Enumeration and exact design of weighted voting games

Bart de Keijzer*
CWI
Amsterdam, The Netherlands
B.de.Keijzer@cwi.nl

Tomas Klos
Delft University of Technology
Delft, The Netherlands
T.B.Klos@tudelft.nl

Yingqian Zhang
Delft University of Technology
Delft, The Netherlands
Yingqian.Zhang@tudelft.nl

ABSTRACT

In many multiagent settings, situations arise in which agents must collectively make decisions while not every agent is supposed to have an equal amount of influence in the outcome of such a decision. Weighted voting games are often used to deal with these situations. The amount of influence that an agent has in a weighted voting game can be measured by means of various power indices.

This paper studies the problem of finding a weighted voting game in which the distribution of the influence among the agents is as close as possible to a given target value. We propose a method to exactly solve this problem. This method relies on a new efficient procedure for enumerating weighted voting games of a fixed number of agents.

The enumeration algorithm we propose works by exploiting the properties of a specific partial order over the class of weighted voting games. The algorithm enumerates weighted voting games of a fixed number of agents in time exponential in the number of agents, and polynomial in the number of games output. As a consequence we obtain an exact anytime algorithm for designing weighted voting games.

Categories and Subject Descriptors

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

General Terms

Algorithms, Theory

Keywords

weighted voting games, simple games, power index

1. INTRODUCTION

In many real-world problems that involve multiple agents, for instance elections, there is a need for fair decision making protocols, in which different agents have different amounts of influence in the outcome of a decision. *Weighted voting games* (WVGs) are often used in these decision making protocols. In a WVG, some quota is given, and each agent in the game has a certain weight. If the total weight of a coalition of agents exceeds the quota, then that coalition is said to be *winning*. In order to measure an agent's *a priori* power in such WVGs, the notion of a *power index* arose. A lot of research has been done in cooperative game theory on how to compute various power indices efficiently.

In this paper, instead of analyzing the power of each agent in a voting game, we investigate the so-called "inverse problem". More specifically, given a target power index for each of the agents, we study how to *design a weighted voting game* for which the power of each agent is as close as possible to the given power index. Only very little work is known which tries to solve this inverse problem [6, 2]. These algorithms are local search methods that do not guarantee an optimal answer. No (optimal) algorithm is known for generating the exact game. Such an algorithm to solve the inverse problem exactly is the topic of the current paper.

The most straightforward approach to solve the inverse problem would be to simply enumerate *all possible* WVGs of n agents, and to compute for each of these weighted voting games its power index. We can then output the game of which the power index is the closest to the given one. Unfortunately, it turns out that enumerating all weighted voting games is not trivial.

Our results.

We first demonstrate that there exists an *infinite* number of weighted representations for each WVG (Proposition 1). Hence, it seems not that surprising that no algorithm has been developed to compute the exact WVG.

We then approach the inverse problem by devising an enumeration method that generates every voting game relatively efficiently. First, we devise a "naive" method that enumerates all WVGs in doubly exponential time (Section 4). Subsequently, we improve on this runtime exponentially by showing how to enumerate all WVGs within exponential time (Section 5). Although the runtime of this enumeration method is still exponential, we will see that the algorithm for the inverse problem that results from this has the *anytime* property: the longer we run it, the better the result becomes. Also, we are guaranteed that the algorithm eventually finds the *optimal* answer. Our enumeration method is based on exploiting a new specific partial order on the class of WVGs. This partial order can be considered interesting in its own right, from a game-theoretical point of view.

The problem that we attempt to solve is a specific case

*The paper was written while this author was a student at Delft University of Technology.

Cite as: Enumeration and exact design of weighted voting games, Bart de Keijzer, Tomas Klos and Yingqian Zhang, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

of the more general class of problems, where we are concerned with finding a weighted voting game for which some property or some set of properties is satisfied. We refer to such problems as *weighted voting game design problems*. Our method for solving the inverse problem is simply an enumeration method, so as a consequence the same method can in fact be used to also solve any other weighted voting game design problem.

We start with some definitions and notations that we use in this paper.

2. PRELIMINARIES

A *coalitional game* is a pair (A, v), where $A = \{a_1, \ldots, a_n\}$ is a set of agents, and $v : 2^A \to \mathbb{R}^+ \cup \{0\}$ is a characteristic (or gain) function that specifies how much collective payoff each coalition of agents can gain. Subsets of A are referred to as *coalitions*, and A is called the *grand coalition*. A *simple* coalitional game (or *simple game*) is a coalitional game (A, v) where v is restricted to $\{0, 1\}$. A coalition $S \subseteq A$ is called a *winning* coalition if $v(S) = 1$, and if $v(S) = 0$, then S is called a *losing* coalition. In this paper, we concentrate on simple games.

An important subclass of simple games are *monotone games*. A game (A, v) is *monotone* if and only if $\forall (S, T) \in (2^A)^2 : S \subset T \to v(S) \leq v(T)$. This means that a superset of a winning coalition is always winning. Another important type of simple game is the *weighted voting game* (WVG), which has a *weighted form* (W, q), where $W = (w_1, \ldots w_{|A|}) \in \mathbb{R}^{+|A|}$ defines the weights of agents, and $q \in \mathbb{R}^+$ is called the quota. For any coalition S in A, if S wins, it implies the total weight that occurs beforof the agents in S is not smaller than the quota, and formally, $\forall S \subseteq N : v(S) = 1$ if and only if $w(S) = \sum_{i \in S} w_i \geq q$. Note that every WVG is monotone, but not every monotone game is a WVG.

For simple games, we can define the following *desirability relation* $\preceq_D$ over the agents. We say that an agent $i \in A$ and another agent $j \in A$ are *equally desirable*, denoted by $i \sim_D j$, if $\forall S \subseteq A \backslash \{i, j\} : v(S \cup \{i\}) = v(S \cup \{j\})$. We say i is *more desirable* than j (denoted $i \succeq_D j$) if $\forall S \subseteq A \backslash \{i, j\} : v(S \cup \{i\}) \geq v(S \cup \{j\})$. If $i \succeq_D j$ and not $i \sim_D j$, we say i is *strictly* more desirable than j ($i \succ_D j$). Moreover, we say that i and j are *incomparable* when neither $i \succeq_D j$ or $j \succeq_D i$. If in a simple game (A, v), no pair of agents in A is incomparable with respect to $\succeq_D$, we call this game a *linear* game.

Clearly, if a game is weighted, then the desirability relation over the agents in that game is complete. So we obtain the simple consequence that the set of WVGs is a subset of the set of linear games. This brings us the following two special classes of games. We call a linear game $G = (N, v)$ a *canonical linear game* whenever $N = \{1, \ldots, n\}$ and the desirability relation $\succeq_D$ satisfies $1 \succeq_D \cdots \succeq_D n$. When G is also weighted, then G is a *canonical weighted voting game*. Note that a WVG is canonical if and only if its weight vector is non-increasing.

