primary approach has been to design mechanisms that are *false-name-proof* [15, 16], meaning that an agent never benefits from participating more than once. (This is analogous to the better-known concept of *strategy-proofness*, meaning that an agent never benefits from misreporting her preferences. In fact, false-name-proofness is often defined in a way that subsumes strategy-proofness.) Unfortunately, existing results on false-name-proofness are quite negative, especially in voting contexts. For the

[4] The result would have been binding if at least 30% of all active users had voted, a seemingly impossibly high turnout in this context.

case where additional identities can be created at zero cost, a general characterization of false-name-proof voting mechanisms has been given [5]; this characterization implies that for the special case where there are only two alternatives, the best we can do is the *unanimity* mechanism. This mechanism works as follows: if all voters agree on which alternative is better, that alternative is chosen; but if there is any disagreement (no matter in which proportions), then a fair coin is flipped to decide between the alternatives. This is an extremely negative result, since the mechanism is almost completely unresponsive to the votes.[5] Several ways to circumvent such negative results have been proposed, such as assuming that creating additional identities comes at a small cost [14] or considering a model in which it is possible to verify some of the identities [4].

These prior results do not consider any social network structure that may hold among the identities. Rather, these earlier results can be thought of as applying to settings where a user creates an account for the sole purpose of casting a vote (or bid, etc.), so that no social network structure is specified. We will show in this paper that by using the social network structure in the mechanism, it is possible to obtain much more positive results, because fake identities will look suspect in the social network (graph) structure. To give some intuition, consider John Doe, who has a legitimate account on the social networking site. In order to cast more votes, he can create several other identities (*false names*), such as Jane Jones and Jimmy Smith. Among the accounts that he controls, he can create any network structure by linking them to each other. However, if the other users behave legitimately, then he will not be able to link his additional accounts to any of the other users' identities (since, after all, they have never heard of Jane Jones or Jimmy Smith); he will only be able to get his friends to link to his legitimate identity (John Doe). This results in an odd-looking social network structure, where his legitimate identity constitutes a vertex cut in the graph, whose removal separates the fake identities from the rest of the graph.

In the remainder of this paper, we generalize the intuition afforded in the above scenario, giving a notion of when a node is "suspect" based on small vertex cuts that separate it from the trusted nodes. In Section 2, we formally define the setting that we will focus on. In Section 3, we discuss false-name-proofness and provide a sufficient condition for guaranteeing it. In Section 4, we discuss how to find all suspect nodes when trusted nodes are given exogenously to the algorithm. Then, in Section 5, we extend our analysis to settings in which we do not have trusted nodes initially, but we can actively verify nodes. We give both correctness and optimality results. Appendix A contains simulation results for random graph models, in which we investigate how many vertices will typically be regarded as suspect (exogenous case) or how many need to be verified (endogenous case). The full version includes all the proofs and some additional examples.

[5] The literature on false-name-proof voting mechanisms is quite recent: earlier work on false-name proofness considered other settings, such as *combinatorial auction* mechanisms, where multiple items are for sale at the same time. Unfortunately, here, too, there are strong impossibility results, including a result that states that under certain conditions, from the perspective of a worst-case efficiency ratio, it is impossible to significantly outperform the simple mechanism that sells all items as a single bundle [8].

Related Work. The basic intuition that the creation of false identities in a social network results in suspiciously small vertex cuts has previously been explored in several papers, in peer-to-peer networks [18, 17] and web spam detection [2, 3, 6, 7, 13].

The work on fraud in peer-to-peer networks attempts to thwart Sybil attacks in which one or more malicious users obtain multiple identities in order to out-vote legitimate users in collaborative tasks like Byzantine failure defenses. These papers propose protocols that ensure that *not too many* false identities are accepted. While this may be sufficient to thwart certain Sybil attacks in decentralized distributed systems, it can still leave incentives for an agent to create multiple identities, especially in applications such as elections in which the electorate is about evenly divided. Furthermore, a major hurdle in the Sybil attack research is that any protocol must be decentralized. In contrast, in this paper, we follow the stricter approach of guaranteeing that the creation of false identities is always weakly suboptimal, corresponding to the standard approach in the mechanism design literature. On the other hand, we allow our mechanisms to be centralized, as we envision them being run by the proprietor of the social network who has access to the network structure.

Fraud is also prevalent in the world wide web where users sometimes create fake webpages and links with the sole intent of boosting the PageRank of given website(s). Several researchers have considered using link structure to combat spam [2, 3, 6, 7, 13]. In SpamRank [2, 3], the authors assume that a node is suspect if the main contribution to its PageRank is generated from a small set of supporting nodes (see also [6]). Our focus on small vertex cuts can be interpreted as an extreme version of the conditions proposed in SpamRank. An alternative approach, as taken by TrustRank [7] and Anti-TrustRank [13], assumes the existence of an oracle (e.g., a human being) which is able to determine the legitimacy of any given website. Calls to the oracle are, however, expensive, and so the main task in the protocol is to select a seed set of pages. The protocol then guesses the legitimacy of the remaining pages based on their connectivity to the seed set. In particular, the protocol assumes that legitimate pages rarely point to illegitimate ones, and hence the illegitimate pages are those that are "approximately isolated." Again, this approach is similar to our approach at a high level; the selection of the seed set corresponds to our verification policy (discussed later in the paper), and the condition of approximate isolation corresponds to the condition of small vertex cuts in our work. Despite these similarities, the particulars of the model and definitions are quite different, as these protocols are designed to combat fraudulent attacks in PageRank, whereas our goal is to prevent fraudulent attacks in voting or other mechanisms.

2 Setting

Our results can be applied to any mechanism design domain, but for the sake of concreteness, it may be helpful to think about the simple setting in which m agents must select between two alternatives. Each agent has a strict preference for one alternative over the other. The mechanism designer wishes to make a socially desirable choice, i.e., select an alternative that is beneficial for society as a whole. The majority rule, in which the alternative preferred by more voters wins, would be ideal; unfortunately, the majority rule will result in incentives to create false names, if naïvely applied.

Agents are arranged in a social network consisting of n nodes where $m \leq n$. Each agent i has a legitimate account in the social network, corresponding to a node v_i^t, as well as a (possibly empty) set of illegitimate accounts V_i^f. There is an arbitrary graph structure among the legitimate nodes in the social network—that is, we impose no structure on the subgraph induced by the legitimate nodes $\{v_i^t\}_{i \in \{1,\ldots,m\}}$.

In the most basic version of our model, we assume that no two manipulating agents can work together, so that an agent can only link her illegitmate nodes to each other and to her own legitimate node. Hence, for any $i \neq j$, there are no edges between V_i^f and $\{v_j^t\} \cup V_j^f$. However, for each agent i, we allow an arbitrary graph structure on $\{v_i^t\} \cup V_i^f$.

In the more general version of our model, we assume that up to k agents can collude together. (The basic model is the special case where $k = 1$.) That is, the agents $1, \ldots, m$ are partitioned into coalitions $S_j \subseteq \{1, \ldots, m\}$, with $|S_j| \leq k$ for each j. Let $V_{S_j}^f$ be the set of all illegitimate nodes used by S_j, that is, $V_{S_j}^f = \bigcup_{i \in S_j} V_i^f$, and let $V_{S_j}^t$ be the set of all legitimate nodes used by S_j, that is, $V_{S_j}^t = \bigcup_{i \in S_j} \{v_i^t\}$. Two distinct coalitions cannot link their illegitimate nodes to each other, so that for any $i \neq j$, there are no edges between $V_{S_i}^f$ and $V_{S_j}^t \cup V_{S_j}^f$. However, for each coalition S_i, we allow an arbitrary graph structure on $V_{S_i}^t \cup V_{S_i}^f$.

To summarize, our social network setting consists of

- a set of m agents denoted $\{1, \ldots, m\}$,
- a set of m legitimate nodes, one for each agent, denoted $V^t = \{v_1^t, \ldots, v_m^t\}$,
- a collection of m (possibly empty) sets of illegitimate nodes, one for each agent, denoted $\{V_1^f, \ldots, V_m^f\}$,
- a partition of the agents $\{1, \ldots, m\}$ into subsets S_j, where $|S_j| \leq k$ (the no-collusion case corresponds to $k = 1$), such that for any i, j, there are no edges between $V_{S_i}^f$ and $V_{S_j}^t \cup V_{S_j}^f$ (apart from this, the graph structure can be arbitrary).

Some of the nodes in the graph will be *trusted*. For example, the mechanism designer may personally know the agents corresponding to these nodes in the real world. This is a case in which trust is *exogenous*, that is, we have no control over which agents are trusted: the trusted agents are given as part of the input. Later in the paper, we will consider settings where we can, with some effort, *verify* whether any particular node is legitimate (for example, by asking the node for information that confirms that there is a corresponding agent in the real world). Nodes that pass this verification step become trusted nodes; this is a case of *endogenous* trust. It should be noted that, in either case, we do *not* assume that a trusted node will refrain from creating additional identifiers. That is, the only sense in which the node is trusted is that we know it corresponds to a real agent.

The mechanisms that we consider in this paper operate as follows. A *suspicion policy* is a function that takes as input the social network graph $G = (V, E)$ as well as a set T of trusted nodes, $T \subseteq V^t \subseteq V$; and as output labels every node in V as either "deemed legitimate" or "suspect." Generally, all the nodes in T will be deemed legitimate, but others may be deemed legitimate as well based on the network structure. Subsequently, all the nodes that have been deemed legitimate get to participate (*e.g.*,

vote) in a standard mechanism f (*e.g.*, the majority rule), and based on this an outcome is chosen. (In this context, we only consider *anonymous* mechanisms that treat all nodes that get to participate identically.) In the case where nodes become trusted through verification, we also have a *verification policy* that takes G as input and determines which nodes to verify.

We consider a game played between the mechanism designer and the agents (more precisely, the coalitions S_j). First, the mechanism designer announces her mechanism, consisting of f and the suspicion policy (and, in the case where trust is obtained through verification, a verification policy). Then, each coalition S_j creates its illegitimate nodes $V^f_{S_j}$, as well as the edges that include these nodes (they can only have edges to other nodes in $V^f_{S_j}$, and to $V^t_{S_j}$). Note that the coalitions do *not* strategically determine edges between legitimate nodes in this game: in order to focus on false-name manipulation, only the creation of false nodes and their edges is modeled in the game. Also note that the mechanism designer, when announcing her mechanism, is unaware of the true graph as well as which agents are in coalitions together.

After obtaining the social network graph (and, possibly, some exogenously trusted nodes), the mechanism designer runs (1) (possibly) the verification policy and (2) the suspicion policy. The designer subsequently asks the nodes that have been deemed legitimate to report their preferences, and then finally runs (3) the standard mechanism f on these reported preferences, to obtain the outcome.

Whether this results in incentives for using false names depends on all of the components (1), (2), and (3), and each one individually can be used to make the whole mechanism false-name-proof. For example (for component 3), if f is by itself false-name-proof, then even if we verify no nodes and deem every node legitimate, there is still no incentive to engage in false-name manipulation. The downside of this approach is that we run into all the impossibility results from the literature on designing false-name-proof mechanisms. Similarly (for component 1), if we verify all nodes and then only deem the trusted nodes (the ones that passed the verification step) legitimate, there is no incentive to use false names. Of course, this generally results in far too much overhead. In this paper, we will be interested in suspicion policies (component 2) that by themselves guarantee that there is no incentive to use false names. For this, we heavily rely on the social network structure. In the first part of the paper, we do not consider verification policies—we take which nodes are trusted as given exogenously.

3 False-name-proofness

To define what it means for a suspicion policy to guarantee false-name-proofness, we first need to define some other properties. The next two definitions assume that a coalition can be thought of as a single player with coherent preferences; this is reasonable in the sense that if there is internal disagreement within the coalition, this will only make it more difficult for them to manipulate the mechanism.

Definition 1. *A standard mechanism f is* k-strategy-proof *if it is a dominant strategy for every coalition of size at most k to report truthfully.*

Definition 2. *A standard (anonymous) mechanism f* satisfies k-voluntary participation *if it never helps a coalition of size at most k to use fewer identifiers.*

Because the coalitions play a game with multiple stages, it is important to specify what we assume the coalitions learn about each other's actions in earlier stages—that is, what are the information sets in the extensive form of the game? Specifically, when a coalition reports its preferences to f, what does the coalition know about the nodes and edges created by other coalitions? We assume that a coalition learns nothing about other coalitions' actions, except that the coalition can (possibly) make inferences about what others have done based on which of its own nodes have been deemed legitimate. Thus, it is assumed that each coalition is rational and has perfect recall, but also that it does not have any other way of observing what other coalitions have done.

Definition 3. *We say that the* Limited Information Assumption (LIA) *holds if, for every coalition S_j, for every two nodes*[6] *ν_1, ν_2 in the extensive form of the game (where S_j is about to report preferences to f), the following holds. If S_j has taken the same node-and-edge creation actions at ν_1 and ν_2, and the same nodes have been deemed legitimate for S_j at ν_1 and ν_2, then these nodes are in the same information set—that is, S_j cannot distinguish them.*

It should be emphasized that LIA does not specify the information sets exactly—it is merely an *upper bound* on how much the coalitions learn about each other's actions. Specifically, we can also require the coalitions to report preferences for nodes *before* informing them exactly which of these nodes have been deemed legitimate. In an extreme special case of this (for which our results still hold), we can consider the situation where a coalition must create nodes and edges and report preferences for its nodes *at the same time*, making the game a single-stage game. In this case, when a coalition is reporting preferences, it clearly knows nothing about what the other coalitions have done at all, since they are moving at the same time. This is equivalent to saying that a coalition first creates nodes and edges, and then reports preferences for these nodes but without learning anything (including which of these nodes have been deemed legitimate). This is consistent with LIA: it just means that even more nodes in the game tree are in the same information set than is strictly required by LIA.

We now define what it means for a suspicion policy to guarantee false-name-proofness.

Definition 4. *A suspicion policy Π* guarantees false-name-proofness for coalitions of size at most k *if, under the LIA assumption, the following holds. For any standard (anonymous) mechanism f that is k-strategy-proof and satisfies k-voluntary participation, if we combine Π with f, then for any true social network structure on V^t, for any initial trusted nodes $T \subseteq V^t$, and for any partition of V^t into coalitions S_j of size at most k each, it is a dominant strategy for each coalition to set $V^f_{S_j} = \emptyset$ and report truthfully.*

A Sufficient Condition for Guaranteeing False-Name-Proofness. We now provide a sufficient condition for guaranteeing false-name-proofness.

Definition 5. *A suspicion policy Π is* k-robust *if, for any true social network structure on V^t, for any initial trusted nodes $T \subseteq V^t$, and for any partition of V^t into coalitions S_j of size at most k each, we have the following. For every coalition S_j, for every profile of actions taken by the other coalitions:*

[6] These are not to be confused with the nodes in the network.

1. *The actions of S_j (in terms of creating new nodes and edges) do not affect which of the other coalitions' identifiers ($V \setminus (V^t_{S_j} \cup V^f_{S_j})$) are deemed legitimate.*
2. *The number of identifiers in $V^t_{S_j} \cup V^f_{S_j}$ that are deemed legitimate is maximized by setting $V^f_{S_j} = \emptyset$.*

Theorem 1. *If a suspicion policy Π is k-robust, then it guarantees false-name-proofness for coalitions of size at most k.*

4 Exogenously Given Trusted Nodes

We begin by studying the case where the trusted nodes T are given exogenously. This could correspond to the case where the mechanism designer personally knows the owners of some of the nodes on the network, or perhaps these nodes have already been successfully verified in an earlier stage. Later in the paper, we will study the case where there are no exogenously given trusted nodes, so that we have to decide which nodes to verify. Given G and T, the next step is to determine which nodes to label as "suspect," based on the fact that they are not well connected to trusted nodes. We will make our suspicion policy precise shortly, but first we illustrate the basic idea on a small example. We recall that k denotes the maximum size of a coalition of colluding agents. Figure 1 gives an example of a network with two exogenously given trusted nodes, for the case where $k = 1$. As the figure illustrates, nodes that are separated from the trusted nodes

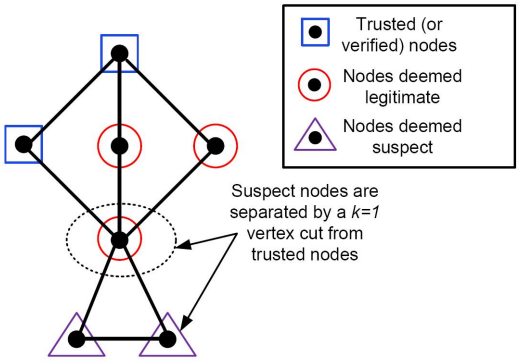


Fig. 1. Example network. The nodes correspond to identities (user accounts), and the edges correspond to (say) friendship relations between the identities. The mechanism designer, at this point for exogenous reasons, considers certain nodes "trusted" (marked by squares), that is, she is sure that they are not false names. The nodes marked with triangles are separated from the trusted nodes by a vertex cut of size one (indicated by the dotted ellipse). As a result, it is conceivable that these nodes are false names, created by the agent corresponding to the vertex-cut node; hence, they are labeled *suspect*. The remaining nodes are not separated from the trusted nodes by a vertex cut of size one, and as a result they are deemed *legitimate* (marked by circles).

by a vertex cut of size 1 could be false identities created by the node on the vertex cut in order to manipulate the outcome of the mechanism. Hence, they are deemed *suspect*.

In the following subsections, we first define our suspicion policy precisely and prove that it has several nice properties, including guaranteeing false-name-proofness. We then prove that this policy is optimal in the sense that any other suspicion policy with these properties would label more nodes as suspect. Finally, we give a polynomial-time algorithm for determining whether nodes are deemed legitimate or not under this policy.

The Suspicion Policy. One natural approach is to label as suspect every node v that is separated from all the trusted nodes by a vertex cut of size at most k (this cut may include some of the trusted nodes). After all, such a node v may have been artificially created by a coalition of nodes corresponding to its vertex cut. On the other hand, for a node v that is *not* separated from the trusted nodes by any vertex cut of size at most k, there is no coalition of nodes that could have artificially created v. While this reasoning is correct, it turns out that, to guarantee false-name-proofness, it is not sufficient to label *only* the nodes separated from the trusted nodes by a vertex cut of size at most k as suspect. The reason is that this approach may still leave an incentive for a coalition to create false nodes: not because these false nodes will be deemed legitimate, but rather because it may prevent *other* nodes from being labeled as suspect. We first observe a fundamental property of nodes being separated from the trusted nodes by a vertex cut of size at most k.

Lemma 1 (cf. Menger [11]). *For an initially untrusted node v, the following two statements are equivalent.*

1. *v is not separated from the initially trusted nodes by a vertex cut of size at most k (which may include initially trusted nodes).*
2. *There exist $k+1$ vertex-disjoint paths from (distinct) initially trusted nodes to v.*

The problem with the approach above is that a coalition may use false nodes that will be labeled suspect, but that help create paths to other nodes that will be deemed legitimate as a result. The solution is to apply the procedure *iteratively*, in each stage removing the nodes that are separated from all the trusted nodes by a vertex cut of size at most k, until convergence.

Definition 6. *Let r take as input $G = (V, E)$ and $T \subseteq V$, and as output produce the subgraph G' of G that results from removing those nodes in $V - T$ that are separated from the trusted nodes T by a vertex cut of size at most k (as well as removing the edges associated with these nodes). These vertex cuts are allowed to include nodes in T. Let $G = G^{(0)}, G' = G^{(1)}, G^{(2)}, \ldots, G^{(n_{G,T})}$ be the sequence of graphs that results from applying r iteratively on $(G^{(i)}, T)$, where $n_{G,T}$ is the smallest number satisfying $G^{(n_{G,T})} = G^{(n_{G,T}-1)}$ (note this sequence must converge as the set of nodes in successive iterations is nonincreasing). Then our suspicion policy Π_k^*, when applied to (G, T), deems all the nodes in $G^{(n_{G,T})}$ legitimate, and all the other nodes in G suspect.*

In each iteration, the procedure for computing Π_k^* removes *all* the nodes that are at that point separated from all the trusted nodes by a vertex cut of size at most k. This

corresponds to eliminating nodes in a particular order. One may wonder if the result would be any different if we eliminated nodes in a different order, for example, in one iteration removing only a subset of the nodes that are at that point separated from all the trusted nodes by a vertex cut of size at most k, before continuing to the next iteration. This is analogous to the notion of path independence of iterated strict dominance in game theory: no matter in which order we eliminate strictly dominated strategies, in the end we obtain the same set of remaining strategies [10]. (This is in contrast to iterated *weak* dominance, where the order of elimination does affect the final remaining strategies.) We will show a similar path independence result for removing nodes in our setting. To do so, we first define the class of suspicion policies that correspond to *some* order; then we show that the class has only one element, namely, Π_k^*.[7]

Definition 7. *Let $\mathbf{\Pi}_k$ be the class of all suspicion policies that correspond to a procedure where:*

- *In each iteration, some subset of the nodes that are at that point separated from all the trusted nodes by a vertex cut of size at most k is eliminated from the graph;*
- *This subset must be nonempty when possible;*
- *When no additional nodes can be eliminated, the remaining nodes are exactly the ones deemed legitimate.*

Lemma 2. *The class $\mathbf{\Pi}_k$ consists of a singleton element Π_k^*,* i.e., *$\mathbf{\Pi}_k = \{\Pi_k^*\}$.*

We now show that our policy Π_k^* guarantees false-name-proofness for coalitions of size at most k.

Lemma 3. *Let $G = (V, E)$ be a graph and let $T \subseteq V$ be the trusted nodes. Let G' be a graph that is obtained from G by adding additional nodes V' and additional edges E' that each have at least one endpoint in V'—in such a way that every node in V' is separated from T by a vertex cut of size at most k. Then, applying Π_k^* to $G' = (V \cup V', E \cup E')$ and T results in the same nodes being deemed legitimate as applying Π_k^* to G and T.*

Theorem 2. *Π_k^* is k-robust (and hence, by Theorem 1, guarantees false-name-proofness for coalitions of size at most k). Moreover, under Π_k^*, a coalition S_j's actions also do not affect which of its own legitimate nodes $V_{S_j}^t$ are deemed legitimate. Finally, Π_k^* is guaranteed to label every illegitimate node as suspect.*

Optimality. We now show that Π_k^* is the best possible suspicion policy in the sense that any other policy satisfying the desirable properties in Theorem 2 must label more nodes as suspect.

Theorem 3. *Let Π' be a suspicion policy that (1) is k-robust, (2) is such that a coalition S_j's actions also do not affect which of its own legitimate nodes $V_{S_j}^t$ are deemed legitimate, and (3) is guaranteed to label every illegitimate node as suspect. Then, if Π_k^* labels a node as suspect, then so must Π'.*

[7] The different orders of course correspond to different *procedures* for computing which nodes are deemed legitimate, but we will show that as a *function* that determines which nodes are finally deemed legitimate, they are all the same.

Polynomial-time Algorithm for Determining Whether a Node is Suspect. In this subsection, we give a polynomial-time algorithm for determining whether nodes are deemed legitimate or suspect according to Π_k^*. The key step is to find an algorithm for figuring out which nodes are separated from the trusted nodes by a vertex cut of size at most k; then we can simply iterate this in order to execute Π_k^* (and by Lemma 2 we do not need to be careful about the order in which we eliminate nodes). It turns out that by Lemma 1, we can do this by solving a sequence of maximum flow problem instances.

Theorem 4. *Given $G = (V, E)$ and $T \subseteq V$, we can determine in polynomial time which nodes are not separated from T by a vertex cut of size at most k. As the number of iterations of Π_k^* is bounded by $|V|$, we can run Π_k^* in polynomial time.*

5 Choosing Nodes to Verify (Endogenous Trust)

Our methodology requires some nodes to be trusted. So far, we have considered settings where some nodes are trusted for exogenous reasons (for example, the organizer's own friends may be the only trusted nodes). However, we can also endogenize which nodes are trusted, by assuming that the organizer can invest some effort in *verifying* some of the identities to establish their legitimacy (for example, by asking these identities for information that identifies them in the real world). This is an approach that has been considered before in the context of false-name-proofness [4], but that prior work paid no regard to social network structure. The social network structure can drastically reduce the amount of verification required, because, as we have seen earlier in this paper, once we have some nodes that are trusted, we can infer that others are legitimate.

