Dynamic Weighted Voting Games

Edith Elkind
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
eelkind@ntu.edu.sg

Dmitrii Pasechnik
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
dima@ntu.edu.sg

Yair Zick
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
yair0001@ntu.edu.sg

ABSTRACT

We initiate the study of *dynamic cooperative games*—cooperative games where the characteristic function may change over time. We introduce two types of algorithmic problems for such games: computing a given solution concept at time t, and checking that a certain function of the game (e.g., the Shapley value of a given player or the value of the least core) remains within given bounds during time interval $[t_0, t_1]$. We then investigate the complexity of these problems for *dynamic weighted voting games*, where the weight of each player and the quota are functions of time that are given by low-degree polynomials with integer coefficients. We provide pseudopolynomial algorithms for problems of both types, for a variety of solution concepts. We then use our results to investigate the changes in power distribution in the Council of the European Union over the next 50 years.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

Keywords

weighted voting; cooperative game; time

1. INTRODUCTION

Cooperative game theory studies interactions among strategic agents who can benefit from working together and sharing costs/payoffs of their joint actions. Briefly, a *(transferable utility) cooperative game* is defined by a set of players and a *characteristic function*, which for every subset of players specifies the amount that the players in this subset can earn by working together. The payoff earned by a coalition is them distributed among the coalition members according to some *solution concept*, such as the core or the Shapley value.

There are several families of cooperative games that received substantial attention from the multiagent research community in recent years, due to their ability to model interesting real-life scenarios. One such example is the family of *weighted voting games*. These are cooperative games that can be described by a vector of weights (one for each player) and a quota: a coalition is winning (has value 1) if its total weight meets or exceeds the quota and losing (has value 0) otherwise. Such games model parliamentary voting, where agents are parties and the weight of each party is the number of representatives it has, or shareholder voting, where each voter's weight is the number of shares she owns. They have been used to formally analyze the power distribution in the Council of the European Union [12] and the United Nations security council [9, 16]. They can also be used to reason about agent societies, where an agent's weight corresponds to the amount of resources (time, money, battery power) it contributes.

Now, traditionally cooperative games are viewed as static objects. However, in some applications of cooperative game theory the characteristic function of the game may evolve over time. For instance, in the Council of the European Union a country's voting weight depends on its population, and over the next few decades the populations of the EU member states are likely to change considerably. Similarly, if we use weighted voting to model decision-making in a multiagent system and associate an agent's weight with the amount of resources it has, the agents' weights will change as they expend their resources to execute tasks at hand. Often it can be assumed that the game changes in a predictable manner: for instance, one can obtain fairly accurate population estimates for the EU countries by analyzing historic data. For weighted voting games, this means that the players' weights can be described by succinctly representable functions of time (e.g., low-degree polynomials). For a given time interval $[t_0, t_1]$, these functions define a continuum of weighted voting games: each $t \in [t_0, t_1]$ corresponds to a game $\mathcal{G}(t)$ obtained by evaluating the weight and the quota functions at time t.

Given an evolving, or *dynamic*, cooperative game, we may wish to understand its properties, either at a certain point in the future, or over a given time interval. For instance, we may want to check if a particular event (such as, e.g., a given agent becoming a dummy player) *never* happens as t changes continuously from t_0 to t_1. More formally, the research challenges associated with dynamic cooperative games can be broadly classified into two categories:

(i) computing various solution concepts (such as the Shapley value or the least core) for the game $\mathcal{G}(t)$ for a given value of t;

(ii) deciding whether the game $\mathcal{G}(t)$ possesses certain properties (such as having a non-empty core) for all t in a given time interval $[t_0, t_1]$.

Now, questions of type (i) may appear to be easy: we can simply instantiate the game $\mathcal{G}(t)$ by substituting the given

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

value of t into the description of the evolving game, and then apply the algorithms for computing the respective solution concept in a static game. However, it is not clear that this naive approach is optimal from the computational perspective: if t is sufficiently "complex", it may be more efficient to solve the game at $t = 0$ and then "evolve" the solution as the time changes from 0 to t; this is reminiscent of the homotopy method in numerical optimization (see, e.g., [3]). Subsequently, we will see that this intuition can be made precise in the context of weighted voting games.

Questions of type (ii) appear to be more difficult, as they involve a continuum of games: one for each value of t in $[t_0, t_1]$. One way to approach this issue is to discretize the time interval $[t_0, t_1]$, i.e., split it into a finite number of subintervals so that the game $\mathcal{G}(t)$ remains unchanged on every subinterval, and solve our problem for every subinterval. While this approach is not universally applicable, it can be shown to work for weighted voting games where the players' weights and the quota are described by polynomial functions of time. However, finding an appropriate discretization and showing that it leads to an efficient algorithm for computing standard solution concepts is not immediate.

Our Contribution We formally define *dynamic weighted voting games*, that is, weighted voting games where the weights and the quota are functions of time. We investigate such games from a computational perspective under the assumption that the weights and the quota are polynomial functions of time whose coefficients are integers given in unary. We consider several standard notions of cooperative game theory, such as the Shapley value, the least core, and the cost of stability. We show that, given a time point t, we can efficiently determine the agents' payoffs according to the Shapley value, as well as compute the value of the least core and the cost of stability in the game $\mathcal{G}(t)$. We also show that, given a time interval $[t_0, t_1]$ and bounds L and B such that $L < B$, we can efficiently decide whether a given agent's Shapley value stays within $[L, B]$ for every game $\mathcal{G}(t)$ with $t \in [t_0, t_1]$; similar results can be obtained for stability-related notions such as the value of the least core and the cost of stability. Our proofs combine dynamic programming techniques with arguments from algorithmic real algebraic geometry. Finally, we observe that our methods generalize to vector weighted voting games and apply them to analyze the dynamic vector weighted voting game that describes the Council of the European Union under the Treaty of Lisbon.

2. PRELIMINARIES

We denote by $\mathbb{R}_+$, $\mathbb{Q}_+$ and $\mathbb{Z}_+$ the sets of all non-negative real, rational and integer numbers, respectively. A *cooperative game* $\mathcal{G} = \langle N, v\rangle$ is given by a set of *agents*, or *players*, $N = \{1, \ldots, n\}$ and a *characteristic function* $v : 2^N \to \mathbb{R}_+$ that satisfies $v(\emptyset) = 0$; for every subset of agents $S \subseteq N$, $v(S)$ is the *value* of S. Subsets of agents are often referred to as *coalitions*; the *grand coalition* is simply the set N of all agents. Given a coalition $S \subseteq N$ and a vector $(x^1, \ldots, x^n) \in \mathbb{R}^n$, we write $x(S) = \sum_{i\in S} x^i$. A cooperative game is called *simple* if for all $S \subseteq N$ we have $v(S) \in \{0, 1\}$ and for every $S, T \in 2^N$ such that $S \subseteq T$ it holds that $v(S) \leq v(T)$. In a simple game $\mathcal{G} = \langle N, v\rangle$ a coalition S is said to be *winning* if $v(S) = 1$ and *losing* otherwise. A player i in a simple game $\mathcal{G} = \langle N, v\rangle$ is said to be a *dummy* if $v(S) = v(S \cup \{i\})$ for every $S \subseteq N \setminus \{i\}$; i is said to be a *veto* player if $v(S) = 0$ for every $S \subseteq N \setminus \{i\}$.