We now introduce some notations used in this paper. We will use the following symbols to denote specific classes of games: $\mathcal{G}_{\mathsf{mon}}$ denotes the class of monotone simple games; $\mathcal{G}_{\mathsf{lin}}$ denotes the class of linear games; $\mathcal{G}_{\mathsf{wvg}}$ and $\mathcal{G}_{\mathsf{cwvg}}$ denote the class of WVGs and canonical WVGs, respectively. In addition, for a class of games $\mathcal{G}$, we use $\mathcal{G}(n)$ to denote the class of games restricted to the set of agents $\{1, \ldots, n\}$, i.e., $\mathcal{G}(n) = \{G \mid G \in \mathcal{G} \wedge G = (\{1, \ldots, n\}, v)\}$.

There are various ways to represent coalitional simple games. The *minimal winning coalition form* is the pair $(A, W_{\min})$, where $W_{\min}$ is the set of minimal winning coalitions of (A, v), i.e., $\forall S \subseteq A : v(S) = 1 \to (\exists T \in W_{\min} : S \supseteq T)$. Similarly, $(A, L_{\max})$ is the *maximal losing coalition form* if $L_{\max}$ is the set of maximal losing coalitions of the game (A, v), i.e., $\forall S \subseteq A : v(S) = 0 \to (\exists T \in L_{\max} : S \subseteq T)$. Note that a simple game can be represented in maximal winning (or losing) coalition form if and only if it is monotone.

We now define various *representation languages* for representing simple games. We will use the following three: $\mathcal{L}_{\mathsf{weights}}$ consits of encodings of lists of numbers $\langle w_1, \ldots, w_n, q \rangle$, representing a WVG with weighted form $((w_1, \ldots, w_n), q)$; $\mathcal{L}_{W,\min}$ ($\mathcal{L}_{L,\max}$) consists of encodings of lists of minimal winning (maximal losing) coalitions, representing the minimal winning form of a game. When ℓ is a string from some representation language, we will write G_ℓ to denote the game that is described by ℓ.

We now introduce an important measure in voting games, i.e., *power indices*. Power indices measures the amount of influence that an agent has in a simple game. Power indices originally were introduced because it was observed that in WVGs, the weight of an agent is not directly proportional to the influence he has in the game. This can be easily observed from the following simple example.

EXAMPLE 1. *Given a WVG* $\left(W, q = \sum_{w \in W} w\right)$ *with only one winning coalition, i.e., the grand coalition where every agent is in. So no matter what the weights of the agents are, they all have the same power to decide whether the coalition wins or not.*

Various power indices have been proposed in order to describe (as a number between 0 and 1) the 'true' influence that an agent has in a WVG, among which the Shapley-Shubik index [14] and the Banzhaf index [3] are by far the most popular two. Computing an agent's Banzhaf or Shapley-Shubik index is known to be #P-complete [5, 13].

In this paper, we study how to design a voting game such that its power index is as close as possible to a given target power index. We thus define a *voting game design problem* as an optimization problem.

Definition 1. Let $\mathcal{G}$ be a class of simple games, let $\mathcal{L}$ be a representation language for $\mathcal{G}$, and let f be a function such that $f(i, G)$ returns a specific power index (e.g. the Banzhaf index) for player i in game G. The *$(f, g, \mathcal{G}, \mathcal{L})$-power index voting game design problem*, or *$(f, g, \mathcal{G}, \mathcal{L})$-PVGD* problem, is the problem where we are given a vector $(p_1, \ldots, p_n) \in [0, 1]^n$ and we need to find an $\ell \in \mathcal{L}$ such that $G_\ell \in \mathcal{G}(n)$, and the error measure $g(G_\ell, f)$ is minimized.

In words, in a $(f, g, \mathcal{G}, \mathcal{L})$-PVGD problem we must find a game in the class $\mathcal{G}$ that is as close as possible to a given target power index $(p_1, \ldots, p_n)$ according to power index function f and error measure g. In this paper, we will be using the sum-of-squared-errors measure, i.e. $g(G_\ell, f) = \sum_{i=1}^{n} (f(i, G_\ell) - p_i)^2$ We can analyze this problem for various power index functions, classes of games, and representation languages. A particularly interesting case that we will focus on is the problem $(f, g, \mathcal{G}_{\mathsf{wvg}}, \mathcal{L}_{\mathsf{weights}})$-PVGD, i.e., the problem of finding a WVG in weighted representation, that is as close as possible to a certain target power index.

3. RELATED WORK

Although some specific variants of $(f, g, \mathcal{G}, \mathcal{L})$-PVGD problems are mentioned sporadically in the literature, not many attempts to solve this problem are known to us. Often, the problem is referred to as *the inverse problem*. To the best of our knowledge, only a few papers propose algorithms for $((f, g, \mathcal{G}, \mathcal{L})$-PVGD). One of them is by Fatima et al. for the case of the Shapley-Shubik index [6], i.e., the problem of finding a weighted voting game in weighted representation for the Shapley-Shubik index. This algorithm works by repeatedly updating a vector of weights using one of two update rules, of which the authors prove that by applying the rule, the Shapley-Shubik index of each agent cannot get worse. Hence, it is an *anytime* algorithm.

Another attempt is by Aziz et al. for finding a WVG for the Banzhaf index [2]. The approach resembles that in [6], in that their algorithm repeatedly updates the weight vector in order to get closer to the target power index. Contrary to Fatima et al.'s method, the algorithm always outputs an integer weighted representation, because the *generating function method* [4] the authors use only works on integer weights. It is not known whether Aziz's algorithm always converges, so it is not certain whether this method is also *anytime*. Also, no approximation guarantee is given.

Leech proposes in [10] an approach that is the same as Aziz et al.'s, but uses a different updating rule. The focus in this paper is on the results that are obtained after applying the method to the 15-member EU council, and to the board of governors of the International Monetary Fund. Because the three approaches that we just discussed do not give an analysis of various aspects of the methods they use, we consider these algorithms to be essentially heuristic methods.

Some applied work has also been done on the design of voting games. Papers [9] and [16] analyze and design the distribution of voting power in the European Union, using iterative methods that resemble the algorithm of Aziz [2].

Lastly, Alon and Edelman argue in [1] that there is a need for *a priori* estimates of what power indices are achievable in simple games, in order to analyze the accuracy of these kinds of iterative algorithms: there is a need for information about the distribution of power indices in $[0, 1]^n$. As a first step into solving this problem, they prove in [1] a specific result for the case of the Banzhaf indices of monotone games.