There are (at least) two approaches to consider here: verify enough nodes so that no suspect nodes remain at all (and try to minimize the number of verified nodes under this constraint), or try to maximize the number of nodes deemed legitimate, given a budget of verifications (say, at most b verifications). In this paper, we focus on the former.

Technically, a verification policy consists of a contingency plan, where the next node to verify depends on the results of earlier verifications of nodes (which can either fail or succeed). If a node fails the verification, that node is classified as illegitimate, and the verification continues. The verification continues until no nodes remain suspect (other than ones that failed the verification step)—that is, until no unverified nodes are separated by a vertex cut of size at most k from the nodes that were successfully verified. (This vertex cut can include successfully verified nodes. We note that in this context there is no longer a reason to *iteratively* remove nodes in the procedure that computes the trust policy (Π_k^*): because our goal is for *all* remaining nodes to be deemed legitimate, we simply need to check whether any nodes are removed in the first iteration.)

Optimally Deciding Which Nodes to Verify. We now turn to the following optimization problem: how do we minimize the number of nodes that we verify before reaching the point where all the remaining nodes are deemed legitimate? To answer this question, we first note that, since there will be no incentive to create illegitimate nodes, we can assume that all nodes will in fact be legitimate. (This does not mean that we can afford to not do the verification, because if we did not, then there would be incentives to create illegitimate nodes again.) Hence, the problem becomes to find a minimum-size subset of nodes so that no other node is separated from these nodes by a vertex cut of size at most k (which may include nodes in this subset)—or, equivalently, by Lemma 1,

to find a minimum-size subset of nodes so that every other node is connected by $k+1$ vertex-disjoint paths to (distinct nodes in) this subset.

This problem is a special case of the *source location* problem. A polynomial-time algorithm for this problem is given in a paper by Nagamochi et al. [12]. They show that the problem has a matroidal property, as follows. Instead of thinking about minimizing the number of verified nodes, we can think about maximizing the number of unverified nodes. Say a subset $U \subseteq V$ is *feasible* if, for every $v \in U$, there exist $k+1$ vertex-disjoint (apart from v) paths to (distinct) nodes in $V \setminus U$.

Theorem 5 ([12]). *The feasible sets satisfy the independence axioms of a matroid.*

Finding an independent set of maximum size in a matroid is easy: start with an empty set, and attempt to include the elements one at a time, being careful not to violate the independence property. In the context of trying to find a minimum-size set of nodes to verify, this corresponds to starting with the set of *all* nodes, and attempting to *ex*clude the nodes one at a time, being careful that it will still result in all the excluded nodes being deemed legitimate. To check the latter, we only need to consider the current node:

Lemma 4. *Suppose $S \subseteq V$ is such that from every $u \in V - S$, there exist $k+1$ vertex-disjoint paths to (distinct nodes in) S, and suppose that for some v, $S - \{v\}$ does not have this property. Then, there do not exist $k+1$ vertex-disjoint paths from v to (distinct nodes in) $S - \{v\}$.*

This results in the following simple polynomial-time algorithm Φ_k for finding a minimum-size set of nodes to verify.

Definition 8. *Φ_k takes as input a graph $G = (V, E)$ and proceeds as follows to determine the nodes S to verify:*

1. *Initialize $S \leftarrow V$.*
2. *For each node $v \in S$: if there are $k+1$ vertex-disjoint paths from $S - \{v\}$ to v, then remove v from S.*
3. *Return S.*

6 Conclusions and Future Research

From the above, it becomes clear that false-name-proofness, while achievable in social networking settings, does not come for free: we either cannot let all agents participate, or we must spend significant effort verifying identities. How severe these downsides are depends on the exact structure of the social network. If we have a sufficiently densely connected social network, then almost everyone can participate even when there are relatively few trusted identities, or, alternatively, we only need to verify a small number of identities to let everyone participate. But, is this likely to be the case in realistic social networks? In the appendix, we provide some results based on experimental simulation; future research may be devoted to obtaining analytical characterizations.

Future research may also be devoted to considering some changes in the basic model and their effect on our results. What happens if agents can decide to drop edges (that is, not declare friendships) for strategic reasons? What happens if agents can get other agents to link to their fake identities at a cost? Results here may be reminiscent of those

obtained in existing models where additional identifiers can be obtained at a cost [14]. What happens when we can only verify a limited number of nodes and try to maximize the number of nodes deemed legitimate?

Acknowledgements. Conitzer and Letchford were supported by NSF under award numbers IIS-0812113 and CAREER 0953756, and by an Alfred P. Sloan Research Fellowship. Munagala was supported by an Alfred P. Sloan Research Fellowship, and by NSF via CAREER award CCF-0745761 and grants CCF-1008065 and CNS-0540347. Wagman was supported by an IIT Stuart School of Business Research Grant.

References

1. Réka Albert and Albert–László Barabási. Statistical mechanics of complex networks, 2001.
2. R. Andersen, C. Borgs, J. Chayes, J. Hopcroft, V. Mirrokni, and S. Teng. Local computation of PageRank contributions. In *WAW*, 2007.
3. Reid Andersen, Christian Borgs, Jennifer T. Chayes, John E. Hopcroft, Kamal Jain, Vahab S. Mirrokni, and Shang-Hua Teng. Robust PageRank and locally computable spam detection features. In *AIRWeb*, pages 69–76, 2008.
4. Vincent Conitzer. Limited verification of identities to induce false-name-proofness. In *TARK*, pages 102–111, Brussels, Belgium, 2007.
5. Vincent Conitzer. Anonymity-proof voting rules. In *Proceedings of WINE*, pages 295–306, Shanghai, China, 2008.
6. Zoltán Gyöngyi, Pavel Berkhin, Hector Garcia-Molina, and Jan Pedersen. Link spam detection based on mass estimation. In *Proceedings of VLDB*. ACM, 2006.
7. Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with TrustRank. In *Proceedings of VLDB*, pages 576–587. Morgan Kaufmann, 2004.
8. Atsushi Iwasaki, Vincent Conitzer, Yoshifusa Omori, Yuko Sakurai, Taiki Todo, Mingyu Guo, and Makoto Yokoo. Worst-case efficiency ratio in false-name-proof combinatorial auction mechanisms. In *Proceedings of AAMAS*, pages 633–640 Toronto, Canada, 2010.
9. Jon M. Kleinberg. Navigation in a small world. *Nature*, 2000.
10. Duncan R. Luce and Howard Raiffa. Games and decisions: introduction and critical survey, New York, 1957
11. Karl Menger. Zur allgemeinen Kurventheorie. *Fund. Math.*, 10:96–115, 1927.
12. Hiroshi Nagamochi, Toshimasa Ishii, and Hiro Ito. Minimum cost source location problem with vertex-connectivity requirements in digraphs. *Information Processing Letters*, 80(6):287–293, 2001.
13. R. Raj and V. Krishnan. Web spam detection with anti-trust rank. In *AIRWeb*, pages 381–389, 2006.
14. Liad Wagman and Vincent Conitzer. Optimal false-name-proof voting rules with costly voting. In *Proceedings of AAAI*, pages 190–195, Chicago, IL, USA, 2008.
15. Makoto Yokoo, Yuko Sakurai, and Shigeo Matsubara. Robust combinatorial auction protocol against false-name bids. *Artificial Intelligence*, 130(2):167–181, 2001.
16. Makoto Yokoo, Yuko Sakurai, and Shigeo Matsubara. The effect of false-name bids in combinatorial auctions: New fraud in Internet auctions. *GEB*, 46(1):174–188, 2004.
17. Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, and Feng Xiao. SybilLimit: A near-optimal social network defense against sybil attacks. *ToN*, 18(3):885-898, 2010.
18. Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. SybilGuard: Defending against sybil attacks via social networks. *ToN*, 16(3):576–589, 2008.
19. Mark Zuckerberg. Voting begins on governing the Facebook site, 2009. http://blog.facebook.com/blog.php?post=76815337130.

A Experimental results

To evaluate how many nodes will be deemed legitimate, or how many nodes we need to verify, in "typical" social networks, we conducted a series of experiments on randomly generated graphs. To generate these graphs we used two models, Kleinberg's model for navigable small world networks [9] and the Barabási-Albert model [1] for generating scale-free networks.

We construct random social networks with Kleinberg's model as follows (labeled "Klein" in the following graphs). Take a square grid and connect each node to its neighbors. Next, for each node v, add one additional connection to a random node v' ($v' \neq v$) in the graph, with probability proportional to 1/(distance(v,v')).

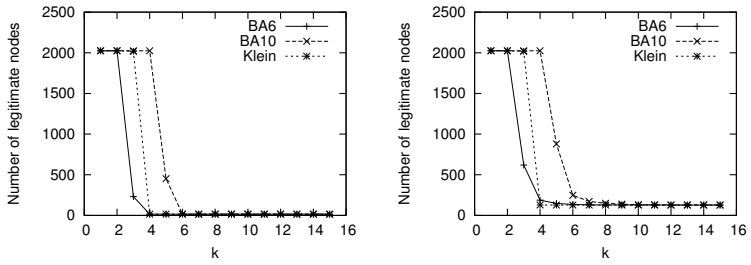
We construct random social networks with the Barabási-Albert model as follows. Start with a small seed graph (we start with a small grid of 9 nodes). We add the remaining nodes one by one. For each node v we add, we connect it to 3 (BA-6) or 5 (BA-10) of the existing nodes. To choose which nodes v will be connected to, we randomize over the existing nodes, but the probablity of picking a specific node v' is proportional to the number of neighbors that v' has (this means that nodes that already have a large number of neighbors have a higher probability of receiving more neighbors).

We used the JUNG libary (Java Universal Network/Graph Framework) and CPLEX in our implementation. Due to running time and memory constraints faced by this methodology, the largest graphs we consider are of 2025 nodes (and in fact we will only present results on graphs of 2025 nodes). In the results in the following two subsections, the numbers are averages taken over 20 graphs.

A.1 Results for the exogenous case

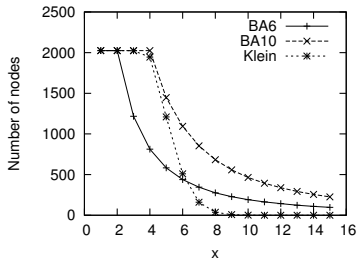
In this subsection, we present our results when some nodes in the graph are chosen at random to be trusted nodes. In our implementation, we took advantage of Lemma 2 to significantly reduce the running time of the experiment, based on the fact that if a node is eliminated when only $k+1$ vertex-disjoint paths are necessary for legitimacy, it will still be eliminated when we require $k+2$ paths. Thus, we can start the step of the experiment where $k+2$ paths are necessary by removing all of the nodes eliminated at the end of of the step where $k+1$ paths are necessary, and still get the same solution that we would have obtained by running the algorithm from the begining.

As shown in Figure 2, for small k the number of nodes deemed legitimate is very high, but as k grows there is eventually a sharp dropoff for every one of the random graph generation models (though at different points for the different models). This dropoff can be partially explained by the fact that eventually, there will be many nodes that have fewer than $k+1$ neighbors (as shown in Figure 2(c)), and these nodes will be considered suspect unless they happen to be trusted themselves. However, the number of nodes deemed legitimate does drop faster than the number of nodes with at least $k+1$ neighbors. This could be due to, for example, the iterated removal of nodes—once some nodes are removed, others may start to be removed as well.



(a) Number of nodes deemed legitimate with 16 random trusted nodes.

(b) Number of nodes deemed legitimate with 128 random trusted nodes.

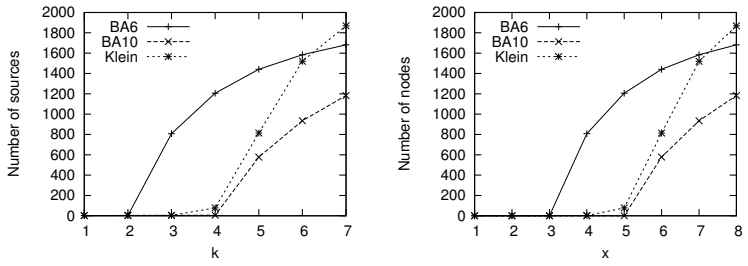


(c) Number of nodes with degree $> x$

Fig. 2. Results for exogenously trusted nodes.

A.2 Results for the endogenous case

In this subsection, we present our results for the case where no nodes are initially trusted, and we choose a subset of nodes to verify so that all nodes will be deemed legitimate. Of course, every node with degree lower than $k + 1$ must be verified. Our results in Figure 3 show that that is basically all we need to do: in this case, the number of nodes that need to be verified very closely matches the number of nodes with degree $\leq k$ for both types of social network. (Though it is difficult to see, the graphs are in fact very slightly different.)



(a) Number of nodes that must be verified for all nodes to be deemed legitimate

(b) Number of nodes with degree $\leq x$

Fig. 3. Results for verification (endogenous trust).

Path coalitional games

Haris Aziz[1] and Troels Bjerre Sørensen[2]

[1] Institut für Informatik, Technische Universität München, 80538 München, Germany
aziz@in.tum.de
[2] Computer Science Dept., University of Warwick, Coventry, UK, CV4 7AL
trold@dcs.warwick.ac.uk

Abstract. We model stable and fair resource allocations in networks via *path coalitional games*. In these games, a coalition of edges or vertices is successful if it can enable an s-t path. We present polynomial-time algorithms to compute and verify least core payoffs of *cost-based generalizations* of path coalitional games and their duals, thereby settling a number of open problems. The least core payoffs of path coalitional games are completely characterized and a polynomial-time algorithm for computing the nucleolus of edge path coalitional games on undirected series-parallel graphs is presented.

1 Introduction

We consider simple coalitional games called *path coalitional games*, in particular *Edge Path Coalitional Games (EPCGs)* and *Vertex Path Coalitional Games (VPCGs)*. In these games, the players control the edges and the vertices respectively, and a coalition of players wins if enables a path from the source s to the sink t and loses otherwise. Both of these coalitional games are natural representations, for which solution concepts such as the Shapley value or the nucleolus represent the amount of payoff the respective edges or vertices deserve for enabling a path from s to t. The payoff can indicate the importance of the players or the proportional resource, profit, maintenance or security allocation required at the respective nodes and vertices. This kind of stability analysis is especially crucial if the underlying graph represents a logistics, communication, military, supply-chain or information network [2, 11]. We study the computational complexity of computing important cooperative game theoretic solutions of path coalitional games.

Path coalitional games also have a natural correspondence with two-person zero-sum noncooperative games. In such games, which we term as *path intercept games*, there are two players, the *interceptor* and the *path-chooser*. The problem is to maximize the probability of intercepting a strategically chosen path in an undirected graph. We refer to the path intercept games as *Edge Path Noncooperative Games (EPNGs)* and *Vertex Path Noncooperative Games (VPNGs)*. The pure strategies of the interceptor are the edges E (or vertices V) and the pure strategies of the *path-chooser* is set $\mathcal{P}$ which contains all paths from vertex s to vertex t. If the edge (or vertex) used by the interceptor intersects with the

chosen path, then the interceptor wins and gets payoff 1. Otherwise it loses and gets payoff 0. Thus, the *value* of the game is the greatest probability that the interceptor can guarantee for successfully intercepting the chosen path.

The area of algorithmic cooperative game theory is beset with negative computational results (see e.g., [4, 6, 9]). In this paper, we present positive algorithmic results for *cost-based generalizations* of path coalitional games and their duals. The cost-based generalization of a simple game is a rich and widely-applicable model. For example in the case of edge path coalitional games, each edge charges a certain cost for its services being utilized. A coalition of edges gets a fixed reward for enabling an s-t path. It is then natural to examine payoffs which are fair and stable and also manage to transport goods from s to t [7]. The cost-based generalizations of path coalitional games have significance in logistics, planning and operations research. Similarly, the cost-based generalizations of duals of path coalitional games have natural importance in proposing stable reward schemes to protect strategic assets or blocking intruders in a network.

Contribution:

- We use dualization and cost-based generalization to provide a unifying framework to model s-t connectivity. The framework also covers some previously studied cooperative and non-cooperative games.
- For the cost-based generalization of path coalitional games and their duals we present the first polynomial-time algorithms to compute and verify least core payoffs. The results are some what surprising because the problem of computing the least core of the dual of vertex path coalitional game (without costs) was claimed to be NP-hard [2] and the problem of computing the least core of cost-based generalization of edge path coalitional game was conjectured to be NP-hard (page 65, [11]).
- We also present an insight that the least core payoffs can be computed for the cost-based generalization of a simple game, as long as the same can be done for the underlying simple game and the game representation is as compact as the representation of the cost function. As a corollary, we show that there exist polynomial-time algorithms to compute the least core payoffs of cost-based generalizations of *spanning connectivity games* and *weighted voting games* with bounded weights and costs.
- The least core payoffs of simple path coalitional games are characterized.
- A polynomial-time algorithm to compute the nucleolus of edge path coalitional games for undirected series-parallel graphs is presented.

2 Literature

Network interdiction is the general framework in which weakening of a network by an adversary or fortification of a network by defenders is considered [17]. Within this body of literature, shortest path interdiction (in which an adversary wants to maximize the s-t shortest path in a directed network) is related to our

setting of *s-t* path coalitional games. Whereas all the variants of shortest path interdiction problem are NP-hard [9], we present positive computational results.

We note that within the interdiction literature, whereas Vertex Path Noncooperative Game (VPNG) has not been considered in the literature (to the best of our knowledge), Edge Path Noncooperative Game (EPNG) is equivalent to the two player zero sum games considered in [19]. Washburn and Wood [19] studied maxmin strategies in EPNG. Our cooperative game formulation helps us in proposing equilibrium refinements such as the nucleolus (which corresponds to a unique refinement of the maxmin strategy of the corresponding path intercept games) and other cooperative-game solution concepts such as the Shapley value. The coalitional model, especially the cost-based generalizations helps reason about more elaborate security settings in which incentives, money and cooperation of agents is involved. The comparison between a simple coalitional game and its natural noncooperative version is similar in spirit to [1] where spanning trees are considered.

The general definition of cost-based generalization of a simple game is inspired by Fragnelli et al. [7] and Bachrach and Porat [2] where cost based versions of specific graphs-based simple games are considered. Different variants of EPCGs were considered under the umbrella of 'shortest path games' in [11] but either the complexity of core-based relaxations is not examined or the complexity of computing least core solutions was left open and in fact conjectured to be computationally hard (page 65, [11]). Similarly, Bachrach and Porat [2] claimed that the least core of dual of VPCGs is NP-hard to compute and that it can be efficiently computed for trees. We present a polynomial-time algorithm to solve a generalization of the same problem for this game as well as three other games on *any graph*. A variant of EPCGs was also considered in [12] but the focus was on strategy-proof mechanisms rather than stability issues. Deng et al. [5] considers a different type of *s-t* connectivity game which is balanced.

The *s-t* path connectivity setting also has natural links with network reliability where the goal is to compute the probability that there exists a connected path. However the network reliability literature does not consider strategic settings and certainly has no equivalent concepts such as the least core and the nucleolus etc.

3 Preliminaries

In this section, we first define the path coalitional games and path intercept games and then consider game theoretic solution.

3.1 Games

We begin with the formal definition of a *coalitional game*.

Definition 1 (Coalitional games). *A* coalitional game *is a pair* (N, v) *where* $N = \{1, \ldots, n\}$ *is a set of players and* $v : 2^N \to \mathbb{R}$ *is a* characteristic or valuation

function *that associates with each coalition* $S \subseteq N$ *a payoff* $v(S)$ *where* $v(\varnothing) = 0$.[3] *A coalitional game* (N, v) *is* monotonic *when it satisfies the property that* $v(S) \leq v(T)$ *if* $S \subseteq T$.

Throughout the paper, when we refer to a coalitional game, we assume such a coalitional game with transferable utility. For the sake of brevity, we will sometimes refer to the game (N, v) as simply v.

Definition 2 (Simple game). *A* simple game *is a monotonic coalitional game* (N, v) *with* $v : 2^N \to \{0, 1\}$ *such that* $v(\varnothing) = 0$ *and* $v(N) = 1$. *A coalition* $S \subseteq N$ *is* winning *if* $v(S) = 1$ *and* losing *if* $v(S) = 0$. *A* minimal winning coalition *(MWC) of a simple game* v *is a winning coalition in which defection of any player makes the coalition losing.*

We now define the following two *path coalitional games*.

Definition 3 (Path Coalitional Games). *For an unweighted directed/undirected graph,* $G = (V \cup \{s, t\}, E)$,

- *the corresponding* Edge Path Coalitional Game (EPCG) *is a simple cooperative game* (N, v) *such that* $N = E$ *and for any* $S \subseteq N$, $v(S) = 1$ *if and only if* S *admits an* s-t *path.*
- *the corresponding* Vertex Path Coalitional Game (VPCG) *is a simple cooperative game* (N, v) *such that* $N = E$ *and for any* $S \subseteq N$, $v(S) = 1$ *if and only if* S *admits an* s-t *path.*

We recall that for a game $G = (N, v)$, the corresponding *dual* game $G^D = (N, v^D)$ can be defined in the following way: $v^D(S) = v(N) - v(N \setminus S)$ for all $S \subseteq N$. For both EPCG and VPCG, and the corresponding duals $EPCG^D$ and $VPCG^D$ can be defined.

For any simple game, we can define a game which is the 'cost-based generalization'.

Definition 4 (Cost-based generalization). *For any given simple game* $G = (N, v)$ *we can define a* cost-based generalization C-$G = (N, v^c)$ *based on* cost vector $c = (c_1, \ldots, c_{|N|})$ *and* reward $r \in \mathbb{N}^+$. *For a coalition* $S \subseteq N$, *the value of the* $v^c(S) = r - \min_{S' \subseteq S, v(S')=1}(c(S'))$ *if* $v(S) = 1$ *and* $v^c(S) = 0$ *if* $v(S) = 1$.

The idea of a cost-based generalization is that each player demands some cost for its services being utilized and a coalition of a players get a reward r only if it is winning and gets the job done. Based on this formulation, we can define Edge Path Coalitional Games with costs C-EPCG and Vertex Path coalitional games with costs, C-VPCG. It is easy to see that for a game C-G, if $r = 1$ and the costs are all zero, then C-G is equivalent to G.

Observation 1. *C-ESPG is equivalent to the* value shortest path game (VSPG) *in [11].* $VPCG^D$ *is equivalent to the* simple path disruption game *in [2].*

[3] Throughout the paper, we assume $0 \in \mathbb{R}^+$.

We now define the following two *path intercept games*.

Definition 5 (Path intercept games). *For an unweighted directed graph, $G = (V \cup \{s, t\}, E)$, the corresponding* Edge Path Noncooperative Game (EPNG) *is a noncooperative game with two players, the* interceptor *and the* passer*. The pure strategies of the interceptor are the edges E and the pure strategies of the* path-chooser *is set $\mathcal{P}$ which contains all paths from vertex s to vertex t. If the edge used by the interceptor intersects with the chosen path, then the interceptor wins and gets payoff* 1*. Otherwise it loses and gets payoff* 0*.*

Vertex Path Noncooperative Games (VPNGs) *has an analogous definition to EPNGs except that the pure strategies of the interceptor are the vertices V and that if the vertex used by the interceptor intersects with the chosen path, then the interceptor wins and gets payoff* 1*.*

Both EPCG and VPNG can be generalized to the case with detection probabilities where the probability that the passer moving through edge e (or vertex v) will be detected if the interceptor inspects e (or v) is p_e (or p_v respectively).

3.2 Cooperative Solutions

A solution for a cooperative game consists of a distribution of the values of the coalitions that form. In this paper, we assume that the grand coalition consisting of all players always forms. Accordingly, a solution consists of a distribution of the value of the grand coalition over the players. Formally speaking, a solution associates with each cooperative game (N, v) a set of *payoff vectors* $(x_1, \ldots, x_n) \in \mathbb{R}^N$ such that $\sum_{i \in N} x_i = v(N)$, where x_i denotes player i's share of $v(N)$. Such efficient payoff vectors are also called *preimputations*. As such, solution concepts formalize the notions of fair and stable payoff vectors. In what follows, we use notation similar to that of Elkind et al. [6].