Weighted voting games are a well-studied class of simple games. A weighted voting game with a set of players N is described by a list of *weights* $\mathbf{w} = (w^1, \ldots, w^n)$, where $w^i \in \mathbb{R}_+$ for each $i \in N$, and a *threshold*, or *quota*, $q \in \mathbb{R}_+$: a coalition $S \subseteq N$ is winning if its total weight $w(S) = \sum_{i\in S} w^i$ is at least q and losing otherwise. We assume that the grand coalition is winning, i.e., $w(N) \geq q$. We will denote a weighted voting game with a set of players N, a weight vector $\mathbf{w}$ and a quota q by $\langle N, \mathbf{w}, q\rangle$.

One can extend the basic framework of weighted voting games to *vector weighted voting games*. A vector weighted voting game with a set of players N is given by a list of weight vectors $(\mathbf{w}_1, \ldots, \mathbf{w}_m)$, where $\mathbf{w}_\ell = (w^1_\ell, \ldots, w^n_\ell)$ for all $\ell = 1, \ldots, m$, and a list of thresholds $(q_1, \ldots, q_m)$. For each $i \in N$, w^i_ℓ is agent i's weight in the weight vector $\mathbf{w}_\ell$. A coalition $S \subseteq N$ is winning in the vector weighted voting game $\mathcal{G} = \langle N, \mathbf{w}_1, \ldots, \mathbf{w}_m, q_1, \ldots, q_m\rangle$ if and only if it is winning in the weighted voting game $\langle N, \mathbf{w}_\ell, q_\ell\rangle$ for each $\ell = 1, \ldots, m$. The number m is called the *dimension* of $\mathcal{G}$.

Example 2.1. According to the Treaty of Lisbon [2], in the Council of the European Union a proposal made by the European Commission passes if and only if it is supported by 55% of the EU countries and 65% of the EU population. Thus, voting in the Council can be seen as a vector weighted voting game $\langle N, \mathbf{w}_1, \mathbf{w}_2, q_1, q_2\rangle$, where N is the set of EU member states, $|N| = 27$, $\mathbf{w}_1$ corresponds to the population of each state, $\mathbf{w}_2 = (1, \ldots, 1)$, $q_1 = 0.65w_1(N)$, and $q_2 = 0.55w_2(N)$.

Solution Concepts for Weighted Voting Games An outcome of a cooperative game $\mathcal{G} = \langle N, v\rangle$ is a partition of players into coalitions together with a payoff vector that specifies how the value of each coalition is distributed among its members. In weighted voting games it is usually assumed that the players form the grand coalition; thus, an outcome of a weighted voting game $\langle N, \mathbf{w}, q\rangle$ is simply a vector $(p^1, \ldots, p^n)$ in $(\mathbb{R}_+)^n$ that satisfies $p(N) = 1$.

A *solution concept* is a function that maps a cooperative game to a subset of outcomes. In this paper, we will consider the classic fairness-based solution concept known as the *Shapley value* [17], as well as a standard stability-based solution concept known as the *least core* [13]; we will also consider a measure of stability for coalitional games known as the *cost of stability* [4].

To define the Shapley value, we first introduce some additional notation. Let us denote by $\Pi(N)$ the set of all permutations of N, i.e., the set of all bijective functions $\pi : N \to N$. Given a permutation $\pi \in \Pi(N)$, we denote by $P^i(\pi)$ the set of all predecessors of i in π, i.e., we set

$$P^i(\pi) = \{j \in N \mid \pi(j) < \pi(i)\}.$$

Given an agent $i \in N$ and a coalition $S \subseteq N \setminus \{i\}$, we say that i is *pivotal* for S in a simple game $\mathcal{G} = \langle N, v\rangle$ if $v(S) = 0$, but $v(S \cup \{i\}) = 1$. Similarly, we say that i is pivotal for a permutation $\pi \in \Pi(N)$ if i is pivotal for $P^i(\pi)$. The *Shapley value* of agent i in $\mathcal{G}$, denoted by $\varphi^i(\mathcal{G})$, is the probability that agent i is pivotal for a permutation $\pi \in \Pi(N)$ chosen uniformly at random. Formally,

$$\varphi^i(\mathcal{G}) = \frac{1}{n!} \sum_{\pi\in\Pi(N)} v(P^i(\pi) \cup \{i\}) - v(P^i(\pi)).$$

The most widely used solution concept for cooperative games that is based on the idea of stability is the core. Given a cooperative game $\mathcal{G} = \langle N, v\rangle$, its *core*, denoted by $Core(\mathcal{G})$, is the set of all outcomes $\mathbf{p} = (p^1, \ldots, p^n)$ such that no coalition can profitably deviate, i.e., $p(S) \geq v(S)$ for all $S \subseteq N$; we say that $\mathbf{p}$ is *stable* if it is in the core of $\mathcal{G}$. While core outcomes are highly desirable, many cooperative games have empty cores. In particular, is is known that a simple game has a non-empty core if and only if it has a veto player. For such games, it is useful to have a weaker notion of stability.

One such notion is that of the ε-core [13]: an outcome $\mathbf{p} = (p^1, \ldots, p^n)$ is said to be in the ε-core of a cooperative game $\mathcal{G} = \langle N, v\rangle$ for some $\varepsilon \in \mathbb{R}$ if $p(S) \geq v(S) - \varepsilon$ for all $S \subseteq N$. We denote the ε-core of $\mathcal{G}$ by $Core_\varepsilon(\mathcal{G})$. Intuitively, the outcomes in the ε-core are stable if a deviation carries a cost of ε. Let $\varepsilon^*(\mathcal{G}) = \inf\{\varepsilon \mid Core_\varepsilon(\mathcal{G}) \neq \emptyset\}$; a simple continuity argument shows that $Core_{\varepsilon^*(\mathcal{G})}(\mathcal{G}) \neq \emptyset$. The $\varepsilon^*(\mathcal{G})$-core of $\mathcal{G}$ is called the *least core* of $\mathcal{G}$; the quantity $\varepsilon^*(\mathcal{G})$ is called the *value of the least core* of $\mathcal{G}$. Note that in the definition of the least core we do not require ε to be non-negative; indeed, if the game has a non-empty core, the value of the least core may be negative.

Another approach to relaxing the notion of the core was recently proposed by Bachrach et al. [4]. Intuitively, it is based on computing the smallest subsidy that should be given to the grand coalition to ensure that no coalition of players has an incentive to deviate. Formally, given a cooperative game $\mathcal{G} = \langle N, v\rangle$ and a $\Delta \geq 0$, we define the *adjusted game* $\mathcal{G}_\Delta = \langle N, v_\Delta\rangle$ by setting $v_\Delta(N) = v(N) + \Delta$, $v_\Delta(S) = v(S)$ for $S \subsetneq N$. The *cost of stability* of $\mathcal{G}$ is then defined as $\mathrm{CoS}(\mathcal{G}) = \inf\{\Delta \geq 0 \mid Core(\mathcal{G}_\Delta) \neq \emptyset\}$; a continuity argument shows that $Core(\mathcal{G}_{\mathrm{CoS}(\mathcal{G})}) \neq \emptyset$.

Polynomial Root Separation and Isolation We will make use of two standard results from algorithmic real algebraic geometry. In what follows, we consider polynomials of degree at most k with integer coefficients whose maximum absolute value is $W > 0$.

