Journal of Artificial Intelligence Research 50 (2014) 105-140 Submitted 07/13; published 05/14

Finding Optimal Solutions for Voting Game Design Problems

Bart de Keijzer DEKEIJZER@DIS.UNIROMA1.IT
Web Algorithmics and Data Mining,
Sapienza — Università di Roma,
Rome, Italy

Tomas B. Klos T.B.KLOS@TUDELFT.NL
Algorithmics; Delft University of Technology,
Delft, The Netherlands

Yingqian Zhang YQZHANG@ESE.EUR.NL
Department of Econometrics; Erasmus University Rotterdam,
Rotterdam, The Netherlands

Abstract

In many circumstances where multiple agents need to make a joint decision, voting is used to aggregate the agents' preferences. Each agent's vote carries a *weight*, and if the sum of the weights of the agents in favor of some outcome is larger than or equal to a given *quota*, then this outcome is decided upon. The distribution of weights leads to a certain distribution of power. Several 'power indices' have been proposed to measure such power. In the so-called *inverse problem*, we are given a target distribution of power, and are asked to come up with a game—in the form of a quota, plus an assignment of weights to the players—whose power distribution is as close as possible to the target distribution (according to some specified distance measure).

Here we study solution approaches for the larger class of *voting game design (VGD) problems*, one of which is the inverse problem. In the general VGD problem, the goal is to find a voting game (with a given number of players) that optimizes some function over these games. In the inverse problem, for example, we look for a weighted voting game that minimizes the distance between the distribution of power among the players and a given target distribution of power (according to a given distance measure).

Our goal is to find algorithms that solve voting game design problems exactly, and we approach this goal by *enumerating* all games in the class of games of interest. We first present a doubly exponential algorithm for enumerating the set of *simple games*. We then improve on this algorithm for the class of *weighted voting games* and obtain a quadratic exponential (i.e., $2^{O(n^2)}$) algorithm for enumerating them. We show that this improved algorithm runs in *output-polynomial time*, making it the fastest possible enumeration algorithm up to a polynomial factor. Finally, we propose an exact *anytime*-algorithm that runs in exponential time for the power index weighted voting game design problem (the 'inverse problem').

We implement this algorithm to find a weighted voting game with a normalized Banzhaf power distribution closest to a target power index, and perform experiments to obtain some insights about the set of weighted voting games. We remark that our algorithm is applicable to optimizing any exponential-time computable function, the distance of the normalized Banzhaf index to a target power index is merely taken as an example.

1. Introduction

In many real-world settings involving multiple players who have to come up with a joint decision, for instance elections, there is a need for fair decision-making protocols in which different players have different amounts of influence on the outcome of a decision. A *weighted voting game* (WVG) is often used as such a decision-making protocol. In a weighted voting game, a quota is given, and each player (or agent) in the game has a certain weight. If the total weight of a coalition of agents is not smaller than the quota, then that coalition is said to be *winning*, and otherwise, *losing*.

Weighted voting games arise in various practical settings, such as in political decision making (decision making among larger and smaller political parties), stockholder companies (where number of shares determines amount of influence), and elections (e.g., in the US presidential election, where each state can be regarded as a player who has a weight equal to its number of electors).

The weight that a player has is not equal to his actual influence on the outcome of the decisions that are made using the weighted voting game. Consider for example a weighted voting game in which the quota is equal to the sum of the weights of all players. In such a game, a player's influence is equal to the influence of any other player regardless of the weight he has. Various *power indices* have been proposed in the literature, as ways to measure a player's (a priori) power in influencing the outcome of a voting game (see Banzhaf III, 1965; Dubey & Shapley, 1979; Shapley & Shubik, 1954). However, computing a power index has turned out to be a challenge in many cases: see the work of Algaba, Bilbao, García, and López (2003), Deng and Papadimitriou (1994), Matsui and Matsui (2001), Prasad and Kelly (1990), and De Keijzer (2009b) for a recent survey.

In this paper, instead of analyzing the power of each agent in a voting game, we investigate the problem referred to as the "inverse problem" (see Alon & Edelman, 2010) or the "generalized apportionment problem" (see Leech, 2003). We will call this problem the *weighted voting game design problem.* In the power index voting game design problem, we are given a target power index for each of the agents, and we study how to design a weighted voting game for which the power of each agent is as close as possible to the given target power index.

The motivation behind our work is a practical one: It is desirable to have an algorithm that can quickly compute a fair voting protocol, given that we want each agent to have some specified amount of influence in the outcome. When new decision making bodies must be formed, or when changes occur in the formation of these bodies, such an algorithm may be used to design a voting method that is as fair as possible.