Given a cooperative game (N, v) and payoff vector $x = (x_1, ..., x_n)$, the *excess of a coalition S under x* is defined by

$$e(x, S) = x(S) - v(S),$$

where $x(S) = \sum_{i \in S} x_i$. We are now in a position to define one of the most fundamental solution concepts of cooperative game theory, viz., the core.

Definition 6 (Core). *A payoff vector $x = (x_1, \ldots, x_n)$ is in the* core *of a cooperative game (N, v) if and only for all $S \subset N$, $e(x, S) \geq 0$.*

A core payoff vector guarantees that each coalition gets at least what it could gain on its own. The core is a desirable solution concept, but, unfortunately it is empty for many games. Games which have a non-empty core are called *balanced*. The possibility of the core being empty led to the development of the *ϵ-core* [16] and the *least core* [10].

The *excess vector* of a payoff vector x, is the vector $(e(x, S_1), ..., e(x, S_{2^n}))$ where $e(x, S_1) \leq e(x, S_2) \leq e(x, S_{2^n})$. We denote the distinct values in the excess vector by $-\epsilon_1(x, v), -\epsilon_2(x, v), \ldots, -\epsilon_m(x, v)$, where $-\epsilon_i(x, v) < -\epsilon_j(x, v)$ for $i < j$.

Definition 7 (Least core). *For $\epsilon > 0$, a payoff vector vector x is in the* ϵ-core *if for all $S \subset N$, $e(x, S) \geq -\epsilon$. The payoff vector x is in the* least core *if it is in the ϵ-core for the smallest ϵ for which the ϵ-core is non-empty. We will denote by $-\epsilon_1(v)$, the minimum excess of any least core payoff vector of (N, v).*

It is easy to see from the definition of the least core, that it is the solution of the following linear program (LP):

$$\begin{array}{ll} \min & \epsilon \\ \text{s.t.} & x(S) \geq v(S) - \epsilon \ \text{ for all } S \subset N, \\ & \epsilon \geq 0, x_i \geq 0 \text{ for all } i \in N, \\ & \sum_{i=1,\ldots,n} x_i = v(N) \ . \end{array} \tag{1}$$

The nucleolus is a special payoff vector which is in the core if the core exists and is otherwise a member of the least core.

Definition 8 (Nucleolus). *A payoff vector x such that $x_i \geq v(\{i\})$ for all $i \in N$ and x has lexicographically the largest excess vector is called the* nucleolus*.*

The nucleolus is unique and always exists as long as $v(S) = 0$ for all singleton coalitions [14].

3.3 Noncooperative solutions

Let $\Delta(A)$ be the set of mixed strategies (probability distributions) on a finite set A. We assume the author is familiar with the concept of a Nash equilibria, and the fact that in two-player zero-sum noncooperative games, the maxmin strategies are equivalent to the Nash strategies. If such games are represented in normal-form, then it is well-known that a linear program formulation can help solve the game in polynomial time. However, this may not be the case for other representations for which the size of the linear program is exponential in the size of the input.

4 Least core of path coalitional game variants

Before considering other computational issues, we notice that the value of a coalition in EPCG and VPCG can be computed in polynomial time. For a coalition S in a EPCG/C-VPCG, use *Depth First Search* to check whether s and t are connected in a graph restricted to S. If not, then $v(S) = 0$. Otherwise, $v(S)$ is equal to 1.

Our first observation is that in all games EPCG, VPCG, $EPCG^D$ and $VPCG^D$, the core can be empty. In fact, the following proposition characterizes when the core of these games is non-empty:

Proposition 1. *The core of*

- *EPCG is non-empty if and only if there exists a s-t cut edge..*
- *VPCG is non-empty if and only if there exists a s-t cut vertex.*

– *$EPCG^D$ is non-empty if and only if there exists an edge (s,t) edge.*
– *(Bachrach and Porat [2]) $VPCG^D$ is non-empty if and only if there exists a vertex x such that (s,x) and (x,t) are edges in the graph.*

Proof. All cases follow directly from the fact that in a simple monotone game, the core is non-empty if and only if there exists a vetoer, i.e., a player $i \in N$ such that $v(N \setminus \{i\}) = 0$. For the dual games, note the following. Let (N, v^d) be the dual game and let x be a player such that $v(x) = 1$. We want to show that player x is a vetoer in (N, v^d) i.e., $v^d(N \setminus x) = 0$. We know that $v(N) = 1$. Then, by definition of dual, $v^d(N \setminus x) = v(N) - v(x) = 1 - 1 = 0$. Thus x is a vetoer in (N, v^d).

□

Since the core can be empty, the least core payoff assumes more importance. We will first present a general positive result (Theorem 1) regarding the computation of least core payoff for cost-based generalizations of simple games. To prove Theorem 1, we first require the following lemma:

Lemma 1. *Let $x = (x_1, \ldots, x_{|N|})$ be an preimputation of a C-G with an empty core. Then construct a weight function x' such that for each player i, $x'(i) = c_i + x_i$. Then, a minimal winning coalition S^* which minimizes total weight $x'(S^*)$ has the minimum excess with respect to payoff x.*

Proof. Assume for the sake of contradiction that S^* is not a minimum excess coalition and there exists a minimum excess coalition S which is not a minimum winning coalition which minimizes total weight $x'(S)$. We know that S is winning. If this were not the case, then the value of S is zero and the minimum excess is $e(x, S) \geq 0$. Without loss of generality there exists another winning coalition S' which is a minimum excess coalition. There may be a non-minimal winning coalition which is a minimum excess coalition but it will contain a minimal winning coalition which also has the minimum excess. We know that $x'(S') > x'(S^*)$ because if this were not the case, then S' would be a minimal winning coalition which minimizes total weight according to weighting x'. This means that $x(S') + c(S') > x(S^*) + c(S^*)$ which implies that $e(x, S') = x(S') - (r - c(S')) = x(S') + c(S') - r = x'(S') - r > x(S^*) - r = e(x, S^*)$. □

Theorem 1. *Let $G = (N, v)$ be the underlying simple game and $C-G = (N, v^c)$ be the cost-based generalization of (N, v). Assume that the representation of (N, v) is as compact as the cost function c. Then, if there exists a polynomial-time separation oracle for the least core LP of the underlying simple game, then a least core payoff of the cost-based generalization may be computed and verified in polynomial time.*

Proof. We first describe the least core LP for (N, v) as follows:.

$$\begin{array}{ll} \min & \epsilon \\ \text{s.t.} & x(S) - v(S) \geq -\epsilon \ \text{ for all } S \subseteq E \\ & \epsilon \geq 0, x_i \geq 0 \text{ for all } i \in N, \\ & \sum_{i=1,\ldots,n} x_i = 1 \ . \end{array} \tag{2}$$

Our first claim is as follows: Let $x = (x_1, \ldots, x_{|N|})$ be a preimputation of G. Then, the minimum excess and the minimum excess coalition can be computed in polynomial time. This follows from the assumption that there exists a polynomial-time separation oracle for the least core LP of the underlying simple game which computes the minimum excess and also computes the minimum excess coalition which signifies a violated constrain of the LP. Since $G = (N, v)$ is a simple game, therefore it follows that if each player $i \in N$ is given a weight $x'(i) = x_i$, then there exists a polynomial-time algorithm (let us call it Algorithm A) to compute a (minimal) winning coalition with the smallest total weight. The implicit assumption here is of course is that the weight function x' is non-negative.

The size of the linear program (1) is exponential in the size of game C-G, with an inequality for every subset of players. However, this linear program can be solved using the ellipsoid method and a separation oracle, which verifies in polynomial time whether a solution is feasible or returns a violated constraint [15]. We now demonstrate how algorithm A can be used to construct the separation oracle for the least core LP of C-G.

For a candidate solution $x = (x_1, \ldots, x_{|N|})$, where $x(N) = r$, construct the weighted function $x' = x_i + c_i$. From Lemma 1, we know that we only need to consider the set of minimal winning coalitions to find the coalition which gets the minimum excess. To do so, we require a polynomial-time algorithm to compute a (minimal) winning coalition which minimizes the weight $x'(S)$. Note that weights $x'(i)$ of all players $i \in N$ are non-negative. Since $c_i \geq 0$ and $x_i \geq 0$ for all $i \in N$, therefore, $x'(i) = c_i + x_i \geq 0$ for all $i \in N$. We already know that there exists a polynomial-time algorithm A which if given a non-negative weight vector x' for the players, can compute a winning coalition with the minimum total weight. The total weight of that coalition can be computed trivially. Use the known algorithm A to compute the winning coalition S^* with the smallest total weight $x'(S^*)$. Since the representation of (N, v) is as compact as the cost function c, we know that algorithm A still runs in polynomial-time. If we have $x(S^*)) + c(S^*) - r \geq -\epsilon$, then $x(S) - v(S) \geq -\epsilon$ for all $S \subseteq N$. Therefore, x is feasible. Otherwise, the constraint $x(S^*) + c(S^*) - r \geq -\epsilon$ is violated. This completes our argument that a polynomial-time separation oracle for the least core LP of the C-G can be constructed.

A payoff $x = (x_1, \ldots, x_{|N|})$ can be verified if it is in the ϵ-core by using the separation oracle. Since the minimum excess $-\epsilon_1$ of the least core payoff can be computed, therefore the separation oracle can also be used directly to check if the given payoff is in the least core. □

The assumption that the representation of (N, v) is as compact as the cost function c is crucial. One has to tread carefully in modifying the known polynomial-time separation oracle of the underlying simple game to the separation oracle for the cost-based generalization. For example, for weighted voting games with bounded weights, there exists a polynomial-time separation oracle for the least core LP [6]. However, it does not follow that the least core of a weighted voting game with small weights but large costs can be computed in

polynomial time. Based on this rule of thumb, we see that the least core can be computed for cost-based generalizations of the following games: *spanning connectivity games* [1] and *weighted voting games with bounded weights and also bounded costs*. Secondly, the success of the reduction crucially depends on the fact that the new weight function x' is non-negative. If this is not that case, then there is no guarantee whether algorithm A still works efficiently or works correctly at all. For example, in Theorem 2, if $x'(i) = x_i - c_i$ for all $i \in N$, there is a possibility of having negative weights and the computation of the shortest path becomes NP-hard. We now apply Theorem 1 to path coalitional games.

Theorem 2. *There exist polynomial-time algorithms to compute and verify least core payoffs of an cost-based generalizations of Edge Path Coalitional Game (C-EPCGs) and Edge Path Coalitional Game (C-EPCGs) for both directed and undirected graphs.*

Proof. We utilize Theorem 1 to prove the statement.

C-EPCGs: It is sufficient to show that for any preimputation, $x = (x_1, \ldots, x_{|E|})$ where $x(N) = 1$, can we compute the minimum excess coalition. Each player (edge) i has a weight x_i and the minimum excess coalition is an s-t simple path P with the smallest weight, that is the shortest s-t path. Use *Dijkstra's Shortest Path Algorithm* to compute the shortest path P from s to t in graph G^x and and then the minimum excess coalition is $E(P)$.

C-VPCGs: It is sufficient to show that for any preimputation, $x = (x_1, \ldots, x_{|V|})$ where $x(N) = 1$, can we compute the minimum excess coalition. Each player (node) i has a weight x_i and the minimum excess coalition is an s-t simple path P with the smallest weight, that is the shortest vertex s-t path. Then compute the shortest vertex weighted path P from s to t in graph G^x and then the minimum excess coalition is $V(P)$. Dijkstra's Shortest Path Algorithm can be used to compute the shortest vertex weighted path as following. The problem can be reduced to the classic shortest path problem in the following way: duplicate each vertex (apart from s and t) with one getting all ingoing edges, and the other getting all the outgoing edges, and an internal edge between them with the node weight as the edge weight. Use the algorithm to compute the shortest vertex path P from s to t in graph G^x. □

The least core of the game may not also be the least core payoff of the dual game. Therefore, we require new algorithms to compute the least core of dual coalitional path games.

Theorem 3. *The least core payoffs of C-EPCG^D can be computed and verified in polynomial time for both directed and undirected graphs.*

Proof. We utilize Theorem 1 to prove the statement.

C-$EPCG^D$: It is sufficient to show that for any preimputation, $x = (x_1, \ldots, x_{|E|})$ where $x(N) = 1$, can we compute the minimum excess coalition. Each player

(edge) i has a weight x_i and the minimum excess coalition is an s-t cut P with the smallest weight, that is the minimum weight s-t path. Use the maximum network flow algorithm (Chapter 27, [3]) to compute the minimum weight edge s-t cut C in graph G^x. This gives us the minimal winning coalition S with the minimum payoff and thereby the minimum excess.

C-$VPCG^D$: It is sufficient to show that for any preimputation, $x = (x_1, \ldots, x_{|V|})$ where $x(N) = 1$, can we compute the minimum excess coalition. Each player (node) i has a weight x_i and the minimum excess coalition is an s-t vertex cut P with the smallest weight, that is a minimum weight s-t vertex cut. Then compute the minimum weight s-t vertex cut in graph G^x and then the minimum excess coalition is $V(P)$. It is known that the minimum weight vertex s-t cut can be computed in polynomial time for directed graphs by standard network-flow methods. The network flow method to compute the minimum edge s-t cut can be used to compute the minimum vertex s-t cut as following. The problem can be reduced to the problem of min weight s-t edge cut of an edge weight directed graph in the following way: duplicate each vertex (apart from s and t) with one getting all ingoing edges, and the other getting all the outgoing edges, and an internal edge between them with the node weight as the edge weight. Set the weight of all original edges as infinite. We use existing algorithms to compute the minimum weight vertex s-t cut to construct the separation oracle for the C-$VPCG^D$ least core LP. □

We note that if instead of using s-t connectivity settings, we consider more than two terminals then some problems such as In-ϵ-Core become NP-hard. This follows from the fact that computing a min cut for more than two terminals is NP-hard.

5 A closer look at path coalitional games without costs

In this section, we take a closer look at simple path coalitional games without costs. We will refer to the minimum size of an s-t cut of a unweighted graph as c_E if we refer to edge cuts and as (c_V) if we refer to vertex cuts. Then we have the following theorem:

Theorem 4. *Consider an EPCG G_{EPCG} and VPCG G_{VPCG}. Then $\epsilon_1(G_{EPCG}) = 1 - 1/c_E$ and $\epsilon_1(G_{VPCG}) = 1 - 1/c_v$.*

Proof. Consider an EPNG based on graph G with detection probabilities $(p_1, \ldots, p_{|E|})$. The equilibrium or *maxmin* strategies of the interceptor are the solutions $\{x \in \Delta(E) \mid \sum_{v \in P} x_e \cdot p_e \geq val(G) \text{ for all } P \in \mathcal{P}\}$ to the following linear program, which has the optimal value $val(G)$.

$$\begin{array}{ll} \max & \alpha \\ \text{s.t.} & \sum_{e \in P} x_e \cdot p_e \geq \alpha \text{ for all } P \in \mathcal{P} \ , \\ & x \in \Delta(E) \ . \end{array} \tag{3}$$

We notice that if $p_e = 1$ for $e \in E$, then LP (3) is equivalent to LP (2). It is clear that maxmin strategy x of EPNG where $p_e = 1$ for all $e \in E$ is equivalent to the least core payoff of EPCG corresponding to G.

LP (3) is equivalent to LP 1 in [19] if p_e is set to 1 for each edge in both LPs. Washburn and Wood [19] conclude that maxmin strategy is obtained by constructing a graph $G^{c'}$ where $c'_e = 1/p_e$ and then computing the minimum weight s-t cut S. Each edge $e \in S$ is then given interdiction probability proportional to $c_e = 1/p_e$. It follows that if $p_e = 1$ for all $e \in E$, then $c'_e = 1/p_e$, and the minimum weight s-t cut S of $G^{c'}$ is simply the min cardinality s-t cut S of G. A maxmin strategy x of the interceptor, each edge in S is inspected with probability $1/|S| = 1/c_E$. Therefore for EPCG corresponding to G, the payoff of each simple s-t path or equivalently minimum winning coalition has payoff $1/c_E$ and the minimum excess $-\epsilon_1$ of the ESPG is $1/c_E - 1$.

We note that a similar analysis holds for VPCGs. □

Theorem 4 helps give a correspondence between EPCG and EPNG and also between VPCG and VPNG. We note that there is no such correspondence between, for example EPNG with detection probabilities and C-EPCG. Theorem 4 helps us formulate combinatorial algorithms to compute the least core of EPCGs and VPCGs (without costs). The problem of computing the least core reduces to computing a minimum cardinality edge cut (or vertex cut) of the graph and uniformly distributing the probability over the minimum cut. *Such least core payoffs are the extreme points of the least core convex polytope and in fact any other least core payoff is a convex combination of the extreme points.*

Our demonstrated connection of EPNGs to the corresponding coalitional EPCG helps examine refinements of the maxmin strategies such as the nucleolus. We will discuss this connection in further detail in this section.

The nucleolus of a coalitional game is arguably the unique and fairest solution concept which is guaranteed to lie in the core if the core is non-empty. The interpretation of the nucleolus in the non-cooperative setting is the maxmin strategy which maximizes the potential extra payoff if the path-chooser does not choose the optimal strategy. We observe that the nucleolus strategy is a refinement of the proper equilibrium strategy,

We will show that for certain graph classes like series-parallel graphs, the nucleolus strategy can be computed in polynomial time (Theorem 5).

Definition 9 (Series-parallel graph). *Let $G = (V, E)$ be a graph with source s and sink t. Then G is a* series-parallel graph *if it may be reduced to K_2 by a sequence of the following operations:*

1. *replacement of a pair of parallel edges by a single edge that connects their common endpoints;*
2. *replacement of a pair of edges incident to a vertex of degree 2 other than s or t by a single edge so that 2 degree vertices get removed.*

Denote the set of edge mincuts of a graph G by $\mathcal{C}(G)$. Denote by $C_e(G)$ the following value $\{S \in \mathcal{C}(G) \mid e' \in S\}$.

Lemma 2. *For an undirected series-parallel graph $G = (V \cup \{s,t\}, E)$, let x be a least core payoff of the corresponding EPCG and let $e \in E$ be such that $C_e(G) = \emptyset$. Then $x_e = 0$.*

Proof. Let $e = (a,b) \in E$ be such that $C_e(G) = \emptyset$ and assume for contradiction that there is a least core payoff of EPCG for G such that $x_e > 0$. Let the graph component in series with and left of e be G_1, the graph component in series with and right of e be G_2, the graph component in parallel and above e be G_3 and the graph component in parallel below e be G_4. Since $C_e(G) = \emptyset$ there exists no edge $e' \in G_3 \cup G_4$ such that $C'_e(G) > 0$. Now assume that the mincut value of G is c^*. The mincut C with size c^* must either be in G_1 or G_2. We also know that since x is a least core payoff, the length of the shortest s-t path in G is $1/c^*$. We show that if $x_e > 0$, then a transfer of payoff from certain edge in $G_3 \cup G_4 \cup \{e\}$ increases the minimum excess, thereby showing that x is not a least core payoff.

If there exists no shortest a-b path which includes e, then we know that e is present in no coalition which gets the minimum excess. Therefore e can donate its payoff uniformly to C and increase the minimum excess by x_e/c^*. Now assume that e is in one of the shortest a-b paths. Clearly, this is not the only simple a-b paths because if this were the case then e would be a bridge be one of the mincuts. We know that mincut value of $G_3 \cup G_4 \cup \{e\}$ is more than c^*. Let S be the minimum cut of $G_3 \cup G_4 \cup \{e\}$. We know that $|S| > |C| = c^*$. Then, we can show that the minimum excess of x increases if $x(S)$ is distributed uniformly over C. Each shortest s-t path if G^x has to pass one edge in C and one edge in S. The the weight of each edge in C has increased by $x(S)/|S|$ and the length of the shortest a-b has decreased by $x(S)/|S|$, the excess increases exactly by positive value $x(s)/|C| - x(S)/|S|$ without decreasing any other excesses. □

Theorem 5. *The nucleolus of EPCGs for undirected series-parallel graphs can be computed in polynomial time.*

Proof. We show that the problem of computing the nucleolus of EPCGs of undirected series-parallel graphs reduces to computing the parallel-series decomposition of the graph. There are known standard algorithms to identify and decompose series-parallel graphs (see e.g., [8]). The reduction is based on an inductive argument in which if we know the nucleolus of two graphs G' and G'', then we can also compute in polynomial time the nucleolus of the graph made by connecting G' and G'' in series or parallel. The proof by induction is as follows:

Base case: The base case if trivial. In any graph G with a single edge e connecting s and t, the (pre)nucleolus x gives payoff 1 to e.

Induction: Our induction involves two case: attaching two graph components in series and parallel. Consider two series-parallel undirected graphs G' and G'' and assume we already know that their nucleolus is x' and x'' respectively. We will show that computing the nucleolus of G formed by joining G' and G'' in series and parallel is polynomial-time easy.

Graph	Game	LC	nucleolus
general	C-EPCG	P (Th. 2)	?
general	C-VPCG	P (Th. 2)	?
general	C-$EPCG^D$	P (Th. 3)	?
general	C-$VPCG^D$	P (Th. 3)	?
series-parallel EPCG		P (Th. 2)	P (Th. 5)

Table 1. Summary of results

1. Assume we attach G' and G'' in series to obtain G. Let the size of any edge mincut be c' and any edge mincut be c''. If $c' < c''$, then by Lemma 2, there is no advantage of giving payoff to any edges in G''. Therefore, the nucleolus of G' is equal to the nucleolus of G and we are done. Assume that $c' = c''$. We recall that the nucleolus satisfies *anonymity* and *covariance* [13, 18]. Then due to Lemma 2 and covariance and anonymity property of the nucleolus, we have $x = (\alpha x', (1-\alpha)x'')$ where $0 < \alpha < 1$. Let m' and m'' be the smallest non-zero payoff of a player in x' and x'' respectively. We then show that $x = (\alpha x', (1-\alpha)x'')$ is the nucleolus if α has the unique value for which $m'(\alpha) = m''(1-\alpha)$ i.e., $\alpha = m''/(m'+m'')$. If this were true, then $x = (m''/(m'+m'')x', (1-m''/(m'+m''))x'')$. In this case, the minimum excess for x is $1/c'-1$ and the number of coalition achieving this in G is $|A| \times 2^{|B|}$ where A is the set number of simple paths in G^x and $B = \{e \in E(G) \mid C_e(G) = \emptyset\}$. We also know that the value of the second minimum excess is $1/c'-1+(m'\cdot m'')/(m'+m'')$. Now assume that there exists another payoff $y = (\alpha x', (1-\alpha)x'')$ for some $\alpha \neq m''/(m'+m'')$ such that y has a lexicographically greater excess vector than x. Clearly y is a least core payoff of G. Then the minimum excess for x is $1/c'-1$ and the number of coalition achieving this in G is still $|A| \times 2^{|B|}$. However the second minimum excess for y is less than $1/c'-1+(m'\cdot m'')/(m'+m'')$. Therefore, y has a smaller lexicographical excess vector than x which is a contradiction.
2. Consider two series-parallel undirected graphs G' and G'' and assume we attach them in parallel to obtain G. Let the size of any edge mincut of G' be c' and the size of edge mincut of G'' be c''. Both the mincut values can be computed in polynomial time for any graph. We know that the size of mincut of G is $c'+c''$. Then due to Lemma 2 and covariance and anonymity property of the nucleolus, we know that $x = (\alpha x', (1-\alpha)x'')$ where $0 < \alpha < 1$. We then show that $x = (\alpha x', (1-\alpha)x'')$ is the nucleolus if alpha has the unique value $c'/(c'+c'')$. Since the size of a mincut of G is $c'+c''$, every least core payoff y of is such that G^y has shortest path $1/(c'+c'')$. We want that every shortest s-t path which passes from G to have length $1/(c'+c'')$. This is only possible if $\alpha = c'/(c'+c'')$.

□

We conjecture that a similar approach may help construct a polynomial-time algorithm to compute the nucleolus of VPCGs for series-parallel graphs.