THEOREM 2.2 (ROOT ISOLATION [5]). *Given a polynomial $p(\cdot)$, let $r_1, \ldots, r_m$ be the roots of p. It is possible to find a list of disjoint open intervals $I_1, \ldots, I_m$ with endpoints in $\mathbb{Q}$ such that for all $1 \leq \ell \leq m$ we have $r_\ell \in I_\ell$; $I_1, \ldots, I_m$ can be found in time polynomial in $||W||^k$. Moreover, given an $\varepsilon > 0$, it is possible to ensure that the length of all open intervals is at most ε in time polynomial in $||W||^k$ and $\frac{1}{\varepsilon}$.*

LEMMA 2.3 (ROOT SEPARATION [6]). *Given two polynomials $p(\cdot), q(\cdot)$, let r, r' be roots of either p or q. Then $|r - r'| \geq \delta$, where $\frac{1}{\delta} \leq ||W||^{2k-1}$.*

Now suppose we have a collection of polynomials $\mathcal{P}$. For each $p \in \mathcal{P}$, let $R(p)$ denote the set of real roots of p, and let $R(\mathcal{P}) = \cup_{p\in\mathcal{P}} R(p)$. We can use Theorem 2.2 and Lemma 2.3 to output a list $\mathcal{I} = (I_1, \ldots, I_s)$ of pairwise disjoint intervals and a mapping $L : \mathcal{I} \to 2^{\mathcal{P}}$ that associates every interval in $\mathcal{I}$ with a subset of $\mathcal{P}$ so that

(1) each element of $R(\mathcal{P})$ belongs to some interval in $\mathcal{I}$;

(2) each interval in $\mathcal{I}$ contains exactly one element of $R(\mathcal{P})$;

(3) for each $I \in \mathcal{I}$, $L(I)$ lists all polynomials in $\mathcal{P}$ whose root is contained in I (note that, even though I contains a single point in $R(\mathcal{P})$, this point can be a root of several polynomials in $\mathcal{P}$).

Moreover, Theorem 2.2 and Lemma 2.3 imply that $(\mathcal{I}, L)$ can be computed in time polynomial in $||W||^{2k-1}$. We will refer to the data structure $(\mathcal{I}, L)$ as the *interval representation* for $R(\mathcal{P})$.

3. OUR MODEL

We will now formally define *dynamic weighted voting games*, which will be the focus of this paper.

DEFINITION 3.1. *A* dynamic weighted voting game *is a tuple*

$$\mathcal{G}(\cdot) = \langle N, \mathbf{w}(\cdot), q(\cdot), t_0, t_1\rangle,$$

where $t_0, t_1 \in \mathbb{R}_+ \cup \{+\infty\}$, $t_0 \leq t_1$, $\mathbf{w}(\cdot) = (w^1(\cdot), \ldots, w^n(\cdot))$, the mappings $w^i : [t_0, t_1] \to \mathbb{R}_+$, $i \in N$, associate every point in the interval $[t_0, t_1]$ with a non-negative real weight, and the mapping $q : [t_0, t_1] \to \mathbb{R}_+$ associates every point in $[t_0, t_1]$ with a non-negative real quota.

Intuitively, we think of $[t_0, t_1]$ as the time interval over which the game evolves: initially, the weight of the i-th player is given by $w^i(t_0)$ and the quota is given by $q(t_0)$, and then the weights and the quota change according to $w^1(\cdot), \ldots, w^n(\cdot)$ and $q(\cdot)$, respectively.

To study algorithmic properties of dynamic weighted voting games, we need to specify a representation formalism for them. In what follows, we assume that the weights and the quota are given by polynomials with integer coefficients. Formally, we assume that there exists a $k \in \mathbb{Z}_+$ such that $w^i(t) = w^i_k t^k + \cdots + w^i_1 t + w^i_0$ for each $i \in N$ and $q(t) = q_k t^k + \cdots + q_1 t + q_0$, where $w^i_j, q_j \in \mathbb{Z}$ for all $i \in N, j = 0, \ldots, k$. This assumption is reasonable, as continuous functions are well approximated by polynomials; besides, polynomial functions arise naturally if the weights are obtained by extrapolation. Note that we do not require the coefficients to be positive; rather, we assume that the interval $[t_0, t_1]$ is selected so that the polynomials $w^i(\cdot)$, $i \in N$, and $q(\cdot)$ take non-negative values on $[t_0, t_1]$. Also, we assume that t_0 and t_1 are rational numbers. We set $W = \max\{\max_{i\in N, j=1,\ldots,k} |w^i_j|, \max_{j=1,\ldots,k} |q_j|\}$; also, given a rational number r, we let $||r||$ denote the length of the binary encoding of r.

We remark that dynamic weighted voting games generalize weighted voting games: it is straightforward to represent a (static) weighted voting game $\langle N, \mathbf{w}, q\rangle$ as a dynamic weighted voting game (e.g., we can set $w^i(t) \equiv w^i$). Therefore, in what follows we seek algorithms whose running time is polynomial in n and W, i.e., we implicitly assume that the coefficients of our polynomials are given in unary: this is because when weights are assumed to be given in binary, the problems considered in this paper are known to be hard even for static weighted voting games [7].

We are now ready to state the computational problems that will be investigated in the rest of the paper.

DEFINITION 3.2. *An instance of* POINT-SHAPLEY *is given by a dynamic weighted voting game*

$$\mathcal{G}(t) = \langle N, \mathbf{w}(\cdot), q(\cdot), t_0, t_1\rangle,$$

a point $t \in [t_0, t_1] \cap \mathbb{Q}$, and an agent $i \in N$. The goal is to compute the Shapley value of agent i in $\mathcal{G}(t)$.

DEFINITION 3.3. *An instance of* INT-SHAPLEY *is given by a dynamic weighted voting game*

$$\mathcal{G}(t) = \langle N, \mathbf{w}(\cdot), q(\cdot), t_0, t_1 \rangle,$$

a pair of points $L, U \in \mathbb{Q}$, $L \leq U$, *and an agent* $i \in N$. *It is a "yes" instance if* $L < \varphi^i(\mathcal{G}(t)) \leq U$ *for all* $t \in [t_0, t_1]$ *and a "no" instance otherwise.*

The problems POINT-LEASTCORE and POINT-COS as well as INT-LEASTCORE and INT-COS, which deal with computing/bounding, respectively, the value of the least core and the cost of stability, are defined similarly; the only difference is that we do not need to specify an agent i in the description of an instance of the problem. We skip the formal definitions due to space constraints.

Our proofs rely on the notion of a *signature* of a set of agents. Given a dynamic weighted voting game $\mathcal{G}(\cdot) = \langle N, \mathbf{w}(\cdot), q(\cdot), t_0, t_1 \rangle$ and a coalition $S \subseteq N$, we consider the polynomial

$$w^S(t) = \sum_{i \in S} w^i(t) = \sum_{j=0}^{k} \sum_{i \in S} w_j^i t^j.$$

The *signature* of a set S, denoted by $sig(S)$, is the list of the coefficients of $w^S(\cdot)$; if $w^S(t) = W_k t^k + \cdots + W_1 t + W_0$, we write $sig(S) = (W_k, \ldots, W_1, W_0)$. Note that the absolute value of each coefficient of $w^S(t)$ is bounded by nW. Let $\Sigma = [-nW, nW]^{k+1}$ denote the space of all possible signatures for $\mathcal{G}(\cdot)$. Overloading notation, we denote by $\sigma(\cdot)$ the polynomial associated with a signature $\sigma \in \Sigma$: if $\sigma = (W_k, \ldots, W_1, W_0)$, then $\sigma(t) = W_k t^k + \cdots + W_1 t + W_0$. A set's signature is important for two reasons. First, it completely describes the set's behavior as the time goes by: if we know the signature of a coalition S, we can tell when it is winning, when it is losing, and when agents become pivotal for S. The second—and more important—reason is that there are not too many different signatures: we have $|\Sigma| \leq (2nW+1)^{k+1}$, so we can explicitly go over all signatures in time polynomial in n and W.