4. A NAIVE ALGORITHM FOR PVGD

The current methods that exist for solving PVGD problems are all hill climbing methods that are based on repeatedly adjusting the weight vector according to some specific heuristic rules. As of yet, no attempt has been made at creating an algorithm that exactly solves $(f, g, \mathcal{G}_{\mathsf{wvg}}, \mathcal{L}_{\mathsf{weights}})$-PVGD problems (for any power index f).

How such an algorithm would work is not immediately obvious. To begin with, it is not even obvious whether the following related decision problem is decidable: We are given as input a vector u of n numbers between 0 and 1, and the question is whether there exists a weighted voting game that has u as its power index (for some choice of power index).

The most straightforward approach to solve a $(f, g, \mathcal{G}_{\mathsf{wvg}}, \mathcal{L}_{\mathsf{weights}})$-PVGD problem, on input $u = (u_1, \ldots, u_n)$ would be to simply enumerate all possible weighted voting games of n agents, and to compute for each of these weighted voting games its power index. We then output the game of which the power index is as close as possible to u. So in this case, the problem becomes one of enumerating all games in $\mathcal{G}_{\mathsf{wvg}}(n)$. However, doing this is not as simple as one might think, as we can not work in a direct manner with *weighted representations* of games. As an example, consider an algorithm that works directly with weighted representations as follows: the algorithm outputs weighted voting games by enumerating all integer (weight vector, quota)-pairs, and outputs such a pair when it is a representation of a game that it has not encountered before. We know that this algorithm eventually outputs all weighted voting games of n players, since if there exists a real-numbered weighted representation for a game, then there exists a rational weighted representation, and by a result that we will state below (i.e. Propostion 1) we know that in that case there also exists an integer representation for that game. The problem with such an algorithm is that we do not know when to stop enumerating. Another problem is that we can not show anything about the runtime of this algorithm and the time spent between two successive outputs. Part of the reason that this approach does not work, is because there is an infinite number of weighted representations for each weighted voting game, which the following observation tells us.

PROPOSITION 1. *Let $G \in \mathcal{G}_{\mathsf{wvg}}(N)$ be a weighted voting game with $N = \{1, \ldots, n\}$, and let $\ell = ((w_1, \ldots, w_n), q) \in L_{\mathsf{weights}}$ be a weighted representation for G. For any $\lambda \in \mathbb{R}^+$, we have that $\ell' = ((\lambda w_1, \ldots, \lambda w_n), \lambda q)$ is a weighted representation for G.*

PROOF. For any coalition $C \subseteq N$ such that $w_\ell(C) < q$ we have $w_{\ell'}(C) = \sum_{i \in C} \lambda w_i = \lambda \sum_{i \in C} w_i = \lambda w_\ell(C) < \lambda q$, and for any coalition $C \subseteq N$ such that $w_\ell(C) \geq q$ we have $w_{\ell'}(C) = \sum_{i \in C} \lambda w_i = \lambda \sum_{i \in C} w_i = \lambda w_\ell(C) \geq \lambda q$. □

Hence, due to the infinite number of weighted representations, it is difficult to find an enumeration algorithm that is only based on inspecting weighted representations. We therefore will look into using other representations.

One thing that is certain, is that the number of weighted voting games on n agents is finite, contrary to the infinite number of weighted representations of weighted voting games. This is because of the simple fact that all weighted voting games are monotone games, and each monotone game can be described as a set of MWCs. There are only a finite number of such sets of MWCs. In fact, it can be seen that any set S of MWCs is an antichain[1] under $\subseteq$: No coalition $C \in S$ is a subset of another coalition $C' \in S$, because otherwise it would mean that either C is not winning or C' is not minimal winning.

For this reason, we at least have a method to enumerate all monotone games, so at least we can solve the problem $(f, g, \mathcal{G}_{\mathsf{mon}}, \mathcal{L}_{W,\min})$-PVGD exactly: consider Algorithm 1.

This algorithm is very simple and obviously correct, but its runtime is very large. It generates all antichains of coalitions. This is a very large number, and is equal to the number D_n of antichains on a set of n elements (agents, in our case). Finding an expression for D_n is a well known open problem in combinatorics, known as *Dedekind's problem*. D_n is also referred to as the *nth Dedekind number*.

Because D_n quickly grows very large (in n), line 3 is what gives the algorithm a very high time complexity. The fol-

[1]An antichain under a binary relation R is a set S such that $\forall (x, y) \in S^2 : x \neq y \rightarrow \neg R(x, y) \wedge \neg R(y, x)$.

Algorithm 1 A straightforward algorithm for solving $(f, g, \mathcal{G}_{\mathsf{mon}}, \mathcal{L}_{W,\min})$-PVGD. The input is a target power index $\vec{p} = (p_1, \ldots, p_n)$. The output is an $\ell \in \mathcal{L}_{W,\min}$ such that $f(G_\ell)$ is as close as possible to $\vec{p}$.

```
bestgame := 0 {bestgame keeps track of the best game that we have found, represented as a string in L_{W,min}.}
besterror := ∞ {besterror is the error of f(G_bestgame) from p, according to the sum-of-squared-errors measure.}
for all ℓ ∈ L_{W,min} do
    Compute f(G_ℓ) = (f(G_ℓ, 1), ..., f(G_ℓ, n)).
    error := Σ_{i=1}^{n} (f(G_ℓ, i) − p_i)^2.
    if error < besterror then
        bestgame := ℓ
        besterror := error
return bestgame
```

lowing bounds are known for D_n [8]:

$$2^{(1+c'\frac{\log n}{n})E_n} \geq D_n \geq 2^{(1+c2^{-n/2})E_n}, \tag{1}$$

WVG where c' and c are constants and E_n is the size of the largest antichain on an n-set. Sperner's theorem states [15]: $E_n = \binom{n}{\lfloor \frac{n}{2} \rfloor}$. From Sperner's theorem and Stirling's approximation of the factorial function, we get

$$E_n \in \Theta\left(\frac{2^n}{\sqrt{n}}\right). \tag{2}$$

A possible approach to enumerate antichains would be to simply enumerate each set of coalitions, and check if that set is an antichain. In total, there are 2^{2^n} families of coalitions. Now let us suppose that D_n equals the upper bound of (1). Substituting the tight bound of (2) into the upper bound of (1), we get $D_n \leq 2^{(1+c'\frac{\log n}{n})k\frac{2^n}{\sqrt{n}}}$ for some constants k and c'. We then see that

$$\sqrt[\frac{\sqrt{n}}{(1+c'\frac{\log n}{n})k}]{D_n} \leq 2^{2^n}.$$

This means that the number of all families of subsets on an n-set is exponential in n relative to the Dedekind number. Hence, the Dedekind number is super-exponential but sub-doubly-exponential in n. We will not explore this enumeration problem any further, for the reason that even an efficient enumeration method for antichains would not result in any practically applicable version of Algorithm 1, simply because the Dedekind number is very large.