6 Conclusion

Path coalitional games provide a simple yet rich framework to model strategic settings in the area of network security and logistics. In this paper we analyzed different generalizations and variants of path coalitional games and classified the computational complexity of computing different cooperative and noncooperative game solutions. One key conclusion is the following insight: *under very weak conditions, linear programming techniques to compute least core payoffs of the underlying simple game can be used to compute the least core payoffs of cost-based generalization of the simple game.* Many of our positive results are based on separation oracles and linear programs. It will be interesting to see if there are purely combinatorial algorithms for the same problems. Apart from the EPCGs on series-parallel graphs, the complexity of computing the nucleolus is open for all other games. For all variants of path coalitional games, we assumed that each edge/vertex is owned by a separate player. It will be interesting to see if our positive results can be extended to the more general scenario where a single player may own more than one edge or vertex.

Acknowledgements

This material is based upon work supported by the Deutsche Forschungsgemeinschaft under grants BR-2312/6-1 (within the European Science Foundation's EUROCORES program LogICCC) and BR 2312/7-1. The research of Troels Bjerre Sørensen was supported by EPSRC award EP/G069034/1. We thank Mingyu Xiao and Evangelia Pyrga for useful comments.

References

1. H. Aziz, O. Lachish, M. Paterson, and R. Savani. Wiretapping a hidden network. In *Proceedings of the 5th International Workshop on Internet and Network Economics (WINE)*, volume 5929 of *Lecture Notes in Computer Science (LNCS)*, pages 438–446. Springer-Verlag, 2009.
2. Y. Bachrach and E. Porat. Path disruption games. In W. van der Hoek, G. Kaminka, Y. Lespérance, M. Luck, and S. Sen, editors, *Proceedings of the 9th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1123–1130, 2010.
3. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.
4. X. Deng and Z. Fang. Algorithmic cooperative game theory. In A. Chinchuluun, P. M. Pardalos, A. Migdalas, and L. Pitsoulis, editors, *Pareto Optimality, Game Theory And Equilibria*, volume 17 of *Springer Optimization and Its Applications*. Springer-Verlag, 2008.
5. X. Deng, T. Ibaraki, and H. Nagamochi. Algorithmic aspects of the core of combinatorial optimization games. *Mathematics of Operations Research*, 24 (3):751–766, 1999.

6. E. Elkind, L. A. Goldberg, P. W. Goldberg, and M. J. Wooldridge. Computational complexity of weighted threshold games. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI)*, pages 718–723. AAAI Press, 2007.
7. V. Fragnelli, I. Garcia-Jurado, and L. Mendez-Naya. On shortest path games. *Mathematical Methods of Operations Research*, 52(2):251–264, 2000.
8. X. He and Y. Yesha. Parallel recognition and decomposition of two terminal series parallel graphs. *Information and Computation*, 75:15–38, October 1987.
9. L. Khachiyan, E. Boros, K. Borys, K. M. Elbassioni, V. Gurvich, G. Rudolf, and J. Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, 43(2):204–233, 2008.
10. 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.
11. F. Nebel. *Shortest Path Games: Computational Complexity of Solution Concepts, MSc thesis*. Universiteit van Amsterdam, 2010.
12. N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1-2):166–196, 2001.
13. B. Peleg and P. Sudholter. *Introduction to the Theory of Cooperative Games*. Kluwer Academic, Boston, first edition edition, 2003.
14. D. Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal of Applied Mathematics*, 17(6):1163–1170, 1969.
15. A. Schrijver. *Combinatorial Optimization : Polyhedra and Efficiency (Algorithms and Combinatorics)*. Springer, 2004.
16. L. S. Shapley and M. Shubik. Quasi-cores in a monetary economy with non-convex preferences. *Econometrica*, 34:805–827, 1966.
17. J. C. Smith and C. Lim. Algorithms for network interdiction and fortification games. In *Pareto Optimality, Game Theory And Equilibria*, volume 17 of *Springer Optimization and Its Applications*, pages 609–644. Springer, 2008.
18. C. Snijders. Axiomatization of the nucleolus. *Mathematics of Operations Research*, 20(1):189–196, 1995.
19. A. Washburn and K. Wood. Two-person zero-sum games for network interdiction. *Operations Research*, 43(2):243–251, 1995.

Self-Configuring Sensors for Uncharted Environments*

Norman Salazar, Juan A. Rodriguez-Aguilar, Josep Lluis Arcos

IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
{norman,jar,arcos}@iiia.csic.es

Abstract. Sensor networks (SN) have arisen as one of the most promising monitoring technologies. So far the majority of SN deployments have assumed that sensors can be configured prior to their deployment because the area and events to monitor are well known at design time. Nevertheless, when the purpose of an SN is to monitor an environment such that the distribution and nature of its events is uncertain, we cannot longer assume that sensors can be configured at design time. Instead, sensors must be endowed with the capacity of autonomously reconfiguring and coordinating in order to maximize the amount of information they perceive over time. In this paper, we propose a low cost (in terms of energy and computation) collective distributed algorithm, which allows the sensors in an SN to collaboratively search for the configurations that maximise the information that they perceive based only on their local knowledge. We empirically show that the proposed algorithm helps an SN efficiently monitor environments where various dynamic events occur while showing high degrees of resilience to sensor failures.

1 Introduction

As technology continuously improves, it is becoming apparent that sensor networks (SN) are a powerful and versatile tool [1]. They have been employed by numerous applications on domains of a wide range of characteristics [2, 3]. Many of these applications rely on static sensor configurations (i.e pre-configured at design time), which can be detrimental. It has been argued that in real-world deployments the complexity, diversity, and dynamicity of the sensing requirements is a major issue that cannot be tackled through static configurations.

In particular, many of the events to sense have a dynamic nature. They may continuously expand or shrink (diffuse events), or move over the environment (moving events) [4]. Examples of these events are wildfires, glacier movements, gas plumes, and warm water currents, among others. Within such dynamic settings, it is necessary that sensors are configurable. Thus, sensors must adapt their configurations to vary the content, resolution and accuracy of their observations to maximize, in an energy-efficient manner, the information gathered over time. Therefore, an SN must count on *active sensing* capabilities[1]: "the

* A full version of this paper has been accepted in IEEE's SASO 2010

capacity of autonomously reconfiguring and coordinating its sensors in order to maximize the amount of information perceived over time". Henceforth, we consider a sensor's configuration as a schedule of (parameterized) actions the sensor must take to monitor/control/track some particular event. For instance, to monitor a wildfire, a sensor needs to measure the heat levels, humidity, look for carbon monoxide, and/or detect smoke.

Moreover, in large environments various distinct events are prone to occur at once (i.e., there is a spatial distribution of concurrent events). Hence, a sensor's configuration depends on the event(s) present on its geographic location (a sensor must be able to adopt as many configurations as events are possible). In these cases, it is likely that neighboring (close-by) sensors experience the same event(s), consequently making them require similar (the same) configurations. *Collective active sensing* [1] strategies are investigated for SNs whose sensors need to coordinate to collectively perform some sensing task. Regarding the settings above, a collective sensing strategy would be necessary to have sensors requiring similar configurations coordinate and cooperate towards a common goal (discovering the most useful configuration).

Collective active sensing strategies in dynamic environments have recently spurred research. On the one hand, according to the *dynamic region theory* (DRT) [5] sensors can select their configurations from a pre-defined set of configurations by identifying their spatial locations (regions) and their neighbors. Notice that this approach relies on the fact that the sensing requirements of each possible event and location are known at design time. In other words, it requires a thorough study of the deployment environment and a deep knowledge of its possible events. On the other hand, coalition formation based approaches have been also employed for active sensing. Sims et al [6] attempt to find the coalition of sensors to perform each task out of a set of available (sub)tasks in such a manner that some utility is maximized. Likewise DRT, this coalition-based approach also depends on a through knowledge (at design time) of the tasks or subtasks to perform. Additionally, the approach assumes that the sensors have (near-) complete information of the other sensors (or at least of a subgroup).

To summarize, the common assumption of previous works in the literature is that the deployment environment has been well studied, and thus that the sensor designers and the sensors themselves can make use of the available domain knowledge for configuration purposes. Nevertheless, this may not always be the case. It has been argued that sensor networks can be particularly useful in remote or hostile environments that have rarely been studied due to their inaccessibility [7]. Therefore, how to employ some collective active sensing in these *uncharted environments* remains an open issue that we address in this paper.

We propose a collective approach to monitor uncharted environments by embedding in an SN a distributed algorithm that: i) has a low computational overhead and a low energy consumption; ii) employs diffusion search to collectively find the most useful sensor configurations for the occurring events; iii) promptly reconfigures the sensors in response to dynamicity of the events; and

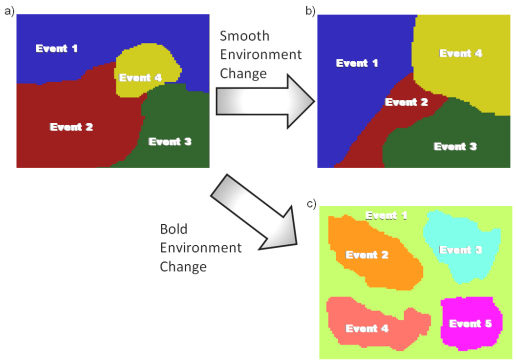


Fig. 1. a) Distribution of four different events in an environment; b) state of the environment after the (diffuse) events expand and/or shrink; c) state of the environment if all previous events disappear, and new distinct ones take their place.

iv) works when the sensor cannot rely on the available domain knowledge (uncharted environment).

2 Problem description

Our objective is to employ an SN to act in an uncharted environment, where acting refers to monitoring, controlling or tracking events. By uncharted environment we mean a location of which we only have partial domain knowledge. In other words, unlike in most current SN applications, we are not aware of the kind of events that may occur, nor of their possible locations in the environment.

An event stands as some phenomenon (of interest) that occurs in some geographic area of the environment. Moreover, various distinct events are likely to occur at the same time in different areas. Figure 1.a illustrates an environment where four different events occur simultaneously (each color represents a distinct event). Observe that the presence of multiple, spatially-distributed events has the effect of partitioning the environment. However, in uncharted environments the events (and thus its partitioning) may not be known at design time. Therefore, sensors need to be capable of configuring themselves according to events occurring in each location in such a manner that their selected configurations allow each of them to efficiently monitor/control/track the events.

Furthermore, in real-world situations, events usually change over time (i.e they are dynamic, with a diffuse or moving nature). For example, figures 1.a and 1.b depict the transition of four diffuse events (by expanding or shrinking); whereas the differences between figures 1.a and 1.c show a harsher transition in which the four events disappear and five new ones take their place (the new colors in figure 1 represent the new events). Hence, the sensors also need to adapt their configuration to cope with changes.

We assume that a large number of sensors may be deployed in a non-deterministic manner (e.g they are dropped from a helicopter) over a large environment.

Thus, only those events that are geographically available to the sensors' locations will be monitored [6]. We also assume that the sensors are technologically capable of monitoring many different types of events (i.e for any event there exists a sensor configuration capable of monitoring it). Following [5], we consider that collections of neighboring sensors are likely to reside on the same environment partition. Hence, experiencing the same event for periods of time during the operation of an SN. Therefore, they are also likely to require similar configurations.

Moreover, the uncharted nature of the environment means that the possible events are not cataloged (identified). Neither are the geographic areas where events may occur. In other words, a mapping between regions (locations) and events is not available. Consequently, the sensors need to have at their disposal a large set of possible configurations to cope with a large variety of possible dynamic events (i.e the sensors are highly configurable).

Moreover, we assume that each sensor has a preference structure [8] that expresses the satisfaction of any particular configuration when faced with a choice between different alternatives. Thus, a preference structure brings together all possible alternatives and represents a sensor's preferences over the set of possible configurations. In order to value a configuration, we assume that each sensor can measure the value of the information (observations) collected. This approach is similar to the one taken in [9], and in general it is a common in the data fusion and the tracking literature. Finally, the sensors may need to be some period of time in the environment to find their proper configurations. Therefore, even though sensors can be added or removed at any point in time, the sensor population does not fluctuate wildly [6].

The main challenges of the problem originate from the lack of a priori information in an uncharted environment. Because neither the possible events nor the environment characteristics are known beforehand (at design time), a region identification algorithm cannot be used to select the appropriate configurations (unlike [5]). Hence, the sensors must *search* for their proper configurations in a likely large configuration space. Additionally, although we assume that neighboring sensors are likely to experience the same event (thus requiring similar configurations), this may not always be the case. Sensors close to the frontier of two (or more) different events can have neighbors that require completely different configurations.

Moreover, there are also challenges related to the (technological) capabilities of sensors. Even though sensors are becoming more and more computationally powerful they are still somewhat limited. Therefore, any approach used to solve the problem should not take a significant amount of processing power away from the sensor duties. In other words, a low computational overhead is desired. Furthermore, it is well known that sensors are quite restricted energy-wise. Hence, the algorithms employed by the sensors must be as energy efficient as possible.

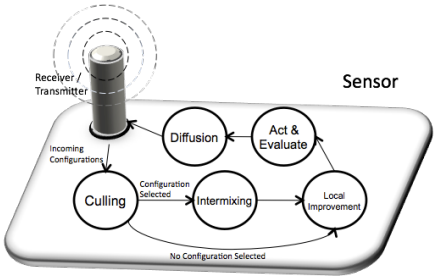


Fig. 2. Diffusion state transition.

3 A Collective Approach

In this section we propose an approach to solve the problem described in section 2, namely for sensors to dynamically find the most useful configurations required to monitor the dynamic events occurring in their geographical location. The proposed approach is designed to function in uncharted environments (where only partial domain knowledge is available) using the sensors' local feedback.

As stated before, it has been argued [5] that in environments with spatial events (e.g diffuse events) it is safe to assume that neighboring sensors may require the same (or a similar) configuration since they are prone to be monitoring the same event. Therefore, it is reasonable for sensors to collectively coordinate and cooperate to discover the most useful configurations.

In situations where the sensors are deployed to an uncharted environment, it may be the case that the only available useful information is the one provided by the sensors' own feedback function. Under such circumstances, cooperation becomes necessary since sensors can improve their partial domain knowledge (regarding the configurations) by sharing their local experiences. Moreover, the number of possible configurations may be very large, thus it may be unfeasible for the sensors to individually search for their configurations. Furthermore, cooperation and coordination are also useful computationally speaking, because even though sensors are becoming more powerful, their resources (e.g CPU) are still constrained. Hence, if multiple sensors search for the same configuration, they can save time and power by searching together. Once a sensor finds a good configuration, it can be promptly shared with its searching peers.

To that aim, we designed the *collective diffusion search* (CDS) as an algorithm based on the collective sharing of configurations amongst neighboring sensors. The state machine of the CDS (as implemented by each sensor) is shown in figure 2. In what follows we describe the main components of CDS and their rationale.

Diffusion We opt to employ diffusion as the component in charge of sharing the configurations since we regard it as an efficient (computation-wise) mechanism. In a sensor, diffusion consists in a broadcast (to its close-range neighbors) of its configuration. However, sending a broadcast (message) requires energy consumption. Hence, if a sensor's priority was to save as much energy as possible, then it is in its best interest to reduce its number of broadcasts. With that aim, a

sensor's likelihood of sending a broadcast can be regulated through a *probability of diffusion* ($p_{diffusion}$). The higher the value of this probability, the most likely a sensor is to broadcast its configuration. From a computational perspective, diffusion has a low overhead on the transmitting (sensor) side, since it amounts to sending a message without caring who will receive it. Nonetheless, receiving various broadcasts raises an issue, because a receiving sensor needs to decide what to do with these received configurations

Culling Attaching in each broadcast the utility of a configuration effectively provides a receiving sensor with the means to decide how to deal with multiple incoming configurations. This new information allows each receiving sensor to implement a culling component to dismiss useless (received) configurations. For instance, we implement this through a filter that selects the best received configuration (in a time window) and only if it is *better* than the sensors own.

Through diffusion and culling, groups of sensors that are close by and that experience the same event will adopt the same configuration. What is more, this has the effect of emerging of a common configuration per event partition, since a configuration is only diffused to where it is useful. For example, imagine an environment partitioned by two different events (e_1 and e_2) and consider a sensor, s_1, located on the e_1 partition but with at least a neighbor, s_2, on the e_2 side. Now assume that sensor s_1 knows the best configuration for its event (it has a high utility), which it will broadcast to its neighbors. Sensor s_2 on the e_2 partition receives the broadcast containing this highly valuated configuration, thus its culling will make the sensor adopt it. Nonetheless, since this configuration is not useful for event e_2, when employed by s_2 it will be valuated poorly. Therefore, even though sensor s_2 will still broadcast it to its neighbors in the e_2 partition, their culling filter will dispose it, halting the diffusion of the configuration on that side.

Intermixing. Notice that diffusion and culling do not have searching capabilities, at most they will establish the best configuration known by any of the sensors (per event). Thus, some searching needs to be incorporated since its unlikely to expect that some sensor knows its most useful configuration a priori. A low-overhead search method, consists in intermixing (combining) two configurations (the selected through culling and the sensor's current one) to create a new one. This can be regarded as using someone else's experience without completely forgetting your own. For instance, we implement this by combining a part of selected configuration with part of the sensor's current one (the parts are randomly decided), in such a manner that new configuration is constructed. Nevertheless, sensors cannot always depend on the usefulness of their neighbors configurations (e.g if all the neighboring sensors have the same configuration and it is not useful).

Local improvement This component makes each sensor capable of searching for new configurations without depending on its neighbors. Moreover, this not only helps to improve the existing configurations, it is particularly necessary in environments with dynamic events since the events can disappear or appear unexpectedly. Local improvement can be accomplished (without expending much

processing power) by introducing a random change to a sensor's configuration with some probability ($p_{improvement}$). Various disciplines have shown this to be effective [10].

3.1 Collective Diffusion Search

Altogether, in collective diffusion search each sensor continuously attempts to propagate its configuration while trying to improve it at the same time. The sensor receives some broadcasts which are then filtered through culling in an attempt to determine if there is a better configuration. In case there is, the sensor's configuration and the selected one are combined in an attempt to create a new (and ideally better) configuration. Afterwards (or if culling fails to select a configuration), local improvement can be used to continue the search for the best configuration. Once this is over, the sensors configuration is used and its utility valuated (act and evaluate in figure 2) through the feedback generated by the performed actions. Lastly (is a matter of perspective) the sensor wraps its configuration along with its utility into a message for broadcasting. An execution of the state machine shall hereafter be referred as a *communication cycle*.

Overall, this approach can be regarded as a distributed evolutionary process, since configurations evolve through time as a consequence of the constant application of diffusion, combination and local improvements. Once the configurations cannot evolve anymore, the end result is a stabilized set of useful configurations.

To summarize, collective diffusion search is a low overhead, but powerful distributed algorithm that when embedded in each sensor empowers them to dynamically find the most useful (utility-wise) configurations even when only partial domain knowledge is available (uncharted environments). The algorithm can be easily implemented in a sensor, since its formed of four *lightweight/low overhead* components: `diffusion`, `culling`, `intermixing`, and `local improvement`.

4 Experimental Results

The aim of our experiments is to verify two hypotheses, if through collective diffusion search: i) the sensors can find the configurations that maximize their utility according to the events that occur in their location; and ii) the sensors can reconfigure themselves in the response to dynamic events.

To that end, we designed three types of experiments: 1) event recognition (figure 1.a): recently deployed sensors must find the configurations needed to monitor the events in the environment; 2) sensor reconfiguration against *smooth event changes* (transition from figure 1.a to 1.b): the existing events' area of covering changes over time (diffuse events); and 3) sensor reconfiguration against *bold event changes* (transition from figure 1.a to 1.c): existing events disappear and completely new events (different covering and features) take their place.

It is important to understand that in real-like situations dynamic events may change slowly through time. However, in our experiments we opt to make it harder for the sensors by introducing the changes in a more abrupt manner. In a real deployment such abruptness may actually occur from a sensor's perspective,

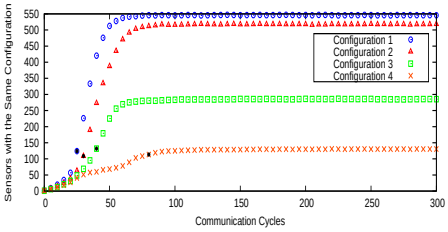


Fig. 3. Results of convergence after the initial deployment. The black dots mark when the best configurations were found.

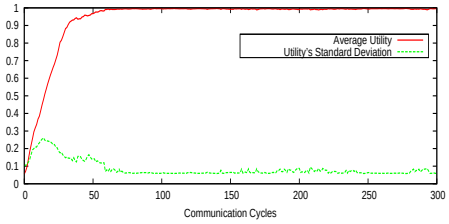


Fig. 4. Normalized average and standard deviation utility (in a single simulation) of the sensors in the region of event 3.

since the sensors may sleep for long intervals. Thus, every time a sensor awakens the events may have different features.

Each of our *experiments* consists of 50 discrete event simulations, each one up to 5000 ticks. Our simulation environment is formed by a 100 x 100 grid initially covered by four distinct events. Figure 1.a depicts with different colors the form of each event at start up. Observe that all events are different from each other in area, shape, and borders. During a simulation a set of 1500 sensors is randomly deployed unto this environment. However, because we are evaluating their self-configuration capability in an uncharted environment, none of the sensors is aware of the environment partitioning nor of the possible dynamic events. Hence, each sensor starts with a random configuration.

A sensor configuration is given by an ordered sequence of 5 actions selected from a pool of 20 possible actions (i.e $\|K\| = 20^5$). Following the assumption in section 2, our environmental feedback function valuates if the actions in the configuration are useful or not. The parameters for the CDS were set to (unless otherwise indicated): a broadcast range of 4 cell and $\sim 20\%$ likelihood of broadcast per sensors at a given point in time ($p_{diffusion} = 0.20$); the sensor's culling occurs once during each tick; and a local improvement probability of 0.0008.

To measure the usefulness of the CDS approach described in section 3, we counted the number of sensors that found the configuration needed to monitor the event in their location. The counts of each simulation in the experiment were then aggregated using the inter-quartile mean. From here on, *configuration 1,2,3...* refers to configuration employed by the majority of the sensors located in the area of *event 1,2,3....* Furthermore we used communication cycles (also defined in section 3) as a time scale for the measurements.

4.1 Initial Deployment

The event recognition experiments aim to verify if CDS allows recently deployed sensors to find and adopt their most useful (utility-wise) configurations required by their location. Moreover, we are interested in measuring: i) how fast these configurations are found and adopted by sensors; and ii) how much energy (as a consequence of the transmitted messages) it requires.

Figure 3 shows that CDS is quite effective in finding the most useful configurations for almost all the sensors. Observe that once the configurations are found

(black dots in the figure) the sensors promptly adopt them. These configurations are found at $\sim 20, \sim 30, \sim 40$ and ~ 60 communication cycles for each of the four events respectively. Figure 4 shows in more detail (for a single simulation and for the sensors in event 3) the search and adoption of the most useful configuration. Firstly, there is an initial variety of different configurations (initial spike in the standard deviation). Next, as a result of the diffusion and culling components collectives of sensors adopt similar (not so bad) configurations. The adopted configuration is then improved (through intermixing and local improvement) at each communication cycle, and with each improvement (observable in the average utility) more sensors start to adopt it (shown by the decrease in the deviation). This continues until the configuration that provides the highest utility is found and consequently adopted by most of the sensors.

However, observe that even though the average utility stabilizes (~ 1.0), the standard deviation still indicates that not all sensors adopt same configuration. On the one hand, this occurs because at any point in time the local improvement component causes a small number of sensors to try new configurations. On the other hand, because a sensor broadcasts its configuration to all its neighbors, the sensors in the frontiers (near the geographic border of two or more different events) may receive conflicting configurations from their neighbors (similar to the example in section 3).

Moreover, notice that depending on features of the event, some configurations require more time to be adopted by sensors. This appears to be related to the dimension of the area occupied by the event, and thus by the number of sensors that require the same configuration. *Event 4* is a particularly pronounced example of this effect (sensors localized in the region of this event take the longest to find the best configuration and thus to adopt it: ~ 60). Event 4 encompasses the smallest area and has the lowest number of sensors by far, which seem to give empirically credence to the idea that the number of sensors influence how fast a configuration can be found and adopted. From the algorithmic point of view, this is reasonable in a collective approach because fewer sensors are looking for the same configuration. Although there may be another factor to consider, the location of the event. Observe that event 4 is completely surrounded by the other events, which means that sensors in that area are constantly receiving conflicting configurations from their neighbors. Additionally, because of its small dimensions a broadcast originated in its frontiers may cover a significant area of the event. In other words, sensors as far as the center of the event may receive conflicting configurations.