4. SHAPLEY VALUE IN DYNAMIC WEIGHTED VOTING GAMES

In this section, we show that both POINT-SHAPLEY and INT-SHAPLEY admit efficient algorithms.

Our algorithms for POINT-SHAPLEY and INT-SHAPLEY are based on the classic dynamic programming algorithm for computing the Shapley value [14]; recall that, given an n-player weighted voting game $\langle N.\mathbf{w}, q \rangle$, this algorithm can determine the Shapley value of any given player in time $\mathcal{O}(n^2 w(N))$. Similarly, the running time of our algorithms on a dynamic weighted voting game $\langle N, \mathbf{w}(\cdot), q(\cdot), t_0, t_1 \rangle$ is polynomial in n and W.

THEOREM 4.1. *An instance of* POINT-SHAPLEY *given by a dynamic weighted voting game* $\mathcal{G}(\cdot) = \langle N, \mathbf{w}(\cdot), q(\cdot), t_0, t_1 \rangle$, *a point* $t \in [t_0, t_1] \cap \mathbb{Q}$, *and an agent* $i \in N$ *can be solved in time* $\text{poly}(n^k, W^k, ||t||)$.

PROOF. To simplify notation, we assume that $i = n$, i.e., we are interested in the Shapley value of the last player. For every $S \subseteq N$ let $\delta^S(t) = 1$ if player n is pivotal for S at time t and $\delta^S(t) = 0$ otherwise. Observe that

$$\varphi^n(\mathcal{G}(t)) = \frac{1}{n!} \sum_{S \subseteq N} |S|!(n - |S| - 1)! \delta^S(t). \qquad (1)$$

We have $\delta^S(t) = 1$ if and only if $w^S(t) < q(t)$ and $w^S(t) \geq q(t) - w^n(t)$. This implies that for every $S, T \subseteq N$ such that $sig(S) = sig(T)$ we have $\delta^S(t) \equiv \delta^T(t)$. Thus, overloading notation, for every $\sigma \in \Sigma$ we let $\delta^\sigma(t) = 1$ if $\sigma(t) < q(t)$ and $\sigma(t) \geq q(t) - w^n(t)$ and $\delta^\sigma(t) = 0$ otherwise. Note that $\delta^\sigma(t) = \delta^S(t)$ for every S such that $sig(S) = \sigma$.

Given a $\sigma \in \Sigma$ and an $s \in \{0, \ldots, n-1\}$, let

$$X[\sigma, s] = |\{S \subseteq N \setminus \{n\} \mid sig(S) = \sigma, |S| = s\}|.$$

Then equation (1) can be rewritten as follows:

$$\varphi^n(\mathcal{G}(t)) = \frac{1}{n!} \sum_{\sigma \in \Sigma} \sum_{s=0}^{n-1} s!(n - s - 1)! X[\sigma, s] \delta^\sigma(t). \qquad (2)$$

The expression on the right-hand side of equation (2) only has $n|\Sigma|$ terms. We will now argue that each of these terms can be computed efficiently.

Observe first that for any given value of t and any $\sigma \in \Sigma$ the quantity $\delta^\sigma(t)$ can be computed by evaluating three polynomials ($\sigma(\cdot)$, $q(\cdot)$, and $q(\cdot) - w^n(\cdot)$); the degree of each of these polynomials is at most k, and their coefficients are integers between $-nW$ and nW. Thus, we can find $\delta^\sigma(t)$ in time polynomial in $\log W$, n, k, and $||t||$.

Further, the quantities $X[\sigma, s]$ are easy to compute by dynamic programming. Indeed, for all $\sigma \in \Sigma$, $s = 0, \ldots, n-1$ and $j = 1, \ldots, n-1$, let

$$X[\sigma, s; j] = |\{S \subseteq \{1, \ldots, j\} \mid sig(S) = \sigma, |S| = s\}|.$$

We can easily compute $X[\sigma, s; j]$ if $j = 1$ or $s = 0$. Specifically, we have $X[\sigma, s; 1] = 1$ if $s = 1$ and $\sigma = sig(\{1\})$ or $s = 0$ and $\sigma = (0, \ldots, 0)$, and $X[\sigma, s; 1] = 0$ otherwise. Similarly, for every $j = 2, \ldots, n$ we have $X[\sigma, 0; j] = 1$ if $\sigma = (0, \ldots, 0)$, and $X[\sigma, 0; j] = 0$ otherwise. Further, for $j > 1, s > 0$ we have $X[\sigma, s; j] = X[\sigma, s; j-1] + X[\sigma - sig(\{j\}), s-1; j-1]$ if $\sigma - sig(\{j\}) \in \Sigma$ and $X[\sigma, s; j] = X[\sigma, s; j-1]$ otherwise. Using this recurrence, we can compute all $X[\sigma, s; j]$ in time $\mathcal{O}(n^2(2nW+1)^{k+1})$. Finally, we have $X[\sigma, s] = X[\sigma, s; n-1]$ for all $\sigma \in \Sigma$ and all $s = 0, \ldots, n-1$.

Thus, we can use equation (2) to solve POINT-SHAPLEY in time $\text{poly}(n^k, W^k, ||t||)$. □

It is interesting to compare Theorem 4.1 with the naive approach to computing the players' Shapley values at time t, which was mentioned in Section 1, namely, evaluating all weights and quota at time t and computing the players values in the resulting static game. Suppose for simplicity that the weights and the quota are linear functions of time, i.e., there exist $a^0, a^1, \ldots, a^n$ and $b^0, b^1, \ldots, b^n$ such that $w^i(t) = a^i t + b^i$ for all $i = 1, \ldots, n$, and $q(t) = a^0 t + b^0$, where a^i, b^i, $i = 0, \ldots, n$, are integers given in unary, and suppose that t is an integer number given in binary. Suppose also $a^i > 0$, $b^i > 0$ for some $i = 1, \ldots, n$. Then we have $w^i(t) \geq t$. The running time of the dynamic programming algorithm for computing the Shapley value scales linearly with the maximum weight, so in this case it will be at least t. On the other hand, the running time of the algorithm presented in Theorem 4.1 is polynomial in $||t|| \sim \log t$. To put it differently, the naive approach requires that not only the coefficients of the polynomials but also the point t are given in unary, whereas our algorithm remains computationally efficient when t is given in binary.

We will now use the techniques developed in the proof of Theorem 4.1 to solve INT-SHAPLEY.

THEOREM 4.2. *An instance of* INT-SHAPLEY *given by a dynamic weighted voting game* $\mathcal{G}(\cdot) = \langle N, \mathbf{w}(\cdot), q(\cdot), t_0, t_1\rangle$, *a pair of points* $L, U \in \mathbb{Q}$, $L < U$, *and an agent* $i \in N$ *can be solved in time* $\text{poly}(n^k, W^k, ||L||, ||U||, ||t_0||, ||t_1||)$.