We conclude that Algorithm 1 achieves a running time in $O^*(2^{2^{n(1+\epsilon)}})$ for any $\epsilon > 0$,[2] under the condition that the computation of the power index f does not take super-exponential time.[3]

Now that we have an exact algorithm for $(f, g, \mathcal{G}_{\mathsf{mon}}, \mathcal{L}_{W,\min})$-PVGD, it follows from the following result given in [12] that there is also an exact algorithm for $(f, g, \mathcal{G}_{\mathsf{wvg}}, \mathcal{L}_{\mathsf{weights}})$-PVGD.

THEOREM 2. *There exists an $O(nm^2 + n^3m)$ time algorithm that decides whether a weighted representation exists for a monotone game described by its list of MWCs, and outputs a weighted representation if it exists. Here, m is the amount of MWCs of the input game. (Moreover, the algorithm can also be used to output the list of maximal losing coalitions of a monotone game.)*

Proof sketch. The polynomial time algorithm that does this, roughly works as follows: on input S the algorithm first decides whether the game G_S described by S is a linear game. If so, the algorithm enumerates all of the maximal losing coalitions of G_S in polynomial time. Finally the algorithm finds a weighting by solving a linear program that ensures that the total weight of each maximal losing coalition is strictly lower than the total weight of each MWC. □

By this result, we can adapt algorithm 1 such that it tries to generate a weighted representation for each monotone game that it finds. If it then turns out that a game is not weighted, we skip that game. Now we arrive at the following corollary.

COROLLARY 3. *There exists an exact algorithm for $(f, g, \mathcal{G}_{\mathsf{wvg}}, \mathcal{L}_{\mathsf{weights}})$-PVGD that runs in time $O^*(2^{2^{n(1+\epsilon)}})$.*

So we finally have derived an algorithm for $(f, g, \mathcal{G}_{\mathsf{wvg}}, \mathcal{L}_{\mathsf{weights}})$-PVGD that terminates within a finite amount of time. The problem is still that the runtime of this algorithm is large.

5. AN IMPROVED EXACT ALGORITHM FOR PVGD

In this section, we will show that we can do much better than the naive approach. For the case of WVGs, we can improve the enumeration method exponentially, so that it runs in only singly exponential time. We will also be able to prove that this new enumeration method runs in output-polynomial time, i.e., a polynomial in the size of the input and the size of the output. Furthermore, the resulting exact algorithm for solving $(f, g, \mathcal{G}_{\mathsf{wvg}}, \mathcal{L}_{\mathsf{weights}})$-PVGD will be an *anytime algorithm.*

5.1 A partial order on the class of WVGs

Let us now develop the necessary theory behind the algorithm that we will propose. We will focus only on the class of canonical WVGs, since for each non-canonical weighted voting game there is a canonical one that can be obtained by merely permuting the agents. We first need some definitions.

Definition 2. A *poset* or *partially ordered set* is a set S equiped with a partial order $\preceq$, i.e., a pair $(S, \preceq)$. A poset $(S, \preceq)$ is *graded* when there exists a *rank function* $\rho : S \to \mathbb{N}$ such that for any pair $(x, y) \in S^2$ it is true that $\rho(x) = \rho(y) - 1$ whenever x covers y in the poset $(S, \preceq)$. We say that x *covers* y in $(S, \preceq)$ when $x \preceq y$ and there is no $z \in S$ such that $x \preceq z \preceq y$. A *least element* of a poset $(S, \preceq)$ is an element $x \in S$ such that $x \preceq y$ for all $y \in S$.

The algorithm we will propose is based on a new structural property that allows us to enumerate the class of canonical WVGs efficiently: We will define a new relation $\trianglerighteq_{\mathsf{MWC}}$ and we will prove that for any number of agents n the class $\mathcal{G}_{\mathsf{cwvg}}(n)$ forms a graded poset with a least element under this relation.

Definition 3. Let G_1 and G_2 be any two monotone games. Let $W_{\min,1}$ and $W_{\min,2}$ be their respective sets of MWCs. Then, $G_1 \trianglerighteq_{\mathsf{MWC}} G_2$ if and only if $W_{\min,1} \supseteq W_{\min,2}$.

[2] We use the O^* notation to disregard polynomial factors, i.e., $f(x) \in O^*(g(x)) \to f(x) \in O(g(x)p(x))$ for some polynomial p.

[3] Fortunately, all power indices that we know of are known to be computable within exponential time.

THEOREM 4. *For each n, $(\mathcal{G}_{\mathsf{cwvg}}(n), \trianglerighteq_{\mathsf{MWC}})$ is a graded poset with rank function*

$$\begin{array}{rccc} \rho & : & \mathcal{G}_{\mathsf{wvg}}(n) & \rightarrow & \mathbb{Z} \\ & & G & \mapsto & |W_{\min}(G)|, \end{array}$$

where $W_{\min}(G)$ is the set of MWCs of G. $(\mathcal{G}_{\mathsf{cwvg}}(n), \trianglerighteq_{\mathsf{MWC}})$ has a least element of rank 0.

Note that the theorem above basically says: "Consider an arbitrary weighted voting game of n agents, and look at its list of MWCs. There is a MWC in this list, such that if we remove that coalition, we obtain a list of winning coalitions that represents yet another weighted voting game of n agents."

PROOF (THEOREM 4). By definition, $(\mathcal{G}_{\mathsf{cwvg}}(n), \trianglerighteq_{\mathsf{MWC}})$ is a valid poset. In order to prove that the poset is graded under the rank function ρ that is specified in the theorem, we will show by construction that

LEMMA 5. *For any game $G \in \mathcal{G}_{\mathsf{cwvg}}(n)$ with a non-empty set $W_{\min}$, there is a coalition $C \in W_{\min}$ and a game $G' \in \mathcal{G}_{\mathsf{cwvg}}(n)$ so that it holds that $W_{\min} \backslash \{C\}$ is the set of MWCs of G'.*

From Lemma 5, the remaining part of the theorem automatically follows: the game with no MWCs is the only WVG with rank 0, and clearly is the least element of the poset.

To prove Lemma 5, we first prove the following two preliminary lemmas (6 and 7).

LEMMA 6. *Let $G = (N = \{1, \ldots, n\}, v)$ be a WVG, and let $\ell = ((w_1, \ldots, w_n), q)$ be a weighted representation for G. For each agent i there exists an $\epsilon > 0$ such that $\ell' = ((w_1, \ldots, w_i + \epsilon', \ldots w_n), q)$ is a weighted representation for G for all $\epsilon' < \epsilon$.*

Informally, this lemma says that it is always possible to increase the weight of an agent by some amount without changing the game.