The previous results have shown that collective search is an effective self-configuration approach. However, it is not clear if our approach provides an advantage against individual search. Therefore, we performed experiments with sensors employing an individual search mechanism instead of a collective one. CDS shares various similarities with evolutionary algorithms (EAs)[11], to the point that it could be considered a distributed EA. Hence, we employ a classic EA as the individual search approach to compare with. Specifically, we endowed each sensor with an EA to help it find its best configuration. The chosen EA

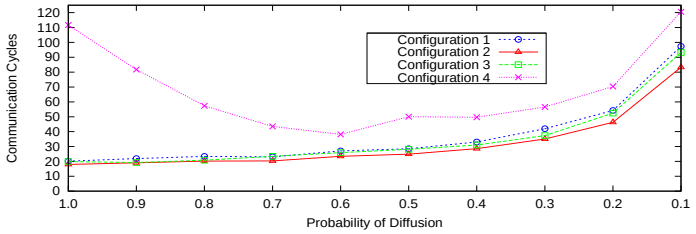


Fig. 5. Communication cycles required by sensors to adopt the best configuration using different diffusion probabilities.

was a classic genetic algorithm [12] with a population of 20 configuration (per sensor). In other words, to complete an iteration of the algorithm each sensor needs to evaluate 20 configurations. This is significantly higher from the CDS which only requires one evaluation per iteration.

As to the actual results of the experiments, each sensor needs around 2000 iterations to individually find its best configuration. In other words, 4×10^5 configurations are evaluated per sensor. This is extremely high when compared to the CDS, which needs around 60 (in the worst case) to find the best configuration for *all* sensors. Hence, we can conclude that collective search can be considered as the more appropriate choice for sensors self-configuration.

4.2 The Role of Diffusion

In section 3, it was stated that a sensor can save energy by reducing its number of broadcasts, and that this is regulated by the probability of diffusion. Moreover, the experiments in the previous subsection showed that even though at any given time only $\sim 20\%$ of the sensors were broadcasting the best configuration was still found and shared in a reasonable amount of time (~ 70 communication cycles). Therefore, next we study how different probability values affect the CDS.

To that end we repeated the previous experiment with different diffusion probabilities. Figure 5 summarizes the experimental results by showing the number of communication cycles required for the sensors (at least 90%) to find and adopt the most useful configuration (according to their localization).

Observe that for most events the higher the diffusion probability (the larger the number of sensors broadcasting) the faster the most useful configurations are adopted. However, small (area-wise) events do not follow this trend. With a high diffusion probability, sensors also require more time to find and adopt the most useful configuration. Such effect occurs because (almost) constant diffusion diminishes the search capabilities of the CDS in small areas, which were already reduced because of the low number of sensors in the event.

Although, at first glance it may appear desirable to use a somewhat high probability of diffusion, such probability means energy-wise. When sensors broadcast configurations at every communication cycle, the number of transmitted messages used by the sensors to reach their best configurations (for the four events) is of ~ 167500 (or ~ 110 messages per sensor). Whereas using a low value (0.2) may slightly increase the number of communication cycles needed by

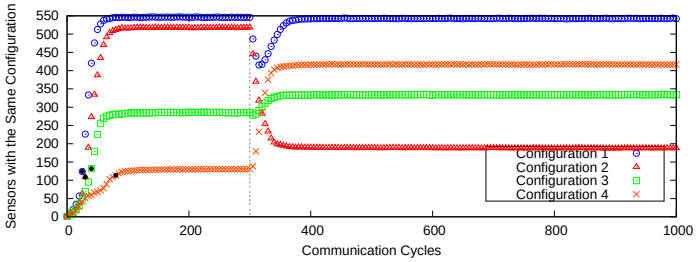


Fig. 6. Reconfiguration after smooth environmental change (at 300 cycles).

most sensors to adopt their configuration but the reduction in the number of messages is quite significative ($\sim$ 18000 overall, or $\sim$ 12 per sensor).

Then, we can conclude that the probability of diffusion represents a straightforward parameter that controls the trade off between the time needed to find the proper configurations and the energetic consumption.

4.3 Dynamic Events

The previous experiments have shown that CDS allows sensors to configure themselves according to the existing environmental events. However, as stated in section 2 events are usually dynamic, i.e they change over time. Therefore, the purpose of this section is to verify if sensors with CDS can reconfigure themselves in response to dynamic events. In what follows, we present the two different types of dynamic events against which we tested the CDS.

Smooth event changes In these experiments some time (300 cycles) after the initial deployment, the area covered by each of the events expands or shrinks, but the configuration required by each event stays the same (transition between figures 1.a and 1.b). In such situations the sensors should reconfigure themselves by redistributing the existing configurations without requiring to search for new configurations (through the diffusion and culling components). Figure 6 shows that through CDS sensors respond promptly to changes in the environment (marked by the vertical line). The sensors, for which the event in their location changes, smoothly transition to their newly required configurations by adopting them from their stable neighbors. For instance, the sensors monitoring *event 1* (configuration 1) suffer the most complicated transition since the region of the event shrinks from one side and expands from another. The configuration redistribution for these events is observable in figure 6 (between the $300 - 400$ communication cycles) by the sudden decline of the sensors with *configuration 1* followed by a sudden increment. Since events may be very dynamic and thus change continuously, we conducted experiments in an scenario that continuously transition back and forth between figures 1.a and 1.b. Experiments showed that every time a smooth change occurred the sensors reacted promptly to adopt their new configuration ($\sim$ 30 communication cycles).

Bold event changes For these experiments we model a more extreme dynamic event: the sudden disappearance of the existing events and the sudden appearance of completely new ones (transition between figures 1.a and 1.c). Thus, unlike

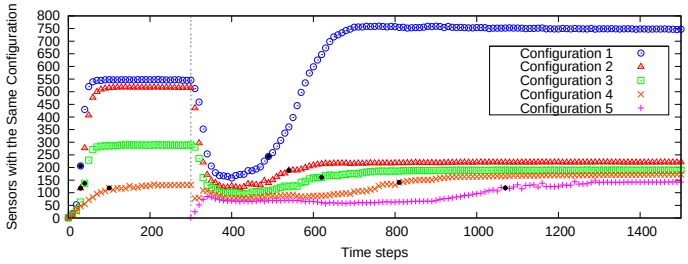


Fig. 7. Reconfiguration after a bold environmental change (at 300 cycles).

in the smooth change, sensors must search (again) for the most useful configurations required by the new events. Furthermore, since most of the sensors had (previously) stable configurations (that now are completely useless) there is an overall lack of configurations diversity. Hence, during this stage diffusion and intermixing does not provide much help, and its up to the local improvement component to increase the configuration diversity.

Figure 7 shows that after the bold change (the vertical line at 300 cycles), the sensors start to diversify their configurations (the number of sensors with the same configuration declines). Once some sensor(s) find some useful configurations they begin to collectively improve it until the most useful ones are found. However, unlike during the initial deployment, it takes longer ($\sim$ 600 cycles) to find the most useful configurations for all the five events. The first cause is the above-mentioned lack of diversity. The second, like the experiments in subsection 4.1, is the size and frontiers of the events that affect how fast configurations are found and adopted. For instance, after the bold change four of the five regions are considerably small (with a low number of sensors located in events), which slows down the search. Furthermore, even though the new *event 1* covers a large area and its configuration is promptly found ($\sim$ 100 cycles) it take some time for most sensors located on the event to adopt it ($\sim$ 200 cycles) because the region of event 1 has a lot of frontiers which affect the effectiveness of diffusion.

To summarize, from these experiments we conclude that: i) through CDS sensors can reconfigure themselves in response to both smooth and bold environmental changes; ii) the diffusion component makes the CDS particularly effective against smooth changes; iii) local improvement is a key component to accomplish reconfiguration if a sudden event where to appear; and iv) the speed of the search and adoption of configurations are indeed affected by the dimensions and frontiers of the events.

4.4 Fault Resilience

The experimental results so far have shown that CDS allows a collective of sensors to successfully (re)configure themselves over uncharted environments. However, our experiments ignored that hazardous conditions are frequently a reason why such environments are uncharted. Thus, sensors may be more prone to failures, e.g because of the deployment impact (sensors may be thrown from an helicopter), fire or extreme heat and animal or vehicular accidents [13]. Therefore, in this section we show the *fault-resilience* capabilities of the CDS.

Failure Probability	Working sensors in the Best Configuration			
	Event 3	Event 1	Event 2	Event 4
None	98.54 % / 291	99.42 % / 540	98.28 % / 535	98.47 % / 133
Medium	96.66 % / 236	99.54 % / 429	97.57 % / 428	94.03 % / 106
High	85.84 % / 147	84.29 % / 270	91.87 % / 271	62.43 % / 66
Very High	19.64 % / 87	22.05 % / 165	19.74 % / 159	19.37 % / 39

Table 1. Percentage of working sensors when failures on reception.

Failure probability	Event 3	Event 1	Event 2	Event 4
None	52	54	46	70
Medium	79	79	66	148
High	321	347	154	>350

Table 2. Communication cycles needed for different reception failure probabilities.

One could naively assume that the worst failure case would be if some sensors cease to function completely. Nonetheless, this type of failure is meaningless for CDS, since it is a collective approach that relies on the existence of multiple sensors. The actual worst failure cases are malfunctioning reception and/or malfunctioning sensing (measuring). The former, mainly refers to a lack of capability in receiving communications from other sensors, whereas the latter refers to measuring errors resulting from either damaged or badly calibrated equipment. Therefore, we can say that the CDS is fault resilient if it is capable of dealing with such kind of (worst case scenario) failures. To that end we designed two sets of failure experiments to evaluate the CDS: a malfunctioning reception scenario; and a malfunctioning reception and sensing scenario.

Malfunctioning Reception These experiments model a scenario where during the initial deployment some sensors (with probability $p_{failure}$) become incapable of receiving communications, thus making them incapable of actually adopting their proper configuration. However, since they are still capable of transmitting they will constantly send useless communications (their initial random configuration) to their neighboring sensors. We ran experiments where the failure probability was medium ($\sim$ 20% of the sensors fail), high medium ($\sim$ 50% of the sensors fail) and very high ($\sim$ 70% of the sensors fail). Table 1 shows the percentage of the functioning sensors that found the best configuration for their event. Observe that even when around 20% of the sensors fail, almost all of the remaining functional ones are able to find and adopt the best configuration. However, as the number of sensors that fail increases the number of sensors that can establish the best configuration decreases. For instance, in the unrealistic case of 70% failure probability, very few of the remaining functional sensors can establish their needed configuration. This is to be expected since the number of failing sensors sending useless configurations is very high. Moreover, from the event 4 results we can once again observe that the size of the event (specifically the number of sensors in the event) can affect the CDS. Nevertheless, overall the level of resilience shown by the CDS for this type of failure is very high.

Regarding how failure affects the convergence time, table 2 shows the number of communication cycles needed so that 90% of the sensors establish the best configuration. As expected, the higher the failure probability the more communication cycles that are needed.

Failure probability	Working sensors in the Best Configuration			
	Event 3	Event 1	Event 2	Event 4
None	98.54 % / 291	99.42 % / 540	98.28 % / 536	98.47 % / 132
Low	97.68 % / 262	97.81 % / 486	97.81 % / 481	90.33 % / 120
Medium	93.33 % / 235	90.71 % / 429	96.59 % / 428	25.13 % / 106
High	03.43 % / 202	11.90 % / 378	04.26 % / 375	05.57 % / 93

Table 3. Percentage of working sensors when failures on sensing and reception.

Failure probability	Event 3	Event 1	Event 2	Event 4
none	52	54	46	70
low	92	103	89	291
medium	266	299	215	0

Table 4. Convergence time when failures on sensing and reception.

Malfunctioning Sensing and Reception Next we model a more extreme failure situation. Besides being incapable of receiving incoming communications, failing sensors also suffer from sensing errors. Moreover, for our experiments we specifically model the worst kind of sensing error. That is, that a failing sensor will always valuate its configuration as the best, regardless of its actual usefulness. In other words, these sensors will constantly mislead their non-failing neighbors by broadcasting their configurations as if it were the best. The experiments were ran with a duration of 400 communication cycles and the following failure probabilities where employed: low ($\sim$ 10% of the sensors fail), medium ($\sim$ 20% of the sensors fail) and high ($\sim$ 30% of the sensors fail). Notice that these probabilities are lower that those in the previous section because this type of failure is considerably worst than the one employed there.

We observe from table 3 that the CDS has a good resilience against a failure probability of 10%. For almost all events more than 97% of the sensors reach their best configurations, and it is only *event 4* (the event with the smallest region) the one that reaches just 90%. Furthermore, sensors sensing *event 4* are those that suffer the most when the failure rate increases. This is not surprising since its low number of sensors makes the non-failing sensors more prone to succumb to the misleading configurations broadcasted from the failing sensors. Nevertheless, the resilience shown by the CDS for the other event regions is very good ($\sim$ 90 %), specially for this kind of failure.

As for the convergence time (see table 4) it is not surprising to observe that more communication cycles are needed for this type of failure. However, the number of cycles is still low enough to make the CDS practical (in particular when compared to an individual search). Overall, the experiments show that the collective diffusion search mechanism has a considerable good resilience against sensor failures. These results are encouraging since failures are more prone to occur in uncharted environments.

5 Conclusions

In this paper we presented collective diffusion search (CDS) as a low overhead distributed algorithm that empowers sensors in a sensor network to collectively find the configurations needed to monitor the events occurring in an uncharted environment. Moreover, our empirical experiments showed that through this

algorithm sensors not only are capable of configuring themselves, they can also reconfigure themselves in response to various levels of dynamic changes in the events. Furthermore, our results indicate that CDS is quite efficient energy-wise since the number of message transmissions required by the sensors' self-configuration is quite low. Additionally, the CDS has shown to be considerably resilient against sensors failures. These are all highly desirable properties for any sensor's algorithm.

Finally, even though CDS accomplished its purpose in all of our experimental scenarios, we observed that the number of sensors, and the dimensions and locations (frontiers) of the event affect the speed of the sensors' re-configurations.

Acknowledgments

N. Salazar thanks CONACyT. Work funded by projects TIN2009-14702-C02-01, CSD2007-0022, and the Generalitat of Catalunya grant 2009-SGR-1434.

References

1. M. Vinyals, J. A. Rodriguez-aguilar, J. Cerquides, A survey on sensor networks from a multi-agent perspective, The Computer Journal.
2. N. Xu, A survey of Sensor Network Applications, http://courses.cs.tamu.edu/rabi/cpsc617/resources/sensorf.
3. D. J. Nagel, Wireless Sensor Systems and Networks: Technologies, Applications, Implications and Impacts, http://intranet.daiict.ac.in/ ranjan/isn2005/papers/APP/wireless.pdf.
4. J. Yin, D. H. Hu, Q. Yang, Spatio-temporal event detection using dynamic conditional random fields, in: In Proceedings of IJCAI'09, 2009.
5. R. M. Ruairí, M. T. Keane, An energy-efficient, multi-agent sensor network for detecting diffuse events, in: In Proceedings of IJCAI'07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007, pp. 1390–1395.
6. M. Sims, C. Goldman, V. Lesser, Self-Organization through Bottom-up Coalition Formation, in: Proceedings of AAMAS 2003, ACM Press, 2003, pp. 867–874.
7. K. Martinez, J. K. Hart, R. Ong, Environmental sensor networks, Computer 37 (2004) 50–56.
8. Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. A. Padget, S. Phelps, J. A. Rodríguez-Aguilar, P. Sousa, Issues in multiagent resource allocation, Informatica (Slovenia) 30 (1) (2006) 3–31.
9. J. Kho, A. Rogers, N. R. Jennings, Decentralized control of adaptive sampling in wireless sensor networks, ACM Trans. Sen. Netw. 5 (3) (2009) 1–35.
10. M. S. San Miguel, V. M. Eguiluz, R. Toral, K. Klemm, Binary and multivariate stochastic models of consensus formation, Computing in Science and Eng. 7 (6) (2005) 67–73.
11. T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, 1996.
12. Z. Michalewicz, Genetic Algorithms+Data Structures=Evolution Programs, 3rd Edition, Springer Verlag, 1996.
13. J. L. Bredin, E. D. Demaine, M. Hajiaghayi, D. Rus, Deploying sensor networks with guaranteed capacity and fault tolerance, in: MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing, ACM, New York, NY, USA, 2005, pp. 309–319.

Coalitional Voting Manipulation: A Game-Theoretic Perspective

Yoram Bachrach, Edith Elkind, Piotr Faliszewski

Abstract

The computational social choice literature has successfully studied the complexity of manipulation in various voting systems. However, the existing models of coalitional manipulation view the manipulating coalition as an exogenous input, ignoring the question of the coalition formation process. While such analysis is useful as a first approximation, a richer framework is required to model voting manipulation in the real world more accurately, and, in particular, to explain how a manipulating coalition arises and chooses its action. In this paper, we apply tools from cooperative game theory to develop a model that considers the coalition formation process and determines which coalitions are likely to form and what actions they are likely to take. We explore the computational complexity of several standard coalitional game theory solution concepts in our setting, and investigate the relationship between our model and the classic coalitional manipulation problem as well as the now-standard bribery model.

1 Introduction

Voting is a standard method of preference aggregation in multi-agent environments. It allows the agents (voters) to make joint decisions by selecting the most suitable alternative from a given set. However, in settings where voters are selfish and aim to optimize their individual utility, voting suffers from a serious problem: essentially all voting rules are manipulable, i.e., a voter may benefit from misrepresenting her preferences over the alternatives [Gibbard1973, Satterthwaite1975]. Consequently, classifying voting rules according to their resistance to manipulation has been an active research topic in the last decade (see [Faliszewski and Procaccia2010] for an overview)

While the possibility of manipulation by a single voter presents a grave concern from a theoretical perspective, in real-life elections this issue does not usually play a significant role: typically, the outcome of a popular vote is not close enough to be influenced by a single voter. Indeed, a more significant problem is that of coalitional manipulation, where a group of voters coordinates their actions in order to affect the election outcome. The problem of coalitional manipulation was first explicitly introduced by [Conitzer *et al.*2007], where the authors also initiated its analysis from the computational perspective. Since then, a number of results on the computational complexity of coalitional manipulation for a variety of voting rules have been obtained (see the Related Work section below).

However, the model of coalitional manipulation proposed in [Conitzer *et al.*2007] abstracts away some of the issues that are crucial for realistic modeling of coalitional manipulation scenarios. Specifically, this model assumes that the set of all voters is partitioned into two groups: the honest voters and the manipulators. The honest voters have preferences over the candidates, while the manipulators are single-minded: they simply want to get a specific candidate elected. Thus, the set of manipulators is an *exogenous* variable, given as a part of the input. This definition does not *explain* how the manipulating coalition forms or how it decides which candidate to promote. Arguably, this provides an overly simplistic view of reality: it is natural to expect that the would-be manipulators start out by having preferences over the entire set of candidates, but then decide to cooperate with each other, as they are not satisfied with the outcome of truthful voting.

Against this background, our goal in this paper is to provide an *endogenous* model of coalitional manipulation that is based on coalitional game theory. We assume that all agents have preferences over the set of candidates; we make the standard assumption that these preferences are common knowledge. In addition, a subset of the agents are strategic and would consider forming a manipulating coalition if they can profit by doing so. Given this setup, each voting rule induces a coalitional game, where the players are the strategic agents (we will refer to them as *colluders*), and the set of outcomes that are feasible for a coalition is determined by the set of candidates that the players in that coalition can turn into election winners. We consider a *transferable utility* model, where the colluders have comparable utilities (given in a "common currency") for each candidate and that they can commit to making payments to each other.

We study several natural computational problems regarding the coalitional game induced by the voting domain, such as finding the optimal action a coalition can take, identifying coalitions whose optimal action is to support a certain candidate, calculating a player's power in the game and testing whether an outcome is in the core. While exploring these issues, we also examine the relation between our model and the classic coalitional manipulation model and the voting bribery model [Faliszewski *et al.*2009]. Our contributions fall into two main categories. First, we introduce a cooperative game-theoretic model of voting manipulation, and study the complexity of natural solution concepts in this model. Second, and on a more fundamental level, even though our work is motivated by a critique of the standard framework of voting manipulation, we show that many classic computational social choice results have natural interpretations in our game-theoretic model. For example, results on the complexity of coalitional manipulation—as defined in [Conitzer *et al.*2007]—translate into results on the complexity of computing coalition values, and results on the complexity of bribery—as defined in [Faliszewski *et al.*2009]—translate into, e.g., results on the complexity of testing whether a given coalition is stable. We believe that our model is a useful formalism that captures many aspects of coalition formation in voting.

Related work The complexity of coalitional manipulation, as defined in [Conitzer *et al.*2007], received a lot of attention in the recent literature (see, e.g., [Hemaspaandra and Hemaspaandra2007, Faliszewski *et al.*2010, Xia *et al.*2009, Walsh2009, Xia *et al.*2010]; a more exhaustive list is provided in [Faliszewski and Procaccia2010]). However, none of these papers discusses the issue of manipulating coalition formation. A recently proposed model

of safe strategic voting [Slinko and White2008] addresses this issue using an approach that is different from ours: specifically, under this model a single voter announces a manipulative vote, and may be followed by other voters with the same preferences (see also [Hazon and Elkind2010] for the algorithmic analysis and extensions of the model of [Slinko and White2008]). While the approach of [Slinko and White2008] is more suitable when it is difficult for the manipulators to coordinate, our model is more appropriate when coordination is not an issue; thus, the two approaches complement each other. There is also a number of very recent papers that analyze strategic behavior in voting using the tools of *non-cooperative* game theory (see [Desmedt and Elkind2010, Xia and Conitzer2010, Meir *et al.*2010] and the references therein).

Organization of the paper The paper is organized as follows. Section 2 provides background on (computational) social choice and coalitional game theory. In Section 3 we formally introduce our model. Section 4 focuses on the problem of computing coalitional values. In Section 5 we study the complexity of testing if there is a manipulating coalition supporting a given candidate. Section 6 considers players' power in the game, and, in particular, computing the Shapley values. Section 7 investigates coalitional stability. We conclude in Section 8. We omit most proof due to space constraints.

2 Preliminaries

We write $\mathbb{N} = \{0, 1, 2, \dots\}$, and given a vector $\mathbf{x} \in \mathbb{R}^n$ and a set $S \subseteq \{1, \dots, n\}$, we set $\mathbf{x}(S) = \sum_{i \in S} x_i$.

Voting. An *election* $E = (C, V, \mathcal{P})$ is given by a set $C = \{c_1, \dots, c_m\}$ of *candidates*, a set $V = \{1, \dots, n\}$ of *voters*, and a *preference profile* $\mathcal{P} = (P_1, \dots, P_n)$, where each P_i, $i \in V$, is a linear order over C. The order P_i represents the preferences of the i-th voter; for readability, we sometimes write $\succ_i$ instead of P_i. We denote the set of all linear orders over C by $L(C)$; thus, for any election $E = (C, V, \mathcal{P})$ with $|V| = n$ we have $\mathcal{P} \in L(C)^n$. For any $U \subseteq V$, we write $\mathcal{P}_U = (P_i)_{i \in U}$ and $\mathcal{P}_{-U} = (P_i)_{i \notin U}$; we have $\mathcal{P} = (\mathcal{P}_U, \mathcal{P}_{-U})$.

A *voting rule* $\mathcal{R}$ is a mapping that given an election $E = (C, V, \mathcal{P})$ outputs a candidate $c = \mathcal{R}(E)$, which is called the *winner* of E. When the sets C and V are clear from the context, we will sometimes omit them from the notation and write $\mathcal{R}(\mathcal{P})$ instead of $\mathcal{R}(E)$. Note that we require that each election has a unique winner. Many classic voting rules are, in fact, voting correspondences, i.e., they may output multiple winners. We assume that whenever this is the case, the resulting tie is broken lexicographically. We restrict our attention to voting rules with a poly-time winner determination algorithm.