PROOF. We assume $i = n$ and use notation introduced in the proof of Theorem 4.1. We first show how to solve our problem under the assumption that $t_0 = 0$, $t_1 = +\infty$; we then explain how to modify our algorithm for arbitrary t_0 and t_1.

For every $\sigma \in \Sigma$ the functions $f^1_\sigma(t) = q(t) - w^n(t) - \sigma(t)$ and $f^2_\sigma(t) = q(t) - \sigma(t)$ are polynomials of degree at most k, so each of them has at most k roots. Let $F_\sigma(t) = f^1_\sigma(t) f^2_\sigma(t)$, and let R_σ be the set of non-negative real roots of F_σ; we have $|R_\sigma| \leq 2k$. The set $\{t \mid \sigma(t) < q(t)\}$ is a union of open intervals; similarly, $\{t \mid \sigma(t) \geq q(t) - w^n(t)\}$ is a union of closed intervals. Hence, the set $\{t \mid \sigma(t) < q(t), \sigma(t) \geq q(t) - w^n(t)\}$ is a union of open, closed, or half-closed intervals with endpoints in $R_\sigma \cup \{0, +\infty\}$. Consequently, the function $\delta^\sigma(\cdot)$ (defined in the proof of Theorem 4.1) remains constant on every interval with endpoints in $R_\sigma \cup \{0, +\infty\}$.

Now, let $R = \cup_{\sigma \in \Sigma} R_\sigma$, and set $\rho = |R|$. Observe that $\rho \leq 2k|\Sigma| = 2k(2nW+1)^{k+1}$. Renumber the points in R so that $R = \{r_1, \ldots, r_\rho\}$, $r_1 \leq \cdots \leq r_\rho$, and let $I_j = (r_j, r_{j+1})$ for $j = 1, \ldots, \rho - 1$; also, set $I_0 = (0, r_1)$, $I_\rho = (r_\rho, +\infty)$. Clearly, for each $j = 1, \ldots, \rho - 1$ and for each $\sigma \in \Sigma$ the function $\delta^\sigma(\cdot)$ remains constant on I_j, and, consequently, $\varphi^n(\cdot)$ is also constant on I_j. We refer to the points in R as *phase transitions*.

We will now describe how $\varphi^n(\cdot)$ changes at a point $r_j \in R$, $j = 1, \ldots, \rho$. Pick some $r_j^- \in I_{j-1}$ and $r_j^+ \in I_j$. The signatures in Σ can be classified into the following five categories with respect to j:

- $\Sigma_j^0 = \{\sigma \mid \delta^\sigma(r_j^-) = \delta^\sigma(r_j^+)\}$;
- $\Sigma_j^1 = \{\sigma \mid \delta^\sigma(r_j) = \delta^\sigma(r_j^+) = 1, \delta^\sigma(r_j^-) = 0\}$;
- $\Sigma_j^2 = \{\sigma \mid \delta^\sigma(r_j) = \delta^\sigma(r_j^+) = 0, \delta^\sigma(r_j^-) = 1\}$;
- $\Sigma_j^3 = \{\sigma \mid \delta^\sigma(r_j) = \delta^\sigma(r_j^-) = 1, \delta^\sigma(r_j^+) = 0\}$;
- $\Sigma_j^4 = \{\sigma \mid \delta^\sigma(r_j) = \delta^\sigma(r_j^-) = 0, \delta^\sigma(r_j^+) = 1\}$.

Further, for any signature $\sigma \in \Sigma$ it is easy to determine which of these five categories it belongs to: it suffices to check whether the polynomials $f^1_\sigma(\cdot)$ and $f^2_\sigma(\cdot)$ change sign at r_j (and, if yes, what is the direction of this change).

If $\sigma \in \Sigma_j^0$, its contribution to the Shapley value of player n does not change as t increases from r_j^- to r_j^+. The contribution of sets with signatures in $\Sigma_j^1 \cup \Sigma_j^2$ is constant on $[r_j^-, r_j)$ and $[r_j, r_j^+]$, while the contribution of sets with signatures in $\Sigma_j^3 \cup \Sigma_j^4$ is constant on $[r_j^-, r_j]$ and $(r_j, r_j^+]$. Thus, we obtain

$$\varphi^n(r_j) = \varphi^n(r_j^-) + \sum_{s=0}^{n-1} s!(n-s-1)!\Big(\sum_{\sigma \in \Sigma_j^1} X[\sigma, s] - \sum_{\sigma \in \Sigma_j^2} X[\sigma, s]\Big) \quad (3)$$

and

$$\varphi^n(r) = \varphi^n(r_j) + \sum_{s=0}^{n-1} s!(n-s-1)!\Big(\sum_{\sigma \in \Sigma_j^4} X[\sigma, s] - \sum_{\sigma \in \Sigma_j^3} X[\sigma, s]\Big) \quad (4)$$

for all $r \in I_j$.

This analysis suggests the following procedure for deciding INT-SHAPLEY. We start by evaluating $\varphi^n(t)$ at $t = 0$: this requires substituting $t = 0$ into the polynomials $f^1_\sigma(\cdot)$ and $f^2_\sigma(\cdot)$ for all $\sigma \in \Sigma$, and using expression (2). We check if the result belongs to $[L, U]$, and output "no" and stop if this is not the case (recall that we assume $t_0 = 0$).

We then process the points in R one by one, from left to right. For each $j = 1, \ldots, \rho$, we first use equation (3) to compute the Shapley value of player n at r_j given his Shapley value on I_{j-1}. Having computed $\varphi^n(r_j)$, we use equation (4) to compute the Shapley value of player n on I_j. We output "no" and stop if at any point in the computation we obtain a value that does not belong to $[L, U]$. If $\varphi^n(\cdot)$ stays between L and U for all r_j, $j = 1, \ldots, \rho$ and on all I_j, $j = 0, \ldots, \rho$, we output "yes".

The difficulty with implementing this procedure is that it seems to require computing the set R, which consists of roots of polynomials of degree up to k, and therefore may contain points in $\mathbb{R} \setminus \mathbb{Q}$. However, it is not hard to see that we can work with the interval representation of R instead. Indeed, to implement our algorithm it suffices to have a sorted list of the elements of R, and, for each $r_j \in R$, to be able to compute the sets Σ_j^ℓ, $\ell = 1, \ldots, 5$. Now, the interval representation $(\mathcal{I}, L)$ of R allows us to sort the elements of R. Further, to decide whether a given polynomial σ belongs to Σ_j^ℓ, $\ell = 1, \ldots, 5$, it suffices to check whether $\sigma \in L(I_j)$ and, if the answer is positive, to evaluate σ at the endpoints of I_j, where I_j is the interval associated with r_j. The endpoints of all intervals in $\mathcal{I}$ are rational numbers that have been computed in time polynomial in $||W||^{2k+1}$, so their bit representation is of size polynomial in $||W||^{2k+1}$. Consequently, one can evaluate σ at these points in time polynomial in $||W||^{2k+1}$. Hence, for $t_0 = 0$, $t_1 = +\infty$ the problem INT-SHAPLEY can be solved in time polynomial in n^k, W^k, $||L||$, and $||U||$, as required.