PROOF. For each C such that $v(C) = 1$ we have that $w_\ell(C) \geq q$, and for each C such that $v(C) = 0$ we have that $w_\ell(C) < q$. Define L_i as the set of losing coalitions containing agent i. Now consider a coalition $C \in L_i$ for which it holds that for all $C' \in L_i : w_\ell(C') \leq w_\ell(C)$.

Because $w_\ell(C) < q$, it follows that $q - w_\ell(C) > 0$. If we increase w_i in ℓ by a number strictly between 0 and $q - w_\ell(C)$ to obtain ℓ', then clearly no losing coalition in G_ℓ is a winning coalition in $G_{\ell'}$. Moreover, all winning coalitions in G_ℓ are also winning coalitions in $G_{\ell'}$. □

The following lemma states that for a WVG there exists a weighted representation such that all winning coalitions have a different weight.

LEMMA 7. *Let $G = (N = \{1, \ldots, n\}, v)$ be a WVG. There exists a weighted representation $\ell \in \mathcal{L}_{\mathsf{weights}}$ such that for all $(C, C') \in N^2, C \neq C'$ for which $v(C) = v(C') = 1$, it is true that $w_\ell(C) \neq w_\ell(C')$.*

PROOF. Let $\ell = ((w_1, \ldots, w_n), q)$ be a weighted representation for G for which there exists a $(C, C') \in (2^N)^2$ with $w_\ell(C) = w_\ell(C')$, $C \neq C'$ and $v(C) = v(C') = 1$. We will show how to obtain an ℓ' from ℓ such that $G_\ell = G_{\ell'}$ and $w_{\ell'}(C) \neq w_{\ell'}(C'')$ for any other coalition $C'' \in N$ with $v(C'') = 1$. This process can then be repeated to obtain a weighted representation for G under which the weights of all winning coalitions differ.

The procedure works as follows: it can be assumed w.l.o.g. that there is an agent i in C but not in C'. By Lemma 6, there is an $\epsilon > 0$ such that $\ell' = ((w_1, \ldots, w_i + \epsilon', w_n), q)$ is a weighted representation for G for any $0 < \epsilon' < \epsilon$. ℓ is then a weighted representation with $w_{\ell'}(C) \neq w_{\ell'}(C')$, so this *almost* proves the lemma; we must only make sure the we adjust i's weight in such a way that C's weight does not become equal to any other coalition. This can clearly be done: Consider the set of winning coalitions W_i containing agent i, and let $D \in W_i$ be a coalition such that $C \neq D$ and for all $D' \in W_i \backslash \{C\}$, we have $w_\ell(D) < w_\ell(D')$. If D exists, we make sure that $0 < \epsilon' < \min\{w_\ell(D) - w_\ell(C), \epsilon\}$, and then $w_{\ell'}(C)$ is clearly different from $w_{\ell'}(C'')$ for any $C'' \subseteq N$. $w_\ell(D) - w_\ell(C) > 0$, so this is possible. Otherwise, if D does not exist, then it suffices to take ϵ' simply strictly between 0 and ϵ. □

Using Lemma 7, we can prove Lemma 5, which establishes Theorem 4.

PROOF (LEMMA 5). Let $G = (\{1, \ldots, n\}, v)$ be a canonical WVG, $W_{\min}$ its set of MWCs and $\ell = ((w_1, \ldots, w_n), q)$ a weighted representation for which it holds that all winning coalitions have a different weight. By Lemma 7, such a representation exists. We will construct an ℓ'' from ℓ for which it holds that it is a weighted representation of a canonical WVG with $W_{\min} \backslash C$ as its list of MWCs, for some $C \in W_{\min}$.

Let i be the highest-numbered agent that is in a coalition in $W_{\min}$. Note that all players j with $j > i$ are dummy players, so for these j we assume that $w_j = 0$. Let $C \in W_{\min}$ be the MWC containing i for which it holds that $\forall C' \in W_{\min} : (C' \neq C \wedge i \in C) \rightarrow w_\ell(C') > w_\ell(C)$. Now obtain $\ell' = ((w_1, \ldots, w_i - (w_\ell(C) - q), \ldots, w_n), q)$. Clearly, $G_{\ell'} = G_\ell = G$ and $w_{\ell'}(C) = q$. Moreover, all coalitions in $W_{\min}$ that contain agent i have a different weight under ℓ'. subseteq We now decrease i's weight by an amount that is so small, that the only MWC that turns into a losing coalition is C. Note that under ℓ', coalition C is still the lightest MWC containing i. Let $C' \in W_{\min}$ then be be the second-lightest MWC containing i. Clearly, by decreasing i's weight (according to ℓ') by a positive amount smaller than $w_{\ell'}(C') - w_{\ell'}(C)$, coalition C will become a losing coalition and all other MWCs will stay winning. No new MWC is introduced in this process: suppose there would be such a new MWC S, then S contains only players that are at least as desirable as i (the other players have weight 0). But then S would also be a MWC in the original game $G_{\ell'}$, which is a contradiction. □

□

In the remainder of this text, we will make use of the notion of a *characteristic vector* of a coalition. For a coalition C, this is the n-dimensional vector for which the ith element is 1 if agent i is in the coalition, and 0 otherwise. We will abbreviate this by simply listing the elements of such a vector as a string of bits, e.g. when $n = 4$ and $S = \{1, 3\}$, then the characteristic vector of S is 1010.

EXAMPLE 2. *Figure 1 depicts $(\mathcal{G}_{\mathsf{cwvg}}(4), \trianglerighteq_{\mathsf{MWC}})$ graphically. Note that, for more convenient representation, this is not*

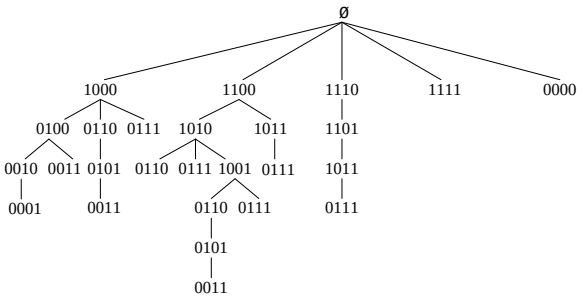

Figure 1: Graphical depiction of $(\mathcal{G}_{\mathsf{cwvg}}(4), \supseteq_{\mathsf{MWC}})$.

precisely the Hasse diagram of the poset $(\mathcal{G}_{\mathsf{cwvg}}(4), \supseteq_{\mathsf{MWC}})$. Each node in this graph represents a canonical WVG of four agents. It should be read as follows: each node has the characteristic vector of a MWC as a label. The set of MWCs of a game that corresponds to a certain node w in the graph, are those coalitions that are described by the set of vectors that are obtained by traversing the path from the top node to w. The top node corresponds to the canonical WVG with zero MWCs (i.e. every coalition loses).