Manipulation and Bribery. Two well-studied forms of dishonest behavior in elections are *manipulation*, i.e., cheating by voters, and *bribery*, i.e., cheating by an external party that wants to influence the outcome of the election. Below, we define the variants of these problems that are relevant to our work, namely, *coalitional manipulation* and *priced bribery*.

Definition 2.1 ([Conitzer *et al.*2007]). *For a voting rule $\mathcal{R}$, an instance $I = (E, S, c)$ of $\mathcal{R}$-*COALITIONAL MANIPULATION *problem is given by an election $E = (C, V, \mathcal{P})$,*

a set of manipulators S, $S \cap V = \emptyset$, *and the manipulators' preferred candidate* $c \in C$. *It is a "yes"-instance if there is a vector* $\mathcal{P}_S = (P_i)_{i \in S} \in (L(C))^{|S|}$ *such that* $\mathcal{R}(\mathcal{P}, \mathcal{P}_S) = c$; *otherwise, it is a "no"-instance.*

Observe that in the traditional definition of coalitional manipulation the manipulators, unlike honest voters, do not have preferences over the candidates: they simply want to get a particular candidate elected. This definition is convenient because it eliminates the problem of deciding which candidates the manipulators should support.

Definition 2.2 ([Faliszewski *et al.*2009]). *For a voting rule* $\mathcal{R}$, *an instance* $I = (E, \mathbf{b}, B, c)$ *of* $\mathcal{R}$-$BRIBERY *problem is given by an election* $E = (C, V, \mathcal{P})$ *with* $|V| = n$, *a vector of prices* $\mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{N}^n$, *a budget* $B \in \mathbb{N}$, *and the briber's preferred candidate* $c \in C$. *It is a "yes"-instance if there is a vector* $\mathcal{P}' = (P'_1, \ldots, P'_n)$ *over* C *and a set of voters* S *such that* $P_i = P'_i$ *for* $i \notin S$, $\mathcal{R}(\mathcal{P}') = c$, *and* $\mathbf{b}(S) \leq B$; *otherwise, it is a "no"-instance.*

We will also consider settings with weighted voters, where each voter $i \in V$ has a non-negative integer weight w_i; we denote the weight vector by $\mathbf{w} = (w_1, \ldots, w_n)$. To apply a voting rule $\mathcal{R}$ to a weighted election, we replace each voter i with w_i voters whose preferences are identical to those of i. The definitions of coalitional manipulation and $bribery can be adapted to this setting in a straightforward manner; in particular, when a voter of weight w_i is bribed or participates in a manipulation, we require that all w_i "copies" of this voter vote in the same way.

Computational Complexity. We assume familiarity with basic notions of computational complexity, such as polynomial-time algorithms and classes NP and coNP. A somewhat less standard notion is that of *strong* NP-hardness: a problem is said to be strongly NP-hard if it remains NP-hard even if all numbers in the input (such as, e.g., bribery prices) are given in unary. A related notion is that of a *pseudopolynomial* algorithm: an algorithm is said to be pseudopolynomial if its running time is polynomial in the numeric value of the input.

Coalitional Games. Coalitional games model settings where players form coalitions and derive benefits from collaboration. We assume *transferable utility* model, that is, the members of a coalition can freely distribute the benefits they obtain by working together. It is convenient to think of these benefits as monetary. Formally, a coalitional game $G = (N, v)$ is given by a set of *players* $N = \{1, \ldots, |N|\}$ and a characteristic function $v : 2^N \to \mathbb{R}^+ \cup \{0\}$, which for each *coalition* of players $S \subseteq N$ outputs the total amount of money that the players in S can earn by working together. It is standard to normalize the characteristic function by requiring $v(\emptyset) = 0$. A game is called *monotone* if $v(S) \leq v(T)$ for any $S, T \subseteq N$ such that $S \subseteq T$. A player $i \in N$ is called a *dummy* if $v(S) = v(S \setminus \{i\})$ for all $S \subseteq N$.

An *outcome* of a game G is a vector $\mathbf{x} = (x_1, \ldots, x_{|N|})$ that satisfies $x_i \geq 0$ for all $i \in N$ and $\mathbf{x}(N) = v(N)$. An outcome $\mathbf{x}$ is said to be *stable* if $\mathbf{x}(S) \geq v(S)$ for any $S \subseteq N$; the set of all stable outcomes of a game is called the *core*.

Another useful solution concept is that of Shapley value, which measures players' average marginal contributions in the game. Given a set of players N, by $\Pi(N)$ we mean the set of all permutations of N and for a permutation $\pi \in \Pi(N)$ we write N_i^π

to mean the set of players preceding i with respect to permutation π (not including i). Shapley value of player i in game $G = (N, v)$ is defined as

$$\phi_i(G) = \frac{1}{|N|!} \sum_{\pi \in \Pi(N)} \left(v(N_i^\pi \cup \{i\}) - v(N_i^\pi)\right).$$

3 Voting Manipulation Games

We consider the scenario where in a given election $E = (C, V, \mathcal{P})$ a subset of voters $M \subseteq V$ have an established communication channel and can agree to act jointly if this can be beneficial for all of them. The two most important issues here are (1) whether the players in a group have a course of action that is more beneficial for them than truthful voting, and (2) whether the players can agree on such a course of action so that no subgroup of players can benefit by deviating from it.

To formally model this scenario, we need to define what actions are considered feasible for a coalition and how the players outside of the coalition are expected to behave.

Definition 3.1. *Given an election $E = (C, V, \mathcal{P})$, a set $M \subseteq V$ and a voting rule $\mathcal{R}$, we say that a candidate $c \in C$ is* feasible *for a coalition $S \subseteq M$ if there is a preference profile $\mathcal{P}'_S$ such that $\mathcal{R}(\mathcal{P}'_S, \mathcal{P}_{-S}) = c$. We denote the set of all candidates that are feasible for S by $F(S)$. When the voters in S vote according to a profile $\mathcal{P}'_S$ and $\mathcal{R}(\mathcal{P}'_S, \mathcal{P}_{-S}) = c$, we say that S* manipulates in favor of *c, or* supports *c.*

Note that the winner of E is feasible for any coalition $S \subseteq M$, i.e., we have $\mathcal{R}(E) \in F(S)$. Also, we emphasize that when the voters in S are trying to decide which candidates are feasible for them, they assume that all other voters (including the remaining voters in $M \setminus S$) vote truthfully. We believe that this assumption is appropriate for the following reasons. First, the issue that we are most interested in in this paper is the process of forming a manipulating coalition. We view this problem from the perspective of a voter that wants to initiate a manipulation. His primary concern is whether he can find partners who are willing to engage in a mutually beneficial collaboration with him. Once he has found such a group of like-minded voters, it is plausible that other potential manipulators—who were not invited to join the coalition—will not notice that a manipulating coalition has been formed, or will decide not to react, e.g., because, unless they coordinate among themselves, the consequences of such a reaction are uncertain. One could, of course, posit that the remaining potential manipulators will respond by forming one or more manipulating coalitions among themselves, and try to counteract the actions of the original manipulator. However, to study the resulting model, one needs to resort to non-cooperative game theory, and non-cooperative game theory models of voting appear to be hard to analyze in all but a handful of settings (see, e.g., [Desmedt and Elkind2010, Xia and Conitzer2010, Meir *et al.*2010]). An adversarial model, where the players in a coalition assume the worst about the actions of other players, suffers from some difficulties of its own. Thus, we decided to employ the current model because it gives a good approximation of the issues we want to focus on in this paper.

We consider the case where colluders, i.e., members of the set M in Definition 3.1, have cardinal utilities for all candidates. and can make side payments to each other. Formally, any voter $i \in M$ has a *utility function* $u_i : C \to \mathbb{R}^+ \cup \{0\}$, which satisfies $u_i(c) \geq u_i(c')$ if and only if $c \succ_i c'$. This definition can be extended to coalitions by setting $u_S(c) = \sum_{i \in S} u_i(c)$ for any $S \subseteq M$ and any $c \in C$. Note that we allow agents to assign the same utility to two different candidates. Indeed, in many voting scenarios a voter may be indifferent between some of the candidates. While the voting rule usually requires voters to provide total orders, there is no need to impose such requirements on utility functions.

Under these assumptions, a coalition $S \subseteq M$ can benefit from manipulating in favor of a candidate $c \in C$ if and only if $u_S(c) > u_S(\mathcal{R}(E))$. Indeed, if this holds, the voters in S who prefer c to $\mathcal{R}(E)$ can compensate the other voters in S by making side payments to them. Thus, a manipulating coalition should aim to elect a feasible candidate that maximizes its total utility. Formally, for any $S \subseteq M$ we set

$$\mathrm{opt}(S) = \{c \in F(S) \mid u_S(c) \geq u_S(c') \text{ for all } c' \in F(S)\}.$$

Since we have $\mathcal{R}(E) \in F(S)$ for any $S \subseteq M$, it follows that $\mathrm{opt}(S) \neq \emptyset$ for any $S \subseteq M$. In what follows, we assume that if $|\mathrm{opt}(S)| > 1$, then the manipulators in S agree on a unique alternative in S using some commonly known tie-breaking rule; therefore, abusing notation, we will treat $\mathrm{opt}(S)$ as an element of C (rather than as an element of 2^C).

We are now ready to define the (transferable utility) coalitional game that can be associated with this setting.

Definition 3.2 (Voting Manipulation Game). *Given an election $E = (C, V, \mathcal{P})$, a set $M \subseteq V$, a vector $\mathbf{u} = (u_i)_{i \in M}$ of utility functions and a voting rule $\mathcal{R}$, a* voting manipulation game *$\mathcal{R}$-$G_{E,M,\mathbf{u}}$ is a coalitional game with a set of players M and a characteristic function v given by $v(S) = u_S(\mathrm{opt}(S)) - u_S(\mathcal{R}(E))$ for any $S \subseteq M$.*

For weighted voters, the description of the game needs to be augmented with a weight vector $\mathbf{w} = (w_1, \ldots, w_{|V|})$; we denote the resulting game by $\mathcal{R}$-$G_{E,M,\mathbf{u},\mathbf{w}}$.

Informally, the value of a coalition S is the maximum joint improvement over the *status quo* that the member of S can achieve, assuming that all other voters vote truthfully. Note that we do not normalize the utility functions. Indeed, some voters may be essentially indifferent to the election outcome, whereas others have strong preferences over outcomes. For computational reasons, we rescale all utilities so that they are nonnegative integers.

4 Computing Coalition Values

As argued above, we always have $\mathcal{R}(E) \in F(S)$, and therefore $v(S) \geq 0$ for any $S \subseteq M$. However, a voting manipulation game is not necessarily monotone. For example, it may happen that $\mathrm{opt}(S) = \mathrm{opt}(S \cup \{i\}) = c$ for some $i \in M \setminus S$, but $\mathcal{R}(E) \succ_i c$. That is, the new voter i does not share the coalition's goal but is too insignificant to affect the action chosen by the coalition. Of course, this does not mean

that i is unwilling to take part in the manipulation: the monetary transfer he gets from other manipulators induces him to participate. However, the remaining players in S may be unwilling to accept him: indeed, they can manipulate in favor of c even if i does not join, and they would have to make transfers to i to keep him happy. Thus, the grand coalition M does not necessarily have a higher value than its proper subsets. Therefore, it is natural to ask if we can identify coalitions with the highest value. An even more basic question is whether we can compute the value of a given coalition. It turns out that the complexity of these questions is closely related to the complexity of, respectively, \$bribery and coalitional manipulation for the underlying voting rule.

Theorem 4.1. *Let $\mathcal{R}$ be a voting rule. There exists a poly-time algorithm for computing the characteristic function of the voting manipulation game $\mathcal{R}$-$G_{E,M,\mathbf{u}}$ if and only if $\mathcal{R}$-*COALITIONAL MANIPULATION *is poly-time solvable.*

Proof sketch. For the "if" direction, given a coalition S, we check, for each $c \in C$, if S can make c the winner, and choose the best feasible candidate. For the "only if" direction, we set $u_i(c) = 1$, $u_i(x) = 0$ for $x \in C \setminus \{c\}$ for all $i \in M$, where c is the manipulator's preferred candidate. □

Theorem 4.2. *Let $\mathcal{R}$ be a voting rule. If $\mathcal{R}$-*\$BRIBERY *is poly-time solvable, then there exists a poly-time algorithm that given a voting manipulation game $\mathcal{R}$-$G_{E,M,\mathbf{u}}$ computes a coalition S such that $v(S) \geq v(T)$ for any $T \subseteq M$.*

Proof sketch. Given a voting manipulation game $\mathcal{R}$-$G_{E,M,\mathbf{u}}$ with $E = (C, V, \mathcal{P})$, for each $c \in C$ we construct an instance $I_c = (E, \mathbf{b}^c, B^c, c)$ of $\mathcal{R}$-\$BRIBERY as follows. We set $w = \mathcal{R}(E)$, $U = \max\{u_i(a) \mid i \in M, a \in C\}$. For each $i \in V$, we set $b_i^c = (n+1)U$. Further, we set $M_c = \{i \in M \mid u_i(c) > u_i(w)\}$ and for each $i \in M_c$ we set $b_i^c = 0$. Finally, for each $i \in Q_c = M \setminus M_c$ we set $b_i^c = u_i(w) - u_i(c)$, and $B^c = \sum_{i \in M_c}(u_i(c) - u_i(w))$.

If I_c is a "no"-instance of $\mathcal{R}$-\$BRIBERY, we discard this value of c. Otherwise, we use binary search to identify the smallest value $\widehat{B}^c$ such that $\widehat{I}_c = (E, \mathbf{b}^c, \widehat{B}^c, c)$ is still a "yes"-instance of $\mathcal{R}$-\$BRIBERY. Finally, we pick the candidate c that corresponds to the maximum value of $u_{M_c}(c) - u_{M_c}(w) - \widehat{B}^c$, over all non-discarded candidates, and let S be the coalition that consists of all voters in M_c together with all voters that receive non-zero bribes in $\widehat{I}_c$. Observe that we have $v(S) \geq u_{M_c}(c) - u_{M_c}(w) - \widehat{B}^c$.

Clearly, our algorithm runs in polynomial time. To see that S is a coalition with the maximum value of the characteristic function, observe that our bribery instances can be interpreted as follows: the colluders that benefit from getting c elected pool their profits from making c the winner and use them to bribe other colluders; the cheapest successful bribery corresponds to a coalition that minimizes the disutility of the colluders who prefer w to c, and therefore maximizes the total utility, among all coalitions that manipulate in favor of c. We omit the formal proof due to space constraints. □

5 Manipulating in Favor of a Given Candidate

From a candidate's perspective, a natural question is whether there exists a coalition that is willing to manipulate in her favor. One might think that the answer to this

question is given by the proof of Theorem 4.2: indeed, in this proof we determine, for each candidate c, if there is a coalition that can profit from manipulating in favor of c. However, this does not necessarily answer the question above: it may happen that any coalition that *can* manipulate in favor of c would in fact prefer to manipulate in favor of some other candidate a. Indeed, it turns out that finding a coalition S such that $\mathrm{opt}(S) = c$ for a given candidate c is hard even for Plurality, and even if the number of candidates is bounded by a small constant.

Theorem 5.1. *Given a voting manipulation game* Plurality-$G_{E,M,\mathbf{u}}$ *with* $E = (C, V, \mathcal{P})$ *and a candidate* $c \in C$*, it is* NP*-complete to decide if there exists a set* $S \subseteq M$ *such that* $\mathrm{opt}(S) = c$*. The hardness result holds even if* $|C| = 5$*.*

The proof of Theorem 5.1 proceeds by a reduction from the classic PARTITION problem, and uses the fact that the players' utilities are given in binary. However, if the number of candidates is non-constant, finding a coalition that manipulates in favor of a given candidate is hard even if all utilities are given in unary.

Theorem 5.2. *Given a voting manipulation game* Plurality-$G_{E,M,\mathbf{u}}$ *with* $E = (C, V, \mathcal{P})$ *and a candidate* $c \in C$*, deciding whether there exists a set* $S \subseteq M$ *such that* $\mathrm{opt}(S) = c$ *is strongly* NP*-complete.*

Under Plurality—as well as under many other rules—if both the number of candidates is bounded by a constant and the utilities are given in unary then finding a coalition that is willing to manipulate in favor of a particular candidate (or, determining if one exists) is easy. We postpone a formal statement of this fact till the next section, as it is closely related to the results presented there.

6 Computing Players' Power

We now explore the role of individual players in voting manipulation games. We first consider the complexity of determining whether a player is a dummy. It turns out that this problem is hard even for Plurality if the number of candidates is constant, or if the utilities are given in unary (but not both). The following two results follow from the proofs of Theorems 5.1 and 5.2, respectively.

Theorem 6.1. *Given a voting manipulation game* Plurality-$G_{E,M,\mathbf{u}}$ *with* $E = (C, V, \mathcal{P})$ *and a player* $j \in M$*, it is* coNP*-complete to decide whether* j *is a dummy in* Plurality-$G_{E,M,\mathbf{u}}$*. The hardness result holds even if* $|C| = 5$*.*

Theorem 6.2. *Given a voting manipulation game* Plurality-$G_{E,M,\mathbf{u}}$ *with* $E = (C, V, \mathcal{P})$ *and a utility vector* $(u_i)_{i \in M}$*, and a player* $j \in M$*, it is strongly* coNP*-complete to decide whether* j *is a dummy in* Plurality-$G_{E,M,\mathbf{u}}$*.*

However, for a constant number of candidates we can check if a player is a dummy in pseudopolynomial time. Moreover, we can extend this result to the problem of computing a player's Shapley value (observe that since our game is not monotone, a player may have Shapley value of 0 without being a dummy).

Theorem 6.3. *Given a voting manipulation game $\mathcal{R}$-$G_{E,M,\mathbf{u}}$ with $E = (C, V, \mathcal{P})$ and a player $i \in M$, we can test if i is a dummy and compute i's Shapley value in pseudopolynomial time as long as $|C|$ is bounded by a constant.*

This algorithm can be adapted to check if there exists a coalition that supports a given candidate.

Corollary 6.4. *Given a voting manipulation game $\mathcal{R}$-$G_{E,M,\mathbf{u}}$ with $E = (C, V, \mathcal{P})$ and a candidate $c \in C$, we can test if there exists a coalition S such that $\mathrm{opt}(S) = c$ and if so, to compute this coalition in pseudopolynomial time as long as $|C|$ is bounded by a constant.*

We remark that if a player is a dummy in a voting manipulation game, it does not mean that he does not influence the outcome of the election. Indeed, by adding a player to a coalition we can change the identity of the candidate promoted by this coalition, without changing its total payoff.

Example 6.5. Suppose that our voting rule is Plurality combined with the lexicographic tie-breaking rule, $C = \{a, b, c\}$, the honest voters grant 1 point to a, 2 points to b, and 2 points to c, $M = \{1, 2\}$. Suppose that $u_1(a) = 3$, $u_1(b) = 2$, $u_1(c) = 0$ and $u_2(c) = 2$, $u_2(a) = 1$, $u_2(b) = 0$, and hence $a \succ_1 b \succ_1 c$, $c \succ_2 a \succ_2 b$. Under truthful voting, c wins. On her own, player 1 cannot change the election outcome to a, but she can change it to b, so we have $v(\{1\}) = u_1(b) - u_1(c) = 2$. On the other hand, 1 and 2 together can change the outcome to a. However, since 2 prefers c to a, he would have to be compensated. Indeed, we have $u_M(a) = 4$, $u_M(b) = 2$, $u_M(c) = 2$, so $\mathrm{opt}(\{1, 2\}) = a$ and $v(\{1, 2\}) = 2 = v(\{1\})$. Also, it is clear that $v(\{2\}) = 0$, since 2 does not want to change the election outcome. Thus, player 2 is a dummy in our voting manipulation game, yet when he joins a coalition, the coalition changes its behavior.

Conversely, a player can change the value of a coalition without changing the candidate that this coalition supports.

Example 6.6. Consider again Plurality with lexicographic tie-breaking and $C = \{a, b, c\}$. Suppose there are 10 honest voters who vote for a and 8 honest voters who vote for c, as well as four manipulators $\{1, 2, 3, 4\}$ who strictly prefer b to c to a. Set $S = \{1, 2, 3\}$. We have $\mathrm{opt}(S) = c$, $\mathrm{opt}(S \cup \{4\}) = c$, and hence $v(S \cup \{4\}) = v(S) + u_4(c) - u_4(a) > v(S)$, i.e., 4 is not a dummy. However, 4 does not have to change his vote when he joins the manipulating coalition, and neither do the voters in S. That is, 4 simply free-rides on S.

These two examples motivate the following definition.

Definition 6.7. *We say that a player i is* powerless *in a voting manipulation game $\mathcal{R}$-$G_{E,M,\mathbf{u}}$ if for any $S \subseteq M \setminus \{i\}$ we have $\mathrm{opt}(S) = \mathrm{opt}(S \cup \{i\})$.*

Intuitively, a player is powerless if whenever he joins a coalition neither himself nor the players already in the coalition can benefit from changing their vote. The discussion above illustrates that a player can be a dummy without being powerless (Example 6.5) and vice versa (Example 6.6). However, it turns out that checking whether a player is powerless has the same complexity as checking whether it is a dummy.

Corollary 6.8. *Given a voting manipulation game* Plurality-$G_{E,M,\mathbf{u}}$ *with* $E = (C, V, \mathcal{P})$*, it is* coNP*-complete to decide if a player is powerless. This holds even if* $|C| = 5$ *or if all utilities are given in unary. However, this problem is pseudopolynomial time-solvable if* $|C|$ *is bounded by a constant.*

7 Coalitional Stability

From game-theoretic perspective, a very important question is whether the collaboration among the colluders can be sustained, i.e., whether the core of the corresponding voting manipulation game is non-empty. We will now show that this problem is easy whenever $bribery is easy.

Theorem 7.1. *If* $\mathcal{R}$-$\textsc{bribery}$ *is in* P*, then there exists a poly-time algorithm that given a game* $\mathcal{R}$-$G_{E,M,\mathbf{u}}$ *with* $E = (C, V, \mathcal{P})$ *and a vector* $\mathbf{x}$ *decides whether* $\mathbf{x}$ *is in the core of* $\mathcal{R}$-$G_{E,M,\mathbf{u}}$*.*

Moreover, whenever $\mathcal{R}$-$\textsc{bribery} is in P, we can check if the core of $\mathcal{R}$-$G_{E,M,\mathbf{u}}$ is non-empty by formulating the core nonemptiness problem as a linear program and using the algorithm from Theorem 7.1 as a separation oracle (see, e.g., [Elkind *et al.*2009] for an exposition of this technique).

It is not clear if the converse of Theorem 7.1 also holds. However, we will now present a construction that allows us to reduce a wide class of $\mathcal{R}$-$\textsc{bribery} instances to testing nonmembership of an imputation $\mathbf{x}$ in the core of an $\mathcal{R}$-$G_{E,M,\mathbf{u}}$ voting game. Whenever the bribery problem is NP-hard, our construction may be used to prove NP-hardness of testing core nonmembership.

We start with an $\mathcal{R}$-$\textsc{bribery} instance $I = (E, \mathbf{b}, B, c)$, where $E = (C, V, \mathcal{P})$ and where we assume the following:

(a) At least one voter has bribery cost 0.

(b) There are at least two candidates and $w = \mathcal{R}(E) \neq c$.

(c) The sum of the bribery prices is greater than B.

We form a voting manipulation game $\mathcal{R}$-$G_{E,M,\mathbf{u}}$, where $M = V$ (we rename voters so that $M = \{1, \ldots, n+1\}$ and so that the bribery price of voter $n+1$ is 0). We set $\mathbf{u}$ and the imputation $\mathbf{x}$ as follows. For each voter $i \in M \setminus \{n+1\}$ we set $u_i(w) = b_i$, $x_i = 0$, and $u_i(d) = 0$ for each candidate $d \in C \setminus \{w\}$. Also, we set $u_{n+1}(c) = B+1$, $x_{n+1} = 0$ and $u_{n+1}(d) = 0$ for each $d \in C \setminus \{c\}$. We see that under truthful voting $u_M(w) = \mathbf{b}(M)$ and that $v(M) = 0$ (recall that by our assumption, $\mathbf{b}(M) > B$). We also see that $\mathbf{x}$ is unstable if and only if there is a coalition S such that $c \in F(S)$ and $u_S \leq B$. Such a coalition exists if and only if our input $\mathcal{R}$-$\textsc{bribery} instance is a "yes"-instance.