The most intuitive approach for solving the weighted voting game design problem would be to simply enumerate *all possible* weighted voting games of n players. However, enumerating all weighted voting games efficiently is not quite as straightforward as it seems. Because even for a single weighted voting game there is an infinite number of weighted representations (the representation of a game as a listing of a quota and a weight for each player), enumerating games in this weighted representation is not an option. To address this problem, we prove the existence of, and exploit, a new partial order on the class of weighted voting games that allows us to efficiently enumerate all weighted voting games. This enumeration method leads to an *exact* algorithm that can be used to solve any weighted voting game design problem. Although this algorithm runs in exponential time, asymptotically it is a big improvement over the naive approach that runs in doubly exponential time, as we will show. Therefore, besides the fact that we broaden our understanding of the problem by this improvement to exponential time, the algorithm allows us to solve the problem to optimality for small numbers of players, and to find approximate solutions for larger numbers of players, thanks to its *anytime* property.[1] We implement our algorithm for the power index voting game design problem where our power index of choice is the *(normalized) Banzhaf index*. We emphasize that this choice of power index is quite arbitrary: we could pick any other power index as well, and the normalized Banzhaf index is merely taken as an example. We use our implementation to illustrate the applicability of our approach and to obtain some relevant statistics on the set of weighted voting games.

In the next section we define relevant concepts and set up some notation. Section 3 then formally defines the problem we address, and gives an overview of related work. In Section 4, we analyze the problem and present our solution. First we address the design of (monotonic) simple games in Section 4.1. Then, in the most important section (4.2), we focus on weighted voting game design. Section 4.3 concludes this part of the paper with some improvements that we wish to discuss separately from the main ideas. In Section 5, we report on some experiments we performed with an implementation of our main algorithm, and Section 6 concludes the paper.

2. Preliminaries

In this section, we will discuss some required preliminary definitions and results related to the theory of cooperative simple games.[2]

A *cooperative game* is a pair (N, v), where N is a finite set of *players*; subsets of N are called *coalitions*, and $v : 2^N \to \mathbb{R}_{\geq 0}$ is the *characteristic function* or *gain function*, mapping coalitions to non-negative real numbers. Intuitively, v describes how much collective payoff a coalition of players can gain when they cooperate. The set N is also called the *grand coalition*. A *simple game* is a cooperative game (N, v) where the codomain of v is restricted to $\{0, 1\}$.[3] In this context, a coalition S is called a *winning coalition* if $v(S) = 1$, and a *losing coalition* otherwise. A cooperative game (N, v) is *monotonic* if and only if $v(S) \leq v(T)$ for all pairs of coalitions $(S, T) \in 2^N \times 2^N$ that satisfy $S \subseteq T$. In other words: in a monotonic game, the value of a coalition can not decrease when players are added to the coalition.

In this paper, we are concerned with the class of monotonic simple games, which we denote as $\mathcal{G}_{\mathsf{mon}}$. In general, if $\mathcal{G}$ is a class of games, we use $\mathcal{G}(n)$ to denote that class of

1. An algorithm is said to have the *anytime* property if: (i) it is able to provide a solution anytime during execution, (ii) the solution quality improves the longer the algorithm runs, and (iii) the algorithm is guaranteed to output the optimal solution eventually.
2. Much of the information in this section can be found in an introductory text on cooperative game theory (e.g., Peleg & Sudhölter, 2003) or on simple games (e.g., Taylor & Zwicker, 1999). Throughout this paper, we assume familiarity with big-O notation and analysis of algorithms, as well as knowledge of some basic order-theoretic notions related to graded partial orders. In some parts of this paper, some basic knowledge of computational complexity theory is assumed as well, although these parts are not crucial for understanding the main results presented. We will not cover these topics in this section.
3. We purposefully do not exclude the games with only losing and only winning coalitions, as is customary. The reason is that including these games will make it more convenient later to show that a particular structure exists in a subclass of the simple games.

games, restricted to the set of players $\{1, \ldots, n\}$. So $\mathcal{G}_{\mathsf{mon}}(3)$ is the class of monotonic simple games with 3 players.