A finite poset is a *tree* whenever its Hasse diagram is a tree. Next, we show that $(\mathcal{G}_{\mathsf{cwvg}}(n), \supseteq_{\mathsf{MWC}})$ is not a tree for $n \geq 4$. When we will state our algorithm in the next section, it will turn out that this fact makes things significantly more complex.

THEOREM 8. *For any $n \geq 4$, $(\mathcal{G}_{\mathsf{cwvg}}(n), \supseteq_{\mathsf{MWC}})$ is not a tree.*

PROOF. We will show an example of a game in $(\mathcal{G}_{\mathsf{cwvg}}(4), \supseteq_{\mathsf{MWC}})$ for which there are multiple games that cover it. The poset $(\mathcal{G}_{\mathsf{cwvg}}(4), \supseteq_{\mathsf{MWC}})$ is in that case not a tree. For $n > 4$ a similar example is obtained by adding dummy agents to the example that we give.[4]

Consider the following weighted representation of a canonical WVG over agents $\{1, \ldots, 4\}$: $\ell = ((3, 2, 2, 1), 4)$. The set of characteristic vectors $C_{\min,\ell}$ of MWCs of G_ℓ is as follows: $C_{\min,\ell} = \{1100, 1010, 0110, 1001\}$. Next, consider the WVGs ℓ' and ℓ'': $\ell' = ((3, 1, 1, 1), 4)$ and $\ell'' = ((1, 1, 1, 0), 2)$, with the following sets of characteristic vectors of MWCs $C_{\min,\ell'} = \{1100, 1010, 1001\}$ and $C_{\min,\ell''} = \{1100, 1010, 0110\}$, respectively. Now we see that $C_{\min,\ell'} = C_{\min,\ell} \backslash \{0110\}$ and $C_{\min,\ell'} = C_{\min,\ell} \backslash \{1001\}$. □

5.2 A fast method for enumerating WVGs

We will use the results from the previous section to develop an exponential-time exact algorithm for $(f, g, \mathcal{G}_{\mathsf{cwvg}}, \mathcal{L}_{\mathsf{weights}})$-PVGD. The way this algorithm works is very straightforward: Just as in algorithm 1, we enumerate the complete class of games (WVGs in this case), and we compute for each game the distance from the target power index.

Recall that the problem with Algorithm 1 was that the enumeration procedure is not very efficient. For the restriction to WVGs, we are able to make the enumeration procedure more efficient. We will use Theorem 4 for this: The key is that it is possible to generate the MWC listing of canonical weighted games of rank i fairly efficiently from the MWC listing of canonical WVGs of rank $i - 1$.

[4]Agents are *dummy* when they are not in any MWC.

The following theorem shows us how to do this. To state this theorem and its proof, we will first need the concept of a *right-truncation* of a coalition and a *left shift* of a coalition.

Definition 4. Let $S \subseteq N$ and $S' \subseteq N$ be two coalitions on agents $N = \{1, \ldots, n\}$. S' is a *left shift* of S when S' can be obtained from S by a sequence of replacements of higher-numbered agents by lower-numbered agents. Note that in a canonical linear game, any left shift of a winning coalition is winning.

Let p_i be the ith highest-numbered agent among the agents in S. The *ith right-truncation* of S, $\mathsf{tr}(S, i)$, is then defined as

$$\mathsf{tr}(S, i) = \begin{cases} S \backslash \{p_i, \ldots n\} \text{ if } 0 < i \leq |S|, \\ S \text{ if } i = 0, \\ \text{undefined otherwise.} \end{cases}$$

THEOREM 9. *For any n, let $(G, G') \in \mathcal{G}_{\mathsf{cwvg}}(n)^2$ be a pair of WVGs such that G covers G' in $(\mathcal{G}_{\mathsf{cwvg}}(n), \supseteq_{\mathsf{MWC}})$. Let $W_{\min,G}$ and $W_{\min,G'}$ be the sets of MWCs of G and G' respectively, and let $L_{\max,G}$ and $L_{\max,G'}$ be the sets of maximal losing coalitions of G and G' respectively. There is a $C \in L_{\max,G}$ and an $i \in \mathbb{N}$ with $0 \leq i \leq n$ such that $W_{\min,G'} = W_{\min,G} \cup \mathsf{tr}(C, i)$.*

PROOF. Because G covers G', by definition there is a coalition $C' \notin W_{\min,G}$ such that $W_{\min,G'} = W_{\min,G} \cup C'$. Clearly C' can not be a superset of any coalition in $W_{\min,G}$, so it must be a subset of a coalition in $L_{\max,G}$. Suppose for contradiction that C' is not a right-truncation of a maximal losing coalition $C \in L_{\max,G}$. Then there is left shift C'' of C' such that C'' is a subset of a coalition in $L_{\max,G}$, which means that C'' is not a superset of any coalition in $W_{\min,G}$, hence C'' is also not a superset of any coalition in $W_{\min,G'}$. So C'' is a losing coalition in G'. But G' is a canonical WVG, hence G' is also a canonical linear game. By the fact that canonical linear games have the total desirability relation $1 \succeq_D \cdots \succeq_D n$, C'' is a winning coalition in G' because it is a left shift of the winning coalition C'. This is a contradiction. □

From Theorem 9 we see how we can use $(\mathcal{G}_{\mathsf{cwvg}}(n), \supseteq_{\mathsf{MWC}})$ for enumerating the class of n-agent canonical WVGs: We start by outputting the n-agent WVG with zero MWCs. After that, we repeat the following process: generate the $\mathcal{L}_{W,\min}$-representation of all canonical WVGs with i MWCs, using the set of canonical weighted voting games games with $i - 1$ MWCs (also represented in $\mathcal{L}_{W,\min}$). Once generated, we have the choice to output the games in their $\mathcal{L}_{W,\min}$-representation or in their $\mathcal{L}_{\mathsf{weights}}$-representation, by using the transformation algorithm from Theorem 2.

Generating the set of games of i MWCs works as follows: For each game of $i-1$ MWCs, we obtain the set of maximal losing coalitions by using the algorithm from Theorem 2. Next, we check for each maximal losing coalition C whether there is a right-truncation of C that we can add to the set of MWCs, such that the resulting set is a weighted voting game. Again, testing whether a game is a WVG is done by using the algorithm from Theorem 2. If a game turns out to be weighted, we can save it and output it.

There is one remaining problem with this approach: It outputs duplicate games. If $(\mathcal{G}_{\mathsf{cwvg}}(n), \supseteq_{\mathsf{MWC}})$ were a tree, then this would not be the case, but by Theorem 8 it is not.