The above construction is particularly useful if $\mathcal{R}$-$\textsc{bribery} is NP-hard, and the proof of its NP-hardness can easily be adapted to output instances that satisfy our requirements. In particular, our requirements are satisfied if the NP-hardness of $\mathcal{R}$-$\textsc{bribery} is derived by combining Theorem 4.6 of [Faliszewski *et al.*2009] (a general

reduction from the coalitional manipulation problem to the $bribery problem) and the fact that $\mathcal{R}$-COALITIONAL MANIPULATION is NP-hard (even if there are at least two truthful voters). Thus, we have the following corollary.

Corollary 7.2. *Suppose that $\mathcal{R}$-*COALITIONAL MANIPULATION *is* NP*-hard even if there are at least two nonmanipulators. Then given a game $\mathcal{R}$-$G_{E,M,\mathbf{u}}$ and an imputation $\mathbf{x}$ it is* NP*-hard to decide if $\mathbf{x}$ is not in the core of $\mathcal{R}$-$G_{E,M,\mathbf{u}}$.*

We stress that our construction is more general and can be used, e.g., if either we do not have a complexity result for coalitional manipulation but we do have one for $bribery, or when coalitional manipulation is easy yet $bribery is NP-hard. As an example, we show that testing core nonmembership for an imputation is NP-hard for weighted Plurality.

Theorem 7.3. *Given a game Plurality-$G_{E,M\mathbf{u},\mathbf{w}}$ with $E = (C, V, \mathcal{P}, \mathbf{w})$ and a vector $\mathbf{x}$, it is* NP*-hard to check whether $\mathbf{x}$ is not in the core of Plurality-$G_{E,M\mathbf{u},\mathbf{w}}$.*

8 Conclusion

We have proposed a model for collusion in voting settings that takes into account the process of forming the manipulative coalition. Our model is based on cooperative game theory and predicts which coalitions and agreements are likely to occur in such settings. In addition, our research shows that computational problems previously studied in the context of voting manipulation, COALITIONAL MANIPULATION and $BRIBERY, which are non-game-theoretic in nature, nevertheless constitute very important building blocks in the cooperative game-theoretic study of election manipulation.

Several questions remain open for future research. First, a key assumption of our model is that agents have comparable utilities (given in a common currency) and that they can make monetary transfers.What happens when monetary transfers are not allowed? Second, regarding solution concepts, we focused on the core and the Shapley value, but other interesting solutions concepts, such as, e.g., the ε-core or the nucleolus, remain to be studied. Finally, it would be interesting to examine the relation between our model and noncooperative models for voting domains, using solution concepts such as strong Nash equilibrium.

References

[Conitzer *et al.*2007] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM*, 54(3):Article 14, 2007.

[Desmedt and Elkind2010] Y. Desmedt and E. Elkind. Equilibria of plurality voting with abstentions. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, pages 347–356. ACM Press, June 2010.

[Elkind *et al.*2009] E. Elkind, L. Goldberg, P. Goldberg, and M. Wooldridge. On the computational complexity of weighted voting games. *Annals of Mathematics and Artificial Intelligence*, 56(2):109–131, 2009.

[Faliszewski and Procaccia2010] P. Faliszewski and A. Procaccia. AI's war on manipulation: Are we winning? *AI Magazine*, 31(4):53–64, 2010.

[Faliszewski *et al.*2009] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. How hard is bribery in elections? *Journal of Artificial Intelligence Research*, 35:485–532, 2009.

[Faliszewski *et al.*2010] P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Manipulation of Copeland elections. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 367–374. International Foundation for Autonomous Agents and Multiagent Systems, May 2010.

[Gibbard1973] A. Gibbard. Manipulation of voting schemes. *Econometrica*, 41(4):587–601, 1973.

[Hazon and Elkind2010] N. Hazon and E. Elkind. Complexity of safe strategic voting. In *Proceedings of SAGT'10*, pages 210–221, 2010.

[Hemaspaandra and Hemaspaandra2007] E. Hemaspaandra and L. Hemaspaandra. Dichotomy for voting systems. *Journal of Computer and System Sciences*, 73(1):73–83, 2007.

[Meir *et al.*2010] R. Meir, M. Polukarov, J. Rosenschein, and N. Jennings. Convergence to equilibria in plurality voting. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 823–828, July 2010.

[Satterthwaite1975] M. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–217, 1975.

[Slinko and White2008] A. Slinko and S. White. Non-dictatorial social choice rules are safely manipulable. In *Proceedings of COMSOC'08*, pages 403–414, 2008.

[Walsh2009] T. Walsh. Where are the really hard manipulation problems? The phase transition in manipulating the Veto rule. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 324–329. AAAI Press, July 2009.

[Xia and Conitzer2010] L. Xia and V. Conitzer. Stackelberg voting games: Computational aspects and paradoxes. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 921–926, July 2010.

[Xia *et al.*2009] L. Xia, V. Conitzer, A. Procaccia, and J. Rosenschein. Complexity of unweighted manipulation under some common voting rules. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 348–353. AAAI Press, July 2009.

[Xia *et al.*2010] L. Xia, V. Conitzer, and A. Procaccia. A scheduling approach to coalitional manipulation. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, pages 275–284. ACM Press, June 2010.

The Shapley Value as a Function of the Quota in Weighted Voting Games

Yair Zick, Alexander Skopalik, and Edith Elkind

School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore

Abstract. In weighted voting games, each agent has a weight, and a coalition of players is deemed to be winning if its weight meets or exceeds the given quota. An agent's power in such games is usually measured by her Shapley value, which depends both on the agent's weight and the quota. [20] show that one can alter a player's power significantly by modifying the quota, and investigate some of the related algorithmic issues. In this paper, we answer a number of questions that were left open by [20]: we show that, while deciding whether a quota maximizes or minimizes an agent's Shapley value is coNP-hard, finding a quota that maximizes a player's Shapley value is easy. Minimizing a player's power appears to be more difficult. However, we propose and evaluate a heuristic for this problem, which takes into account the voter's ranking and the overall weight distribution. We also explore a number of other algorithmic issues related to quota manipulation.

1 Introduction

Collective decision making is a crucial component of multi-agent interaction. Consequently, assessing the power of individual voters in decision-making bodies is an important concern in the analysis of multi-agent systems. This issue is often studied within the framework of weighted voting games, where each player is associated with a weight; to win, a coalition needs to amass a weight that meets or exceeds a given threshold, or quota. Usually, the voter's power in such games is associated with her Shapley value [18], which in the context of weighted voting games is also known as the Shapley–Shubik power index [19]. This quantity depends on both the players' weights and the quota of the game.

Now, in a weighted voting game, each voter's weight is determined either by his contribution to the system (money, shares, etc.) or the size of the electorate that he represents. In either case, the voters' weights are usually hard to alter. In contrast, the quota of the game can easily be modified by the entity in charge of the decision-making process: for instance, a legislative body may raise the quota for decisions on certain issues from 51% of all votes to 66%. However, changing the quota can have a profound effect on players' power. This phenomenon has been observed in real-life voting systems [11–13], and recently [20] embarked on a systematic study of this issue from the algorithmic perspective. Specifically, [20] show that one can determine in polynomial time if a player's power can be

reduced to 0 by changing the quota; however, deciding which of the two given values of the quota is preferable for a given player is computationally hard.

In this paper, we continue to study the dependence between the players' power and the quota in weighted voting games. We focus on finding values of the quota that maximize/minimize the power of a given player. This is perhaps the most important problem from the perspective of a manipulator who cares about the impact of a certain agent in a decision-making body; however, it has not been addressed by the previous work.

First, we show that if arbitrary values of the quota are allowed, a player's power can be maximized by setting the quota to that player's weight. In contrast, the associated decision problem, i.e., determining whether the current value of the quota is already optimal for a given player, is computationally hard. Thus, if the manipulation is costly, it is hard for the manipulator to determine whether it is worth the effort.

If the goal is to minimize the player's power rather than to maximize it, then the respective decision problem remains hard, but the status of the optimization problem (finding a value of the quota that minimizes the player's power) is unclear. However, we identify two values of the quota that are very likely to be good choices. The first of them is $q = 1$ (assuming integer weights): when the quota is small enough, all players have the same power, which is likely to be a bad deal for larger players. The second candidate is $q = w + 1$, where w is the target player's weight. This quota is more likely to be harmful for smaller players. We perform empirical analysis, drawing the players' weights from uniform, normal, and Poisson distributions, and show that with high probability one of these values of the quota minimizes the target player's power, with $q = w + 1$ usually being the right choice for the smaller players and $q = 1$ being the right choice for the larger players. We provide a (partial) analytic explanation of these results, by showing that for the bottom half of the voters (with respect to the weight) the quota $q = w + 1$ is strictly worse than $q = 1$.

While it is hard to determine whether a given value of the quota is optimal/pessimal for a given player, there are interesting special cases of this problem that admit an efficient algorithm, namely, checking if a given quota maximizes the power of the smallest player or minimizes the power of the largest player. Both questions can be reduced to deciding whether all players are equally powerful, and this problem turns out to be polynomial-time solvable.

The rest of the paper is structured as follows. We give a brief overview of related work in Section 1.1. Section 2 introduces the necessary terminology. Section 3 provides several examples that illustrate the behavior of the Shapley value as a function of the quota. Section 4 details the main theoretical results of our work, and Section 5 complements them by empirical analysis. Section 6 presents our conclusions and suggests directions for future research.

1.1 Related Work

We have already mentioned several papers that are relevant to the topic of our research, with [20] being the direct precursor of this work. In this section we survey a number of papers on closely related topics.

The complexity of computing the Shapley–Shubik index is well understood: [4, 15, 17] show that even deciding whether a player has zero power is hard (and hence computing the exact value of the index is hard, too). We remark that these hardness results do not preclude the existence of efficient algorithms for manipulating the quota: it might be possible to change a player's power in the desired direction even without knowing the exact value of his power before and after the change. Further, there are a number of heuristic and approximation algorithms for power computation [14, 10, 5, 2, 7, 16]. [1] study a different form of manipulation in weighted voting games, namely, players splitting their weight among several identities, or, conversely, merging into a single identity. [6] consider the complexity of comparing the players' power across different weighted voting games. An alternative approach to measuring a player's power is by means of the Banzhaf power index [3]. The behavior of this index as a function of the quota has been studied in [5, 10, 16]; the results of this analysis have been used in developing approximation algorithms for this index [7].

2 Preliminaries

A *weighted voting game* $G = (\mathbf{w}, q)$ is given by a vector $\mathbf{w} = (w_1, \ldots, w_n)$ of non-negative integer *weights* and a non-negative integer *quota* $q \in \mathbb{Z}_+$. It is associated with a set of *players* $N = \{1, ..., n\}$, where the i-th player has weight w_i. We order the players so that $w_1 \leq w_2 \leq \ldots \leq w_n$. A subset, or *coalition*, $S \subseteq N$ is called *winning* if $w(S) := \sum_{j \in S} w_j \geq q$, and *losing* otherwise. We write $v(S) = 1$ if S is winning and $v(S) = 0$ if S is losing. It is usually stipulated that $v(N) = 1$, i.e., $q \leq w(N)$. A player i is called *q-pivotal* for $S \subseteq N \setminus \{i\}$ if $q - w_i \leq w(S) < q$, or, equivalently, if $v(S) = 0$, but $v(S \cup \{i\}) = 1$. When q is clear from the context, we will simply say that i is pivotal for S. A player i is called a *dummy* if he is not pivotal for any coalition, i.e, $v(S) = v(S \cup \{i\})$ for any $S \subseteq N$.

Let $\Pi(N)$ be the set of permutations over N, and let $P_i(\sigma) \subseteq N$ denote the set of all *predecessors* of player i in a permutation $\sigma \in \Pi(N)$, i.e., $P_i(\sigma) = \{j \in N \mid \sigma(j) < \sigma(i)\}$. We say that i is q-pivotal for σ if i is q-pivotal for $P_i(\sigma)$. The set of all permutations for which a player $i \in N$ is q-pivotal is denoted by $\Pi_i(q)$. The *Shapley value*, or *Shapley–Shubik power index*, [18, 19] of player i in a game with quota q is $\phi_i(q) = \frac{|\Pi_i(q)|}{n!}$. This power index has a number of very attractive properties; it is *efficient*, i.e., $\sum_{i=1}^{n} \phi_i(q) = 1$, *symmetric*, i.e., if $v(A \cup \{i\}) = v(A \cup \{j\})$ for all $A \subseteq N \setminus \{i, j\}$, then $\phi_i(q) = \phi_j(q)$, and *monotone*, i.e., $w_i \leq w_j$ implies $\phi_i(q) \leq \phi_j(q)$.

As we vary the quota q, $\phi_i(q)$ becomes a function from $\mathbb{N}$ to $[0, 1]$. Note, however, that we require $q \leq w(N)$ to ensure $v(N) = 1$. Also, if $q \leq 0$, all

coalitions are winning and hence by symmetry and efficiency we have $\phi_i(q) = 1/n$ for all $i = 1, \ldots, n$. Thus, we limit our analysis to the values of q in the interval $[1, w(N)] \cap \mathbb{N}$. Note also that there is no loss of generality is assuming $q \in \mathbb{N}$: while $\phi_i(q)$ is well-defined for any real $q \in [1, w(N)]$, all players' weights are integer, so a game $(\mathbf{w}, q)$ with $q \in \mathbb{R}$ is equivalent to the game $(\mathbf{w}, \lfloor q \rfloor)$. We set $opt(\phi_i) = \{q \in \mathbb{N} \mid \phi_i(q) \geq \phi_i(q') \text{ for all } q' \in \mathbb{N}\}$ and $pess(\phi_i) = \{q \in \mathbb{N} \mid \phi_i(q) \leq \phi_i(q') \text{ for all } q' \in \mathbb{N}\}$; these are the sets of quota values that, respectively, maximize and minimize the power of player i.

3 Examples

We start by providing several examples of weighted voting games, and investigate the behavior of a given player's power as a function of the quota in these games.

Example 1. We construct a 20-player game by drawing weights uniformly at random from $[1, 40]$; the resulting weight vector is $\mathbf{w}^1 = (1, 2, 4, 5, 16, 17, 20, 21, 21, 23, 24, 24, 27, 28, 28, 33, 33, 36, 36, 40)$. Figure 1 shows the Shapley value of player 10 with weight 23 in games of the form $(\mathbf{w}, q)$, where q varies from 1 to $w(N)$. We note several interesting properties of this graph. First, $\phi_i(q)$ is symmetric; this is a well-known property of the Shapley value, referred to as *self-duality* [8]. Second, the graph has two distinct peaks at $w_{10} = 23$ and $w(N) - w_{10} + 1 = 417$. This observation is in line with our theoretical results: Section 4.1 shows that $\phi_i(q)$ always peaks at $q = w_i$. Third, $\phi_{10}(q)$ has a global minimum at $q = 24 = w_{10} + 1$; Section 5 demonstrates that $q = w_i + 1$ is often (though not always) the worst possible value of the quota for player i. Finally, the graph plateaus at $w_{10}/w(N) \approx 0.052$ as the quota goes to $\frac{w(N)+1}{2}$; this phenomenon has been observed (and explained) in [12, 13].

Example 2. This time, we construct a 20-player game by drawing the players' weights from the Poisson distribution with mean 30, obtaining weight vector $\mathbf{w}^2 = (23, 24, 24, 25, 25, 25, 25, 27, 28, 28, 29, 30, 30, 32, 32, 33, 34, 34, 35, 36)$; we focus on the 2-nd largest player. In this case, we observe a high degree of fluctuation in the player's Shapley value.

Example 3. Finally, consider a weight vector of the form $1, 2, \ldots, 2^n$. The graphs for $n = 7$ and players with weights 4, 16 and 64 are given in Figure 3. A remarkable property of this set of weights is the abundance of local minima and maxima; ϕ_1 has a local maximum at any even quota and a minimum at any odd quota, and $\phi_4(q) = 0$ for $q = 16, 32, 48, \ldots$. This is true in general for weight vectors of this form.

Proposition 1. *If $q = 2^k r$ for some $r \in \mathbb{N}$, then $\phi_k(q) = 0$. Further, $\phi_k(q)$ has a local maximum at $q = 2^{k-1}(2r-1)$, for all $r \in \mathbb{N}$ such that $2^{k-1}(2r-1) \leq w(N)$.*

The intuition behind Proposition 1 is that for k to be $2^k r$-pivotal for a coalition S, it must be the case that $2^k r - 2^{k-1} \leq w(S) < 2^k r$. But then the $(k-1)$-st digit of $w(S)$ is set to 1, i.e., $k \in S$, a contradiction.

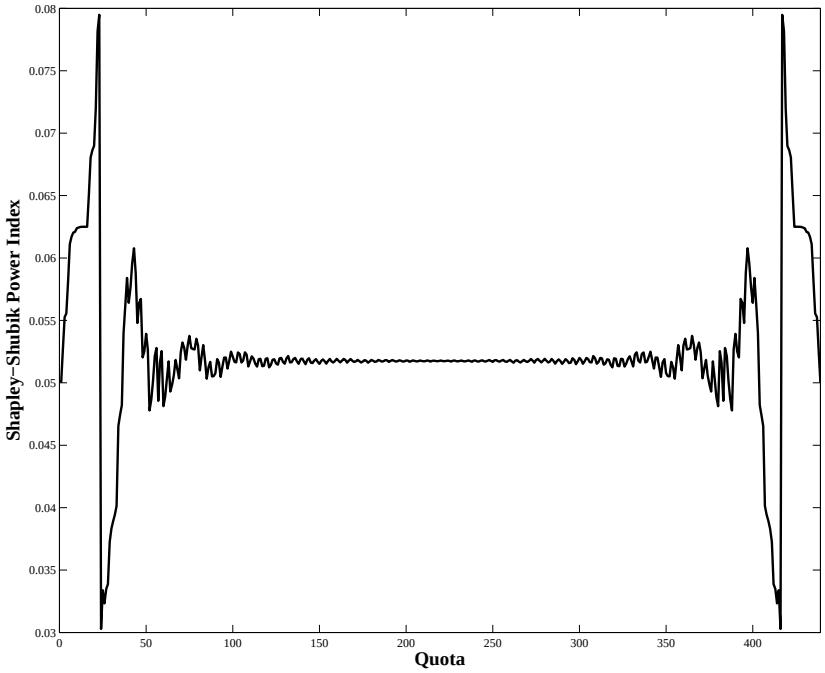


Fig. 1: The Shapley value of player 10 (weight 23) for weight vector $\mathbf{w}^1$ (Example 1)

Examples 2 and 3 show that $\phi_i(q)$ may be highly non-monotone: thus, if we are only allowed to change the quota within a given (small) interval, the best value of the quota is not necessarily at an endpoint of this interval.

4 Theoretical Results

In this section, we provide algorithms and hardness results for a number of problems related to maximizing or minimizing the power of a given player.

4.1 Maximizing the Shapley Value

Before we formally state the main result of this section, let us prove the following useful lemma. We define $T_i(x) = \{\sigma \in \Pi(N) \mid w(P_i(\sigma)) < x\}$ for all $x > 0$.

Lemma 1. $|T_i(a)| + |T_i(b)| \geq |T_i(a+b)|$ *for any* $a, b \in \mathbb{N}$.

Proof. Without loss of generality, we assume $a \geq b$. Set $T_i(a, a+b) = \{\sigma \in \Pi(N) \mid a \leq w(P_i(\sigma)) < a+b\}$; since $T_i(a) \subseteq T_i(a+b)$, we have $|T_i(a+b)| - |T_i(a)| = |T_i(a, a+b)|$. Thus, it suffices to show that $|T_i(b)| \geq |T_i(a, a+b)|$.

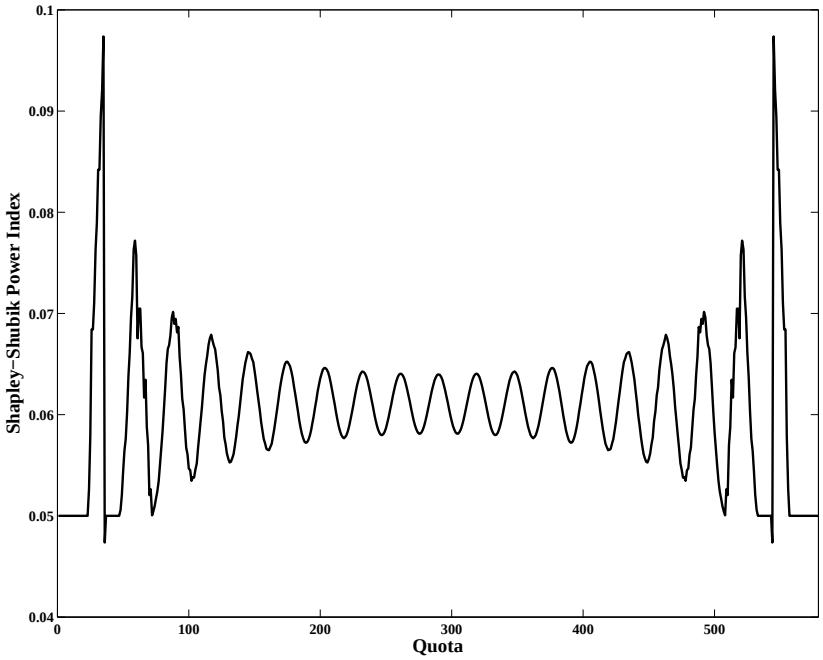


Fig. 2: The Shapley value of player 19 (weight 35) for weight vector $\mathbf{w}^2$ (Example 2)

We construct a injective mapping $\psi : T_i(a, a+b) \to T_i(b)$ as follows. If $\sigma \in T_i(a, a+b)$ is a permutation of the form $\sigma = (x_1, ..., x_k, y_1, ..., y_\ell, i, z_1, ..., z_r)$, where k is the first index for which $\sum_{j=1}^{k} w(x_j) \geq a$, then we set $\psi(\sigma) = (y_1, \ldots, y_\ell, i, x_1, \ldots, x_k, z_1, \ldots, z_r)$. Note that since i and a are given, ψ is invertible and hence injective. We denote $X = \{x_1, \ldots, x_k\}$ and $Y = \{y_1, \ldots, y_\ell\}$; it is possible that $Y = \emptyset$, but this does not affect our analysis. Since $\sigma \in T_i(a, a+b)$, we have $w(X \cup Y) < a + b$. However, $w(X) \geq a$, so $w(Y) < b$. This means that $\psi(\sigma) \in T_i(b)$. Thus, there exists an injective mapping from $T_i(a, a+b)$ to $T_i(b)$, and hence $|T_i(b)| \geq |T_i(a, a+b)|$.

Theorem 1. *For any weight vector* $\mathbf{w}$*, a quota of* w_i *maximizes the Shapley value of player* i*; that is,* $w_i \in opt(\phi_i)$*.*

Proof. We differentiate between the following two cases:

$q \leq w_i$**:** For any $\sigma \in \Pi_i(q)$, $w(P_i(\sigma)) < q \leq w_i$ and $w(P_i(\sigma)) + w_i \geq w_i$, hence $\sigma \in \Pi_i(w_i)$. Therefore, for all $q \leq w_i$ it holds that $\Pi_i(q) \subseteq \Pi_i(w_i)$, and hence $\phi_i(q) \leq \phi_i(w_i)$.