If t_0, t_1 can be arbitrary, we proceed in the same manner, starting from $t = 0$; however, we only check that the result of our computation belongs to $[L, U]$ for points that lie in $[t_0, t_1]$ and intervals that intersect $[t_0, t_1]$. □

EXAMPLE 4.3. Consider a dynamic weighted voting game with a set of players $N = \{1, 2, 3\}$, weights $w^1(t) = 2t$, $w^2(t) = 3 - t$, $w^3(t) = 1$, quota $q(t) = 3$, and time interval $[t_0, t_1] = [0, 2]$. Player 3 is pivotal for $\{1\}$ if $1 \leq t < 1.5$ and for $\{2\}$ if $0 < t \leq 1$; he is never pivotal for the empty coalition or for $\{1, 2\}$. Thus, we have

$$\varphi^3(t) = \begin{cases} 0 & \text{if } t = 0 \text{ or } t \geq 1.5 \\ \frac{1}{6} & \text{if } 0 < t < 1 \text{ or } 1 < t < 1.5 \\ \frac{1}{3} & \text{if } t = 1 \end{cases}$$

Note that the Shapley value of player 3 at $t = 1$ differs from his Shapley value on the adjacent intervals $(0, 1)$ and $(1, 1.5)$.

5. STABILITY IN DYNAMIC WEIGHTED VOTING GAMES

In this section, we prove analogues of Theorem 4.1 and 4.2 for the least core. That is, we show that POINT-LEASTCORE and INT-LEASTCORE admits efficient algorithms as long as the coefficients of the polynomials that describe the weights are given in unary. We then explain how to extend these results to POINT-COS and INT-COS.

THEOREM 5.1. *An instance of* POINT-LEASTCORE *given by a dynamic weighted voting game* $\mathcal{G}(\cdot) = \langle N, \mathbf{w}(\cdot), q(\cdot), t_0, t_1\rangle$ *together with a point* $t \in [t_0, t_1] \cap \mathbb{Q}$ *can be solved in time* $\mathrm{poly}(n^k, W^k, ||t||)$.

PROOF. The value of the least core of a weighted voting game $\langle N, \mathbf{w}, q\rangle$ can be obtained by solving the following linear program:

$$\begin{aligned} \min \quad & \varepsilon & & & \\ & p^i & \geq & 0 \quad \text{for each } i \in N & (5)\\ & \sum_{i\in N} p^i & = & 1 & (6)\\ & \sum_{i\in S} p^i & \geq & 1-\varepsilon \text{ for each } S \subseteq N \text{ s. t. } \sum_{i\in S} w^i \geq q & (7) \end{aligned}$$

This linear program is known to possess a separation oracle whose running time is polynomial in n and $w(N)$ [8]; this implies that it can be solved in pseudopolynomial time [15]. We will now show that this result can be extended to dynamic weighted voting games, leading to an efficient algorithm for POINT-LEASTCORE.

Given a dynamic weighted voting game $\mathcal{G}(\cdot)$ and a $t \in [t_0, t_1] \cap \mathbb{Q}$, let $LP(t)$ be the linear program for the least core associated with $\mathcal{G}(t)$. As t changes from t_0 to t_1, the linear program $LP(t)$ changes, too: when a losing coalition S becomes winning, we add a constraint of the form $\sum_{i\in S} p_i \geq 1-\varepsilon$, and when a winning coalition S becomes losing, we remove the corresponding constraint. If $sig(S) = sig(T)$, the constraints corresponding to S and T are added/deleted simultaneously. We will use this observation to design a separation oracle for $LP(t)$, for each $t \in [t_0, t_1]$.

Recall that a separation oracle for $LP(t)$ receives a candidate solution $(p^1, \ldots, p^n, \varepsilon) \in \mathbb{Q}^{n+1}$ as its input; it should output "yes" if $(p^1, \ldots, p^n, \varepsilon)$ satisfies all constraints of $LP(t)$ and identify a violated constraint otherwise. To implement such a separation oracle, we proceed as follows.

Consider a candidate solution $(p^1, \ldots, p^n, \varepsilon)$. We first check that it satisfies constraints (5) and (6); this can be done in time polynomial in the bit size of $(p^1, \ldots, p^n, \varepsilon)$. Then for each signature $\sigma \in \Sigma$, let $Y[\sigma] = \min\{p(S) \mid sig(S) = \sigma\}$; we use the convention that $\min \emptyset = +\infty$, i.e., if there are no sets with signature σ, then $Y[\sigma] = +\infty$. One can think of $Y[\sigma]$ as a measure of maximal "unhappiness" of sets whose signature is σ. The quantities $Y[\sigma]$, $\sigma \in \Sigma$, are easy to compute by dynamic programming. Specifically, for every $j = 1, \ldots, n$ we let $Y[\sigma; j] = \min\{p(S) \mid sig(S) = \sigma, S \subseteq \{1, \ldots, j\}\}$. We have $Y[\sigma; 1] = p^1$ if $\sigma = sig(\{1\})$ and $Y[\sigma; 1] = +\infty$ otherwise. Further, it is not hard to see that for $j > 1$ we have $Y[\sigma; j] = \min\{Y[\sigma; j-1], Y[\sigma - sig(\{j\}); j-1] + p^j\}$ if $\sigma - sig(\{j\}) \in \Sigma$ and $Y[\sigma; j] = Y[\sigma; j-1]$ otherwise. Finally, we have $Y[\sigma] = Y[\sigma; n]$. The running time of this procedure is polynomial in n, $|\Sigma|$, and the bit size of $(p^1, \ldots, p^n, \varepsilon)$.

Having computed $Y[\sigma]$ for all $\sigma \in \Sigma$, we can check for each $\sigma \in \Sigma$ whether $\sigma(t) \geq q(t)$. This requires evaluating two polynomials of degree k, which can be done efficiently. Let $\Sigma^+(t) = \{\sigma \in \Sigma \mid \sigma(t) \geq q(t)\}$; this is the set of signatures that correspond to coalitions that are winning at time t. Therefore, $LP(t)$ contains a constraint that corresponds to a coalition $S \subseteq N$ if and only if $sig(S) \in \Sigma^+(t)$. We then compute $y(t) = \min_{\sigma\in\Sigma^+(t)} Y[\sigma]$; this requires $\mathcal{O}(|\Sigma|)$ comparisons of rational numbers whose bit size is polynomial in that of $(p^1, \ldots, p^n, \varepsilon)$.

Note that $y(t)$ is the minimal payoff that *any* winning set gets at time t; therefore, if $y(t) \geq 1-\varepsilon$, then every coalition with signature in $\Sigma^+(t)$ (i.e., every coalition that is winning in $\mathcal{G}(t)$) is paid at least $1-\varepsilon$, so we output "yes". Otherwise, we identify a coalition S with $p(S) = y(t)$ (this can be done using standard dynamic programming techniques, i.e., keeping track of a signature σ^* such that $Y[\sigma] = y(t)$, and some set S^* such that $sig(S^*) = \sigma^*$) and output it. Clearly, such a coalition corresponds to a violated constraint of $LP(t)$. Thus, this algorithm correctly implements a separation oracle for $LP(t)$. Also, its running time is polynomial in n^k, W^k, $||t||$, and the bit size of $(p^1, \ldots, p^n, \varepsilon)$. Hence, we can solve $LP(t)$ (i.e., compute $\varepsilon^*(\mathcal{G}(t))$) in time $\mathrm{poly}(n^k, W^k, ||t||)$. □

THEOREM 5.2. *An instance of* INT-LEASTCORE *given by a dynamic weighted voting game* $\mathcal{G}(\cdot) = \langle N, \mathbf{w}(\cdot), q(\cdot), t_0, t_1\rangle$, *and a pair of points* $L, U \in \mathbb{Q}$, $L < U$, *can be solved in time* $\mathrm{poly}(n^k, W^k, ||L||, ||U||, ||t_0||, ||t_1||)$.