Therefore, we have to do a *duplicates-check* for each WVG that we find. We have to check whether we did not already generate it. We use the following algorithm for this check: Suppose that we have found an n-agent canonical WVG G of i MWCs by adding a coalition C to some MWC listing of a canonical WVG. We first sort G's list of MWCs. After that, we check each coalition C' that occurs before C in this sorted list. For each C', we check whether C''s removal from the list results in a list of MWCs of a canonical WVG. If this is the case for at least one such C', then we *do not* output the game G. By using this method, it is certain that each canonical WVG will be generated only once.

Algorithm 2 gives the pseudocode for the enumeration procedure.

Algorithm 2 An enumeration algorithm for the class of n-agent canonical WVGs.

1: {games[i] will contain the list of canonical WVGs that have i MWCs. The games are represented in language $\mathcal{L}_{W,\min}$. games[0] is our starting point. First we output the n-agent canonical WVG with zero MWCs.}
2: **Output** $((0,\ldots,0),1)$.
3: games[0] := $\{\varnothing\}$
4: **for** $i := 1$ to $\binom{n}{\lfloor \frac{n}{2} \rfloor}$ **do**
5: **for all** $W_{\min} \in$ games[$i-1$] **do**
6: {Obtain the maximal losing coalitions (see Th 2)}
7: $L_{\max} :=$ computeMLCs($W_{\min}$)
8: **for all** $C \in L_{\max}$ **do**
9: **for** $j := 1$ to n **do**
10: **if** isweighted($W_{\min} \cup \mathsf{tr}(C,i)$) **then**
11: **if** $W_{\min} \cup \mathsf{tr}(C,i)$ passes the duplicates-check (see discussion above) **then**
12: **Output** the weighted representation of the voting game with MWCs $W_{\min} \cup \mathsf{tr}(C,i)$).
13: **Append** $W_{\min} \cup \mathsf{tr}(C,i))$ to games[i].

THEOREM 10. *Algorithm 2 runs in $O^*(2^{n^2+2n})$ time.*

PROOF. Lines 6–13 are executed at most once for every canonical weighted voting game. From Sperner's theorem we know that any list of MWCs has fewer than $\binom{n}{\lfloor \frac{n}{2} \rfloor}$ elements. So line 7 runs in time $O(n\binom{n}{\lfloor \frac{n}{2} \rfloor}^2 + n^3\binom{n}{\lfloor \frac{n}{2} \rfloor}) = O(n^2\sqrt{n}2^n)$. Within one iteration of the outer loop (line 4), lines 10 to 13 are executed at most $n\binom{n}{\lfloor \frac{n}{2} \rfloor} = O(\sqrt{n}2^n)$ times (because $L_{\max}$ is also an antichain, so Sperner's theorem also applies for maximal losing coalitions). The time-complexity of one execution of lines 10 to 13 is:

- At line 10 we must solve a linear program. Using using e.g. Karmarkar's interior point algorithm [7] this takes time $O(n^{4.5}\binom{n}{\lfloor \frac{n}{2} \rfloor}) = O(n^4 2^n)$.
- At line 11, we must execute the duplicates-check. This consists of checking for at most $\binom{n}{\lfloor \frac{n}{2} \rfloor}$ sets of MWCs whether it is weighted. This involves running the computeMLCs algorithm, followed by solving a linear program. So in total this takes $O(n^3\sqrt{n}2^{2n})$.
- Lines 12 and 13 clearly take linear time.

Bringing everything together, we see that a single iteration through lines 6–13 costs $O(n^4 2^{3n})$ time. As said, these lines are executed at most $|\mathcal{G}_{\mathsf{cwvg}}(n)|$ times. We know that $|\mathcal{G}_{\mathsf{wvg}}(n)| \in O(2^{n^2-n})$ (see Appendix A), and of course that $|\mathcal{G}_{\mathsf{cwvg}}(n)| < |\mathcal{G}_{\mathsf{wvg}}(n)|$, so lines 6 to 16 are executed at most $O(2^{n^2-n})$ times, and therefore the runtime of the algorithm is in $O(2^{n^2+2n}n^4) = O^*(2^{n^2+2n})$. □

Although the runtime analysis of this algorithm that we gave is not very precise, the main point of interest that we want to emphasize is that this method runs in exponential time, instead of doubly exponential time. We can also show that this algorithm runs within an amount of time that is only polynomially greater than the amount of data output:

THEOREM 11. *Algorithm 2 runs in output-polynomial time.*

PROOF. The loop of line 5 is executed at most $|\mathcal{G}_{\mathsf{cwvg}}(n)|$ times. From Appendix A, we have as a lower bound that $|\mathcal{G}_{\mathsf{cwvg}}(n)| \in \Omega(2^{n^2(1-\frac{10}{\log n})}/n!2^n)$. One execution of lines 6–13 costs $O(n^4 2^{3n})$ time, and

$$O(n^4 2^{3n}) \subseteq O(2^{n^2(1-\frac{10}{\log n})}/n!2^n) \subseteq O(|\mathcal{G}_{\mathsf{cwvg}}(n)|).$$

Hence, the algorithm runs in $O(|\mathcal{G}_{\mathsf{cwvg}}(n)|^2)$ time. □

We can not give a very sharp bound on the space complexity of the algorithm, because we do not know about the maximum cardinality of an antichain in $(\mathcal{G}_{\mathsf{cwvg}}(n), \supseteq_{\mathsf{MWC}})$. However, it can be seen that it is also possible to generate the games in this poset in a depth-first manner, instead of a breadth-first manner like we do now. In that case, the number of space that needs to be used is bounded by the maximum length of a chain in $(\mathcal{G}_{\mathsf{cwvg}}(n), \supseteq_{\mathsf{MWC}})$, times the maximum number of MWCs. This is a total amount of $O(\frac{2^{2n}}{n})$ space.

Now that we have this enumeration algorithm for WVGs, we can use the same approach as in algorithm 1 in order to solve the $(f, g, \mathcal{G}_{\mathsf{cwvg}}, \mathcal{L}_{\mathsf{weights}})$-PVGD problem: for each game that is output, we simply compute the power index of that game and check if it is closer to the optimum than the best game we have found up till that point. We denote this algorithm by ExactEnum.

COROLLARY 12. ExactEnum *is an exact, anytime algorithm that runs in exponential time for $(f, g, \mathcal{G}_{\mathsf{cwvg}}, \mathcal{L}_{\mathsf{weights}})$-PVGD, for any choice of power index function f that is computable within exponential time.*

6. CONCLUSIONS AND FUTURE WORK

We derived the first *exact* algorithm for solving weighted voting game design problems. Although the time complexity of this algorithm is high, it is an exponential improvement over the "naive" method that was explained in Section 4. Moreover deriving this algorithm turned out not to be straightforward. Also, the algorithm has the anytime property, giving it potential for being used in practice.