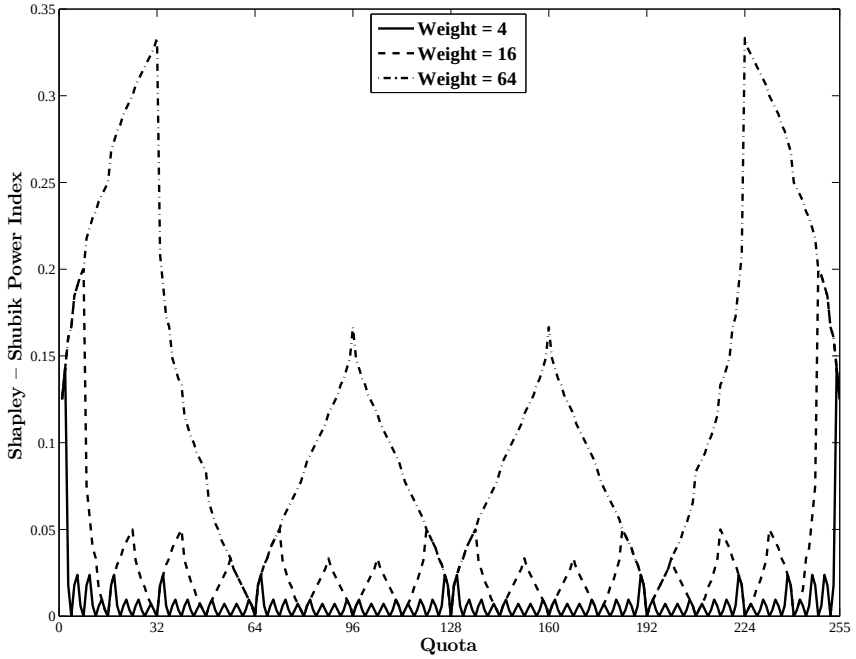


Fig. 3: The Shapley values of players $3, 5, 7$ (weights $4, 16, 64$) for weight vector $\mathbf{w}^3 = (1, 2, \ldots, 128)$ (Example 3)

$q > w_i$**:** Note that $\Pi_i(q) = T_i(q) \setminus T_i(q - w_i)$ and $\Pi_i(w_i) = T_i(w_i)$. By Lemma 1 we have $|T_i(w_i)| + |T_i(q - w_i)| \geq |T_i(q)|$. Thus, we obtain

$$\begin{aligned} |\Pi_i(w_i)| = |T_i(w_i)| &\geq |T_i(q)| - |T_i(q - w_i)| \\ &= |T_i(q) \setminus T_i(q - w_i)| = |\Pi_i(q)|, \end{aligned}$$

and hence $\phi_i(w_i) \geq \phi_i(q)$. □

Theorem 1 provides a simple recipe for the manipulator who favors player i; note also that by self-duality the manipulator may also use $q = w(N) - w_i + 1$. However, even if the manipulator has the ability to set the quota wherever it pleases, such changes might be costly. Thus, she might want to know whether the current quota is already optimal. Since we already know that $w_i \in opt(\phi_i)$, this is equivalent to asking whether $\phi_i(q) = \phi_i(w_i)$; we call this decision problem MaxSV. MaxSV can be viewed as a special case of the Quota problem considered in [20], where we are given $\mathbf{w}, i, q$, and q', and the goal is to check whether $\phi_i(q) > \phi_i(q')$. [20] prove that Quota is computationally hard; however, this does not imply that MaxSV is hard, since in MaxSV one of the candidate quotas is fixed to be w_i, which potentially could make MaxSV an easier problem.

Nevertheless, we can show that MaxSV is hard, too; the proof proceeds by a reduction from SubsetSum [9], and is omitted due to space constraints.

Theorem 2. MaxSV *is* coNP*-hard.*

4.2 Minimizing the Shapley Value

So far we focused on maximizing an agent's Shapley value. However, the manipulator may wish to *minimize* the power of a player by changing the quota. We first establish that, just as in the case of maximization, the corresponding decision problem is hard. Specifically, we define the problem MinSV as follows: given a weighted voting game $G = (\mathbf{w}, q)$, and a player $i \in N$, is it the case that $q \in pess(\phi_i)$? We have the following result (proof omitted).

Theorem 3. MinSV *is* coNP*-hard.*

For the rest of this section, we focus on *finding* a quota in $pess(\phi_i)$. This task appears to be more challenging than finding a maximizing quota. Indeed, the graphs in Section 3 suggest that a suitable value of the quota may be $q = w_i + 1$. However, our experiments (see Section 5) show that for many games $w_i + 1 \notin pess(\phi_i)$. In particular, it is often the case for relatively large players. For such players it is often a good solution to set $q = 1$: this ensures that these players are no more powerful than smaller players. Indeed, for the largest player, $q = 1$ is clearly the worst possible quota, since $\phi_n(q) \geq \frac{1}{n}$ for any $q \in [1, w(N)]$. This intuition is consistent with the empirical results of Section 5. However, this approach only works for above-median players: for below-median players $q = w_i + 1$ turns out to be strictly better than $q = 1$.

Theorem 4. *If* $i \leq \frac{n}{2}$ *and* $w_{i+1} > w_i$*, then* $\phi_i(w_i + 1) < \frac{1}{n}$.

Proof. If player i is pivotal for a set $S \subseteq N \setminus \{i\}$, and the quota is $w_i + 1$, then $S \subseteq \{1, \ldots, i-1\}$. Let us denote by A_k the sets of size k for which player i is pivotal. For any $1 \leq k \leq i-1$, we have $|A_k| \leq \binom{i-1}{k}$. Note also that the contribution of a set of size k to the Shapley value of player i equals to $\frac{k!(n-k-1)!}{n!} = \frac{1}{n} \cdot \frac{1}{\binom{n-1}{k}}$. The total contribution from A_k is at most $\frac{1}{n} \cdot \frac{\binom{i-1}{k}}{\binom{n-1}{k}}$. Therefore,

$$\phi_i(w_i + 1) \leq \frac{1}{n} \sum_{k=1}^{i-1} \frac{\binom{i-1}{k}}{\binom{n-1}{k}} \leq \frac{1}{n} \sum_{k=1}^{i-1} \left(\frac{1}{2}\right)^k < \frac{1}{n}.$$

Theorem 4 shows that if there are at most $\frac{n}{2} - 1$ players with weight at most w_i, the quota $w_i + 1$ will always be worse for player i than the quota 1.

As Theorems 3 and 2 show, deciding whether a given quota q is in $opt(\phi_i)$ or $pess(\phi_i)$ is coNP-hard. However, there are certain values of i for which these problems become easy. Specifically, consider the problem of checking if $q \in opt(\phi_1)$. By monotonicity, we have $\phi_1(q) \leq \cdots \leq \phi_n(q)$; thus, $\phi_1(q) \leq \frac{1}{n}$ and, moreover,

$\phi_1(q) = \frac{1}{n}$ if and only if $\phi_1(q) = \ldots = \phi_n(q)$, which happens, e.g., if $q = 1$. Thus, $q \in opt(\phi_1)$ if and only if $\phi_1(q) = \ldots = \phi_n(q)$. Similarly, $q \in pess(\phi_n)$ if and only if $\phi_1(q) = \ldots = \phi_n(q)$. It turns out that deciding whether all players have the same Shapley value (or, equivalently, whether $\phi_n(q) = \phi_1(q)$) is easy.

Theorem 5. *There exists a poly-time algorithm that checks whether* $\phi_n(q) = \phi_1(q)$.

Algorithm 1: FIND-SET($\mathbf{w}, q$)

```
for k = 1 to n − 2 do
    A ← {2, ..., k + 1}, B ← N\(A ∪ {1, n});
    while B ≠ ∅ do
        if q − w_n ≤ w(A) < q − w_1 then
            return A;
        i ← min(A), j ← min(B) ;
        A ← A \ {i} ∪ {i}, B ← B \ {j} ;
return "no";
```

Proof. Observe that $\phi_n(q) > \phi_1(q)$ if and only if there is a set $A \subseteq N \setminus \{i, j\}$ for which player n is pivotal but player 1 is not, i.e., $q - w_n \leq w(A) < q - w_1$. Algorithm 1 finds such a set if it exists, and returns "no" otherwise. It iteratively tries to find such a set of size $1 \leq k \leq n-2$ as follows. It starts with a set of size k that minimizes $w(A)$ and repeatedly (i) removes the smallest element and (ii) adds the smallest yet unused element. Each swap does not decrease the weight of set A and increases it by at most $w_{n-1} - w_2 \leq w_n - w_1$. This process stops if it either finds a set with the desired weight or if there a no elements left to swap in. Note that in the latter case the last set to be considered contained the k largest elements. This guarantees that this process finds a set of size k with $q - w_n \leq w(A) < q - w_1$, if such a set exists. There are at most $n - 2$ swaps in each of the $n - 2$ iterations, which guarantees polynomial running time.

Algorithm 1 gives rise to an even simpler method of deciding whether $\phi_n(q) > \phi_1(q)$. An iteration for a size of k is guaranteed to return a set A if and only it starts with a sufficiently small set and the largest set of size k is large enough.

Corollary 1. $\phi_n(q) > \phi_1(q)$ *if and only there exist a* $k \in [1, n-2]$ *with* $\sum_{i=2}^{k+1} w_i < q - w_1$ *and* $\sum_{i=n-k}^{n-1} w_i \geq q - w_n$.

Corollary 2. *There exist poly-time algorithms for checking whether* $q \in opt(\phi_1)$ *and whether* $q \in pess(\phi_n)$.

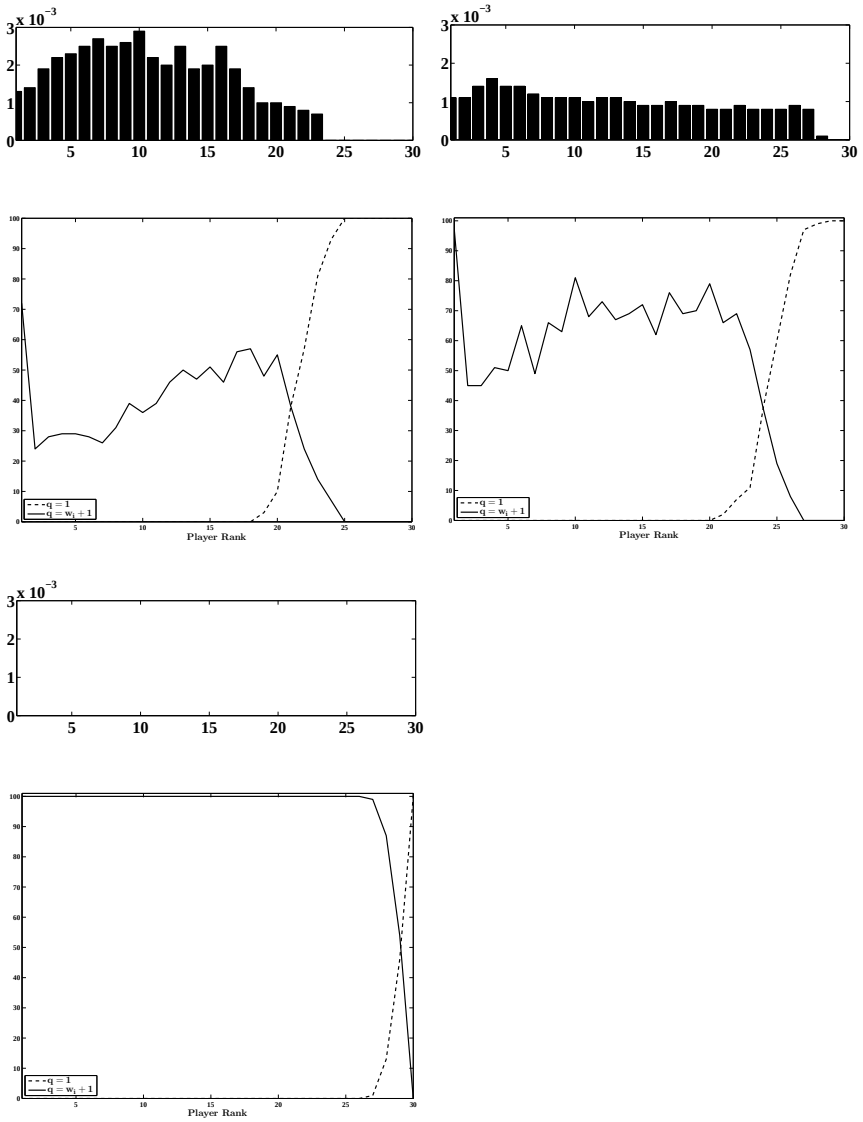


Fig. 4: The X-axis is the rank of the player. In the first row of graphs, the bar in position i indicates the difference between $\min(\phi_i(w_i+1), \phi_i(1))$ and $\min \phi_i(q)$. In the second row of graphs, the Y-axis indicates the number of times (out of 100 trials) that, respectively, $1 \in \mathit{pess}(\phi_i)$ and $w_i + 1 \in \mathit{pess}(\phi_i)$. The graphs show the results for weights drawn according to the uniform (top left), normal (top right) and Poisson (bottom) distributions

5 Empirical Results

We have conjectured that two values of the quota that are likely to minimize the Shapley value of player i are the quotas 1 and w_i+1. To verify this empirically, we considered three different distributions of weights: uniform on $[1, 40]$, normal with $\mu = 30, \sigma^2 = 15$ (negative weights were corrected by increasing μ appropriately) and Poisson distribution with mean 20. For each distribution, we conducted 100 tests. In each test, we generated 30 weights according to the given distribution, and checked whether the Shapley value of player $i \in [1, 30]$ is minimized at $q \in \{1, w_i + 1\}$. The results are graphed in Figure 4.

It appears that for the uniform distribution, the likelihood of the global minimum being at $w_i + 1$ is relatively low. However, when the weights are distributed according to the normal or Poisson distributions, the likelihood of this event increases dramatically. Moreover, in all 100 experiments for the Poisson distribution the minimum occurred at $w_i + 1$ or 1, i.e., $pess(\phi_i) \subseteq \{1, w_i\}$ (and therefore in the rightmost graph on top all bars are of 0 height). In contrast, for the uniform distribution, it is often the case that $1, w_i + 1 \notin pess(\phi_i)$, especially for small values of i. Even when the global minimum was not at $w_i + 1$ or 1, the average difference between a value in $pess(\phi_i)$ and $\min(\phi_i(w_i + 1), \phi_i(1))$ is small. The bars in Figure 4 show the average difference between the minimum of $\phi_i(q)$ and $\min(\phi_i(w_i + 1), \phi_i(1))$ We see that our heuristic produces results that are very close (or equal) to optimal, especially for bigger players.

We conclude that when the players' weights are tightly clustered (as it typically happens for normal and Poisson distribution) either $q = w_i + 1$ or $q = 1$ is likely to minimize player i's power. When choosing between these two options, the rule of thumb is to set $q = w_i + 1$ for the bottom 70–80% of all voters, and $q = 1$ for all other voters.

Another interesting question that merits empirical investigation is whether the manipulator can incur significant changes of the players' Shapley values if the quota is required to be reasonably close to 50% of the total weight, since such constraints on the quota are very common in practice. Now, in Example 1 any choice of quota between—roughly—25% and 75% of the total weight results in the player's power being very close to his relative weight, i.e., w_{10}/w_N, whereas in Example 2 this is not the case. Our next experiment aims to establish which of these scenarios is more frequent.

To do so, we generate a 30-player game with weights coming either from the uniform distribution on $[1, 40]$ or the Poisson distribution with mean 30. For each player i, we measure the length r of the longest contiguous interval of the quota values (normalized by $w(N)$) for which i's Shapley value is within ε from $w_i/w(N)$, for $\varepsilon = 0.0001, 0.00025, 0.001$. For instance, if $r = 0.7$ for $\varepsilon = 0.001$, then for quotas between $0.15w(N)$ and $0.85w(N)$ player i's Shapley value lies in the interval $[w_i/w(N) - 0.001, w_i/w(N) + 0.001]$. We average over 50 trials. The results are presented in Figure 5 (uniform) and Figure 6 (Poisson).

We observe that, under both distributions, for most players their power is very close to their relative weight for a significant proportion of the quotas. However, for very large players this is less likely to be the case, as illustrated

by Example 2. Interestingly, for different values of ε the graphs are shaped differently; in particular, for very small values of ε the graphs peak around player 20, with the position of the peak being different for the two distributions.

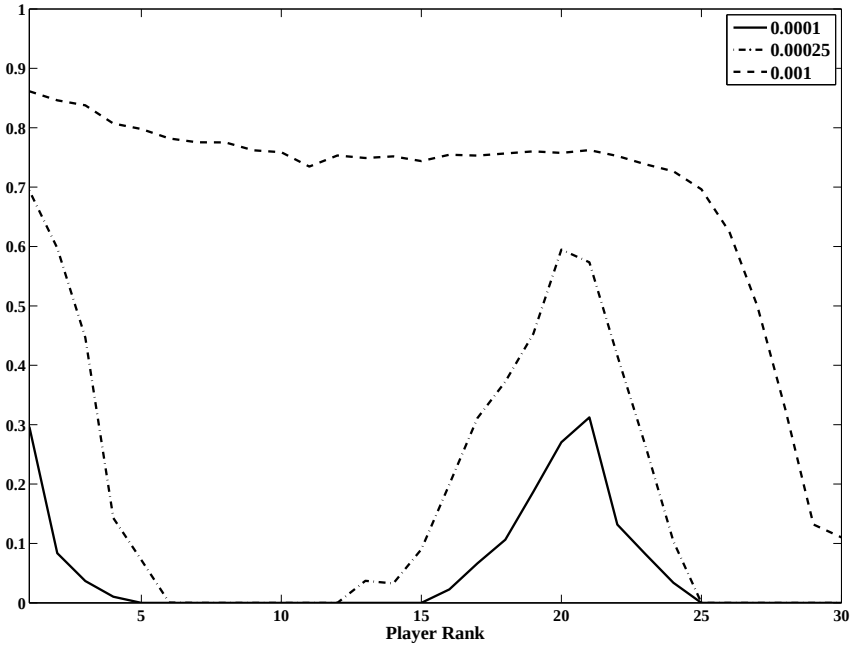


Fig. 5: Length of the interval $\{q : |\phi_i(q) - \frac{w_i}{w(N)}| < \varepsilon\}$ (normalized) for the uniform distribution on $[1 : 40]$ and $\varepsilon = 0.0001, 0.00025, 0.001$

6 Conclusions and Future Work

We explored the behavior of the Shapley value as a function of the quota in weighted voting games. We viewed this problem from the position of a manipulator who aims to maximize/minimize a given player's power. We have shown that, despite a number of hardness results for related problems, maximizing a player's power is easy. While we do not have a polynomial-time algorithm for the minimization problem, our heuristic approach works extremely well, especially for large players. However, in a more realistic scenario where the quota is not allowed to stray too far from 50%, the manipulator cannot do much, especially for smaller players: for a large, centrally symmetric range of quotas the small players' power is fairly close to their weight. In summary, it appears that it is the large players who are most vulnerable to quota manipulation: small changes of the quota may be sufficient to change their power significantly. However, to

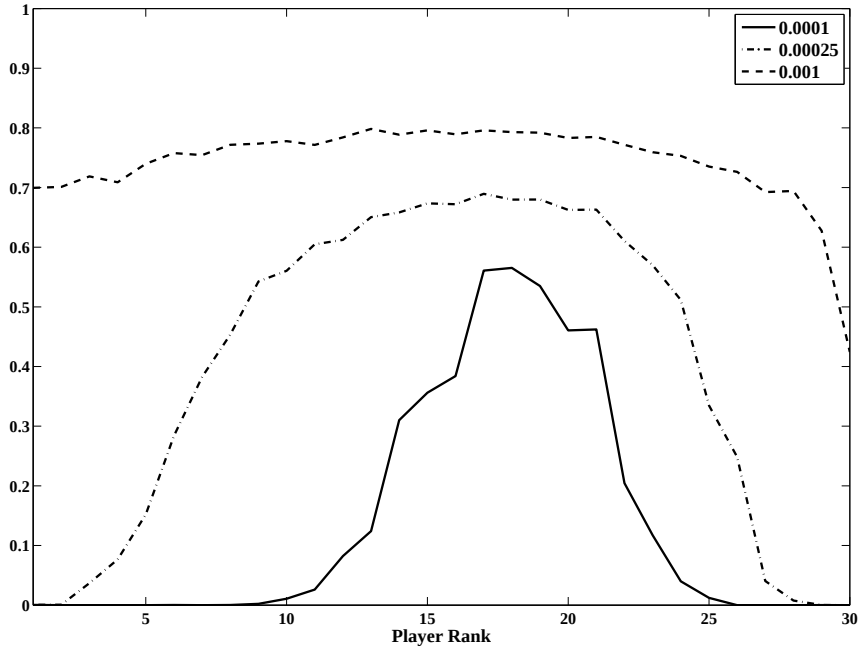


Fig. 6: Length of the interval $\{q : |\phi_i(q) - \frac{w_i}{w(N)}| < \varepsilon\}$ (normalized) for the Poisson distribution with mean 20 and $\varepsilon = 0.0001, 0.00025, 0.001$

change the power of small players in a measurable way, one may need the ability to choose very high/low quota values.

Perhaps the most interesting open question inspired by this work is whether one can find a power-minimizing quota efficiently. A related question is whether there exists a polynomial-time algorithm for maximizing the total power of a *set* of players: indeed, $\phi_i(q)$ is minimal if and only if $\sum_{j \in N \setminus \{i\}} \phi_j(q)$ is maximal.

References

1. Haris Aziz, Yoram Bachrach, Edith Elkind, and Mike Paterson. False-name manipulations in weighted voting games. *JAIR*, 40:57–93, 2011.
2. Y. Bachrach, E. Markakis, E. Resnick, A.D. Procaccia, J.S. Rosenschein, and A. Saberi. Approximating power indices: theoretical and empirical analysis. *JAAMAS*, 20(2):105–122, 2010.
3. J.F. Banzhaf. Weighted voting doesn't work: a mathematical analysis. *Rutgers Law Review*, 19:317–343, 1965.
4. X. Deng and C.H. Papadimitriou. On the complexity of cooperative solution concepts. *Math. Oper. Res.*, 19(2):257–266, 1994.
5. P. Dubey and L.S. Shapley. Mathematical properties of the Banzhaf power index. *Math. Oper. Res.*, 4(2):99–131, 1979.

6. Piotr Faliszewski and Lane Hemaspaandra. The complexity of power-index comparison. *Theor. Comp. Sci.*, 410(1):101 – 107, 2009.
7. S.S. Fatima, M. Wooldridge, and N.R. Jennings. A linear approximation method for the Shapley value. *AIJ*, 172(14):1673–1699, 2008.
8. D. S. Felsenthal and M. Machover. *The Measurement of Voting Power: Theory and Practice, Problems and Paradoxes*. Edward Elgar Publishing, 1998.
9. M.R. Garey and D.S. Johnson. *Computers and Intractibility*. W.H. Freeman and Company, 1979.
10. D. Leech. Computation of power indices. *Warwick Economic Research Papers*, 2002.
11. D. Leech. Designing the voting system for the council of the European Union. *Public Choice*, 113:437–464, 2002.
12. D. Leech and M. Machover. Qualified majority voting: the effect of the quota. *LSE Res. Online*, 2003.
13. M. Machover. Penrose's square-root rule and the EU council of ministers: Significance of the quota. *Distribution of power and voting procedures in the EU*, 2007.
14. I. Mann and L.S. Shapley. *Values of Large Games VI: Evaluating the Electoral College Exactly*. The RAND Corporation, 1962.
15. Y. Matsui and T. Matsui. NP-completeness for calculating power indices of weighted majority games. *Theor. Comp. Sci.*, 263(1-2):305–310, 2001.
16. S. Merrill. Approximations to the Banzhaf index. *American Mathematical Monthly*, 89:108–110, 1982.
17. K. Prasad and J.S. Kelly. NP-completeness of some problems concerning voting games. *Int. J. Game Theory*, 19:1–9, 1990.
18. L.S. Shapley. A value for n-person games. In *Contributions to the Theory of Games, vol. 2*, Annals of Mathematics Studies, no. 28, pages 307–317. 1953.
19. L.S. Shapley and M. Shubik. A method for evaluating the distribution of power in a committee system. *Am. Polit. Sci. Rev.*, 48(3):787–792, 1954.
20. Michael Zuckerman, Piotr Faliszewski, Yoram Bachrach, and Edith Elkind. Manipulating the quota in weighted voting games. *AAAI'08*, pages 215–220, 2008.