PROOF. The argument is similar to the one used in the proof of Theorem 4.2: we show that $[t_0, t_1]$ can be divided into a finite number of intervals so that the value of the least core remains constant on every such interval and changes in an easy-to-describe way as we move from one interval to the other.

In more detail, given a signature $\sigma \in \Sigma$, let $f_\sigma(t) = \sigma(t) - q(t)$ and let R_σ be the set of non-negative real roots of $f_\sigma(\cdot)$. Let $R = \cup_{\sigma\in\Sigma} R^\sigma$ and set $\rho = |R|$; we have $\rho \leq 2k|\Sigma| = 2k(2nW+1)^{k+1}$. Renumber the points in R so that $R = \{r_1, \ldots, r_\rho\}$, $r_1 \leq \cdots \leq r_\rho$, and let $I_j = (r_j, r_{j+1})$ for $j = 1, \ldots, \rho-1$; also, set $I_0 = (0, r_1)$, $I_\rho = (r_\rho, +\infty)$. For each $j = 0, \ldots, \rho$ the set of winning coalitions is the same for all games $\mathcal{G}(t)$ with $t \in I_j$; consequently, $\varepsilon^*(\mathcal{G}(t))$ is constant on I_j.

Just as in the proof of Theorem 4.2, we can describe how $\varepsilon^*(\mathcal{G}(\cdot))$ changes at a point $r_j \in R$, $j = 1, \ldots, \rho$. Pick some $r_j^- \in I_{j-1}$ and $r_j^+ \in I_j$. The signatures in Σ can be classified into the following three categories with respect to j:

- $\Sigma_j^\perp = \{\sigma \mid f_\sigma(r_j) \neq 0\}$;
- $\Sigma_j^\uparrow = \{\sigma \mid f_\sigma(r_j) = 0, f_\sigma(r_j^+) > 0, f_\sigma(r_j^-) < 0\}$;
- $\Sigma_j^\downarrow = \{\sigma \mid f_\sigma(r_j) = 0, f_\sigma(r_j^+) < 0, f_\sigma(r_j^-) > 0\}$.

Moreover, for any signature $\sigma \in \Sigma$ it is easy to determine which of these categories it belongs to: it suffices to compute the sign of $f_\sigma(\cdot)$ and all of its derivatives at $t = r_j$.

Recall that we denote by $\Sigma^+(t)$ the set of all signatures that correspond to coalitions that are winning in $\mathcal{G}(t)$ (cf. the proof of Theorem 5.1). We have

$$\Sigma^+(r_j) = \Sigma^+(r_j^-) \cup \Sigma_j^\uparrow$$

and

$$\Sigma^+(r) = \Sigma^+(r_j) \setminus \Sigma_j^\downarrow \quad \text{for all } r \in I_j.$$

Note that $\Sigma^+(r_j^-)$ contains $\Sigma_j^\downarrow$ and $\Sigma^+(r) = \Sigma^+(r_j) \setminus \Sigma_j^\downarrow$ for all $r \in I_j$. Further, we have shown in the proof of Theorem 5.1 that we can efficiently compute $\varepsilon^*(\mathcal{G}(t))$ as long as we know $\Sigma^+(t)$. Thus, we can now proceed as in the proof of Theorem 4.2: we start at $t = 0$, compute $\Sigma^+(0)$ and $\varepsilon^*(\mathcal{G}(0))$, and then process the points in R one by one, recomputing Σ^+ and the value of the least core at each point

r_j and for each interval I_j, $j = 1, \ldots, \rho$. Further, we can avoid computing the set R explicitly; this can be shown in the same way as in the proof of Theorem 4.2. $\square$

Theorems 5.1 and 5.2 can be extended to Point-CoS and Int-CoS. To see this, note that the cost of stability of a weighted voting game $\mathcal{G}(t)$ can be obtained as a solution to a linear program that is very similar to $LP(t)$: we replace constraints (6) and (7) with

$$\sum_{i \in N} p^i = 1 + \varepsilon$$

and

$$\sum_{i \in S} p^i \geq 1 \text{ for each } S \subseteq N \text{ s. t. } \sum_{i \in S} w^i \geq q,$$

respectively. It is easy to see that the arguments in the proofs of Theorems 5.1 and 5.2 apply to this linear program *mutatis mutandis*.

6. EMPIRICAL RESULTS

In this section we use a modified version of our algorithm for Int-Shapley to analyze the power distribution in the European Union over the next 50 years.

Recall that decision-making in the Council of the European Union (the EU Council) under the Treaty of Lisbon can be modeled by a 27-player vector weighted voting game of dimension 2 (see Example 2.1). One of the weight vectors in this game corresponds to the states' population, and therefore this game evolves over time. While we presented our model and results for weighted voting games rather than for vector weighted voting games of arbitrary dimension, it is easy to see that both the model itself and the algorithms described in Sections 4 and 5 can be extended to vector weighted voting games; moreover, our algorithms still run in pseudopolynomial time if the dimension of the game is bounded by a constant. Thus, the framework proposed in this paper can be used to analyze the changes in the members' voting power.

Of course, we do not know for sure what the population of each member state will be in the future. However, we can use statistical data to extrapolate the current demographic trends. In our experiments, we have used statistical data on the EU states' population from the last decade[1] in order to predict the populations of EU member states via linear regression analysis. We have generated linear estimates of population growth for each of the 27 member states[2]. That is, for each state we have obtained a polynomial of degree 1 describing its population growth. We have normalized the weight of each country to percentiles with a single decimal point (i.e. 33.3%), for ease of computation.

We have extended the algorithm described in Theorem 4.2 to vector weighted voting games and implemented it in C++. However, instead of invoking the dynamic programming algorithm, i.e., enumerating all possible signatures, we simply enumerated all possible coalitions: for 27 players and weights that can take 1000 different values $(.1, \ldots, 99.9, 100)$ the latter approach turns out to be more efficient. We emphasize, however, that the dynamic programming-based approach would also be feasible and scales better with the number of players, so it is plausible that it would have been a better choice for a larger number of players.