Note that in most real-life examples, the number of players in a weighted voting game is rather small: usually 10 to 50 agents are involved. For our future work, our goal is to get this algorithm to yield good results within a reasonable amount of time, when the number of players is somewhere in this range. There is still a lot of room for improving the

proposed algorithm. For example, it turns out that significant speedup can be attained by making use of some specific properties of linear games, allowing us to only keep track of a small subset of the MWCs of each game, instead of all MWCs.

It will also be interesting to study in more depth the partial order that we presented in this paper. With regard to the design of WVGs, we think that it is possible to prune a lot of "branches" in this partial order, i.e., it is safe to skip the enumeration of a lot of games in the partial order. Moreover, we are also curious to see how an algorithm performs that searches through the partial order in a greedy manner, or what will happen if we use some other heuristic method to search the partial order. Lastly, we can use this idea as a postprocessing step to extend the existing algorithms by Fatima et al. [6] and Aziz et al. [2], that we mentioned in Section 3.

7. REFERENCES

[1] N. Alon and P. H. Edelman. The inverse banzhaf problem. *Social Choice and Welfare*, to appear.

[2] H. Aziz, M. Paterson, and D. Leech. Efficient algorithm for designing weighted voting games. In *Proc. Int. Multitopic Conf.*, 2007.

[3] J. F. Banzhaf, III. Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Rev.*, 19, 1965.

[4] J. Bilbao, J. Fernández, A. Losada, and J. López. Generating functions for computing power indices efficiently. *TOP*, 8, 2000.

[5] X. Deng and C. Papadimitriou. On the complexity of cooperative solution concepts. *Math. Oper. Res.*, 1994.

[6] S. Fatima, M. Wooldridge, and N. Jennings. An anytime approximation method for the inverse Shapley value problem. In *Proc. AAMAS*, 2008.

[7] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proc. STOC*, 1984.

[8] D. Kleitman and M. Markowski. On Dedekind's problem: The number of isotone boolean functions II. In *Trans. Am. Math. Soc.*, 1975.

[9] A. Laurelle and M. Widgren. Is the allocation of voting power among EU states fair? *Publ. Choice*, 94, 1998.

[10] D. Leech. Power indices as an aid to institutional design: the generalised apportionment problem. In *Yearbook on New Polit. Econ.* Mohr Siebeck, 2003.

[11] S. Muroga. *Threshold logic and its applications.* Wiley-Interscience, 1971.

[12] U. Peled and B. Simeone. Polynomial-time algorithms for regular set-covering and threshold synthesis. *Discr. Appl. Math.*, 12, 1985.

[13] K. Prasad and J. Kelly. NP-completeness of some problems concerning voting games. *Int. J. of Game Theory*, 19, 1990.

[14] L. S. Shapley. A value for N-person games. *Annals of Mathematics Study*, 28:307–317, 1953.

[15] E. Sperner. Ein Satz über Untermengen einer endlichen Menge. *Math. Zeitschr.*, 27(1), 1928.

[16] M. Sutter. Fair allocation and re-weighting of votes and voting power in the EU before and after the next enlargement. *J. of Theor. Politics*, 12, 2000.

[17] Y. A. Zuev. Asymptotics of the logarithm of the number of threshold functions of the algebra of logic. *Soviet Math. Dokl.*, 39:512–513, 1989.

[18] J. Žunić. On encoding and enumerating threshold functions. *IEEE Trans. Neur. Networks*, 15, 2004.

APPENDIX

A. LOWER AND UPPER BOUNDS ON THE NUMBER OF WVGS

To our knowledge, in game theory literature there has not been any research in the amount of WVGs on n agents. Fortunately there is a closely related field of research, called *threshold logic* (see e.g. [11]), that has some relevant results.

Definition 5. Let f be a boolean function on n boolean variables. f is a *(boolean) threshold function* when there exists a weight vector of real numbers $r = (r_0, r_1, \dots r_n) \in \mathbb{R}^{n+1}$ such that $r_1x_1 + \cdots + r_nx_n \geq r_0$ iff $f(x_1, \dots, x_n) = 1$. We say that r *realizes* f. We denote the set of threshold functions of n variables $\{x_1, \dots, x_n\}$ by $\mathbf{LT}(n)$.[5]

Threshold functions resemble WVGs, except that we talk about *boolean variables* instead of *agents*. Also, an important difference is that $r_0, r_1, \dots, r_n$ are allowed to be negative for threshold functions, whereas $q, w_1, \dots, w_n$, must be non-negative in WVGs. Žunić presents in [18] an upper bound on the number of threshold functions of n variables $|\mathbf{LT}(n)|$: $|\mathbf{LT}(n)| \leq 2^{n^2-n+1}$. Also, the following asymptotic lower bound is known, as shown in [17]: For large enough n, we have $|\mathbf{LT}(n)| \geq 2^{n^2(1-\frac{10}{\log n})}$.

From these bounds, we can deduce some easy upper and lower bounds for $|\mathcal{G}_{\mathsf{wvg}}|$. Let $\mathbf{LT}^+(n)$ be the set of *non-negative threshold functions* of variables $\{x_1, \dots, x_n\}$: threshold functions $f \in \mathbf{LT}(n)$ for which there exists a *non-negative weight vector* r that realizes f, i.e. r realizes f and only has non-negative entries. There is a clear one-to-one correspondence between non-negative threshold functions and WVGs, so we can conclude that $|\mathcal{G}_{\mathsf{wvg}}(n)| = |\mathbf{LT}^+(n)|$. So now we can upper bound the number of WVGs.

COROLLARY 13. $|\mathcal{G}_{\mathsf{wvg}}(n)| \leq 2^{n^2-n+1}$.

PROOF. This follows directly from $|\mathcal{G}_{\mathsf{wvg}}(n)| = |\mathbf{LT}^+(n)|$ and $|\mathbf{LT}^+(n)| \subseteq |\mathbf{LT}(n)|$, and the upper bound on the number of threshold functions. □

We will proceed by obtaining a lower bound on the number of WVGs. For any n, for every threshold function in $\mathbf{LT}(n) \backslash \mathbf{LT}^+(n)$ and each $r \in \mathbb{R}^{n+1}$ that realizes f, there exists a $r' \in (\mathbb{R}^+)^{n+1}$ such that r is obtained by negating some of the entries in r', and r' is a realization of some threshold function in $\mathbf{LT}^+(n)$. This implies that $|\mathcal{G}_{\mathsf{wvg}}(n)| \geq |\frac{\mathbf{LT}(n)}{2^{n+1}}|$. So the following lower bound follows:

COROLLARY 14. *For large enough n, it holds that* $|\mathcal{G}_{\mathsf{wvg}}(n)| \leq 2^{n^2(1-\frac{10}{\log n})-n-1}$.

[5] "LT" stands for "Linear Threshold function".