	Max	Min
Austria	0.0199584 (2061)	0.0182570 (2012)
Belgium	0.0259888 (2061)	0.0222924 (2013)
Bulgaria	**0.0167102 (2014)**	**0.0106988 (2055.5)**
Cyprus	0.0075948 (2059)	0.0064017 (2020.5)
Czech Republic	0.0218486 (2039)	0.0210618 (2039.5)
Denmark	0.0149227 (2061)	0.0142745 (2012)
Estonia	0.0075948 (2059)	0.0071847 (2020.5)
Finland	0.0140850 (2061)	0.0134919 (2012)
France	0.1294509 (2060)	0.1200631 (2012)
Germany	**0.1585939 (2012)**	**0.1245375 (2060.5)**
Greece	0.0242410 (2061)	0.0231040 (2013)
Hungary	0.0208000 (2020)	0.0181716 (2054)
Ireland	**0.0165897 (2061)**	**0.0119190 (2012)**
Italy	0.1162729 (2056)	0.1101284 (2012.5)
Latvia	0.0088840 (2014)	0.0074231 (2049)
Lithuania	0.0104944 (2018)	0.0089559 (2043)
Luxemburg	0.0068000 (2059)	0.0064017 (2020.5)
Malta	0.0059956 (2059)	0.0056207 (2020.5)
Netherlands	0.0340808 (2060)	0.0320960 (2013)
Poland	0.0700801 (2015)	0.0654310 (2059.5)
Portugal	0.0233734 (2061)	0.0214740 (2013.5)
Romania	0.0399632 (2012)	0.0286066 (2059.5)
Slovakia	0.0137573 (2037)	0.0129451 (2039.5)
Slovenia	0.0089034 (2018)	0.0080772 (2028)
Spain	**0.1079667 (2061)**	**0.0822549 (2012)**
Sweden	0.0216593 (2061)	0.0198683 (2013)
UK	0.1179877 (2053)	0.1151077 (2014)

Table 1: The maximal and minimal Shapley values of all EU states, based on a linear population growth estimate. The number in brackets denotes the year at which the minimum (or maximum) is reached.

It turns out that even when we round the weights to obtain a more tractable instance, the number of time steps in which a phase transition occurs remains quite large, well over tens of thousands. Phase transitions seem to be fairly evenly dispersed over time. In other words, even under a highly simplified model of population dynamics in the EU, power shifts occur very often. One may argue that, in practice, a country's weight only changes after a population census has been conducted, but it is plausible that in the future population registries will be maintained electronically, and daily updates of the voting weights would become feasible, making frequent power shifts a reality.

In Table 1 we have used biannual measurements to obtain predictions on the Shapley value of member states. As Table 1 shows, it is impossible for any member state to become a dummy player; however, there are countries whose voting power changes significantly over the years. For instance, Ireland and Spain enjoy an increase of $\sim 39\%$ and $\sim 31\%$ in voting power, respectively, in 50 years' time; Bulgaria and Germany on the other hand, stand to lose $\sim 35\%$ and $\sim 22\%$

[1]Data was obtained from the European Commission Department of Statistics (Eurostat) [1].

[2]Using linear estimates also means that we do not have to deal with irrational numbers—see the proof of Theorem 4.2.

of their voting power, respectively (see Table 2). This shift in voting power displaces Germany from its position as the most influential member state in the EU Council, making France the player with the highest Shapley value.

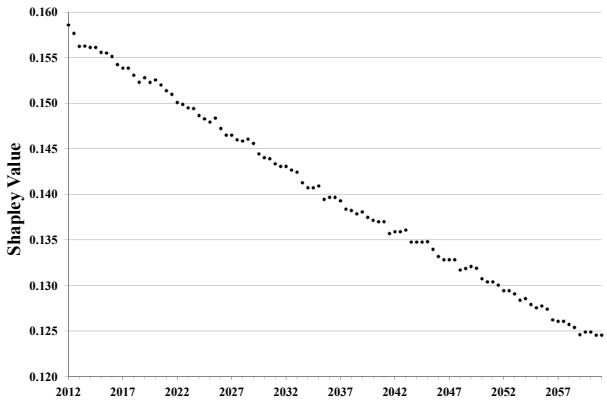

Table 2: The change in Shapley value for Germany, using a linear estimate for population size, projecting 50 years ahead.

Our experiments demonstrate that changes in power occur often, and the power distribution can shift dramatically in a short period of time, especially in the presence of states that undergo dramatic population increases. For example, should the state of Turkey join the European Union, this is likely to result in dramatic shifts in voting power, not only due to Turkey being a large state, but also because of its relatively high population growth rate.

We remark that the issue of power shifts in the EU Council of Members under the Treaty of Lisbon has been recently investigated by Kóczy [11]. However, in contrast with our work, Kóczy simply uses existing software and population estimates for computing the Shapley values at several points in the future. His results are broadly similar to ours; the discrepancies can be explained by using different methods for predicting the population of the member states.

7. CONCLUSIONS AND FUTURE WORK

We have introduced the notion of dynamic weighted voting games, which model weighted voting games that evolve over time. We developed algorithms for answering natural computational questions about such games, and applied them to understand the dynamics of the Council of the European Union over the next 50 years. It would be interesting to see if we can use similar methods to analyze other classes of coalitional games where the characteristic function may change over time, such as, for instance, network flow games [10] where the arc capacity may go up or down as the time passes, or matching games with evolving edge values.

Acknowledgements Edith Elkind was supported by National Research Foundation (Singapore) under grant RF2009-08. Dmitrii Pasechnik was supported by Singapore MOE Tier 2 Grant MOE2011-T2-1-090 (ARC 19/11). Yair Zick was supported by SINGA graduate fellowship. The first two authors would like to thank their son Yasha, who was born 6 weeks before the AAMAS submission deadline, for being cooperative.

8. REFERENCES

[1] Statistical database of the European Union: Population — Demography. http://epp.eurostat.ec.europa.eu/portal/page/portal/population/data/main_tables.

[2] *Consolidated versions of the Treaty on European Union and the Treaty on the functioning of the European Union*. Publications Office of the European Union, Luxembourg, 2010.

[3] E. L. Allgower and K. Georg. *Introduction to numerical continuation methods*, volume 45 of *Classics in Applied Mathematics*. SIAM, Philadelphia, PA, 2003.

[4] Y. Bachrach, E. Elkind, R. Meir, D. Pasechnik, M. Zuckerman, J. Rothe, and J. S. Rosenschein. The cost of stability in coalitional games. In *SAGT'09*, pages 122–134, 2009.

[5] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in real algebraic geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, second edition, 2006.

[6] Y. Bugeaud and M. Mignotte. Polynomial root separation. *International Journal of Number Theory*, 6(3):587–602, 2010.

[7] G. Chalkiadakis, E. Elkind, and M. Wooldridge. *Computational Aspects of Cooperative Game Theory*. Morgan and Claypool, 2011.

[8] 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.

[9] D. Felsenthal and M. Machover. Ternary voting games. *International Journal of Game Theory*, 26(3):335–351, 1997.

[10] E. Kalai and E. Zemel. Totally balanced games and games of flow. *Mathematics of Operations Research*, 7(3):476–478, 1982.

[11] L. A. Kóczy. Beyond Lisbon: Demographic trends and voting power in the European Union Council of Ministers. *Mathematical Social Sciences*, 63(2):152–158, 2012.

[12] D. Leech. Designing the voting system for the Council of the European Union. *Public Choice*, 113(3):437–464, 2002.

[13] M. Maschler, B. Peleg, and L. S. Shapley. Geometric properties of the kernel, nucleolus, and related solution concepts. *Mathematics of Operations Research*, 4(4):303–338, 1979.

[14] T. Matsui and Y. Matsui. A survey of algorithms for calculating power indices of weighted majority games. *Journal of the Operations Research Society of Japan*, 43(1):71–86, 2000.

[15] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, Chichester, 1986.

[16] L. Shapley and M. Shubik. A method for evaluating the distribution of power in a committee system. *The American Political Science Review*, 48(3):787–792, 1954.

[17] L. S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games, volume II*, pages 307–317. Princeton University Press, 1953.