There are various important ways to represent (classes of) simple games. If $G = (N, v)$ is a simple game, we let W_G and L_G be G's sets of *winning* and *losing* coalitions, respectively, so that $W_G \cup L_G = 2^N$ and $W_G \cap L_G = \varnothing$. The set $W_{\min,G} \subseteq W_G$ of G's *minimal winning coalitions* (MWC) contains every winning coalition from which we cannot remove a player without making it a losing coalition. The set $L_{\max,G} \subseteq L_G$ of G's *maximal losing coalitions* (MLC) is defined analogously: A coalition is *maximal losing* if it's losing and no player can be added to it without making it winning. We can now describe a simple game in the following forms:

Winning coalition form: (N, W_G) is called the *winning coalition form* of G.

Losing coalition form: (N, L_G) is called the *losing coalition form* of G.

Minimal winning coalition form: $(N, W_{\min,G})$ is the *minimal winning coalition form* of G. Observe that $W_{\min,G}$ fully describes v if and only if G is monotonic.

Maximal losing coalition form: $(N, L_{\max,G})$ is the *maximal losing coalition form* of G. Again, $L_{\max,G}$ fully describes v if and only if G is monotonic.

For some, but not all simple games, the exists another representation.

Weighted form: For a simple game $G = (N, v)$, if there exists a quota $q \in \mathbb{R}_{\geq 0}$ and a weight $w_i \in \mathbb{R}_{\geq 0}$ for each player $i \in N$, such that for each coalition $S \in 2^N$ it holds that $v(S) = 1 \Leftrightarrow \sum_{i \in S} w_i \geq q$, then we say that G is *weighted* or *has a weighted form*, and the vector $w = (q, w_1, \ldots, w_n)$, also written as $[q; w_1, \ldots, w_n]$, is called a *weighted representation* of G. (We will write $w(S)$ as a shorthand for $\sum_{i \in S} w_i$.) Observe that every game that has a weighted form is also monotonic. The converse is not true in general,[4] but we are interested only in those monotonic games that *do* have a weighted form.

Games that have a weighted form, or *weighted voting games*, are our main interest. We denote the class of all weighted voting games by $\mathcal{G}_{\mathsf{wvg}}$. A weighted voting game is an important type of simple game because it has a compact representation, and because it is used in many practical situations, such as elections, (European Union) political decision making, and stockholder meetings. An important property of weighted voting games that we will use is that a weighted representation of such a game is *invariant to scaling*: if we multiply the quota and each of the weights in the weighted form of a WVG G with a constant $c \in \mathbb{R}^+$, then the resulting weighted form $[cq; cw_1, \ldots, cw_n]$ represents the same game G, that is, it has the same winning and losing coalitions.

We next turn our attention to the topic of influence and power in monotonic simple games. For a monotonic simple game, it is possible to define a relation called the *desirability relation* among the players (see Isbell, 1958):

4. To see this, consider the four-player monotonic game with MWCs $\{1, 2\}$ and $\{3, 4\}$. Suppose, for the sake of contradiction, that there exist weights $w_1, \ldots, w_4$ and a quota q that form a weighted representation of this game. Then $w_1 + w_2 \geq q$ and $w_3 + w_4 \geq q$, so $w_1 + w_2 + w_3 + w_4 \geq 2q$. But coalitions $\{1, 3\}$ and $\{2, 4\}$ are losing, since they are not (supersets of) MWCs, so $w_1 + w_3 < q$ and $w_2 + w_4 < q$, which means that $w_1 + w_2 + w_3 + w_4 < 2q$, yielding a contradiction.

Definition 1 (Desirability relation)**.** *For a monotonic simple game (N, v), the* desirability relation $\succeq_v$ *is defined as follows: For any $(i, j) \in N^2$:*

- *if $\forall S \subseteq N \setminus \{i, j\} : v(S \cup \{i\}) \geq v(S \cup \{j\})$, then $i \succeq_v j$. We say that i is* more desirable *than j.*
- *if $\forall S \subseteq N \setminus \{i, j\} : v(S \cup \{i\}) = v(S \cup \{j\})$, then $i \sim_v j$. We say that i and j are* equally desirable*.*
- *if $\forall S \subseteq N \setminus \{i, j\} : v(S \cup \{i\}) \leq v(S \cup \{j\})$, then $j \succeq_v i$ (also written as $i \preceq_v j$). We say that i is* less desirable *than j.*
- *if $i \succeq_v j$ and not $i \sim_v j$, then $i \succ_v j$. We say that i is* strictly more desirable *than j.*
- *if $i \preceq_v j$ and not $i \sim_v j$, then $i \prec_v j$. We say that i is* strictly less desirable *than j.*

Moreover, if neither $i \succeq_v j$ nor $j \succeq_v i$ holds for some $i, j \in N$, then we call i and j incomparable*.*

Using this notion of desirability, we can now define the class of *linear games.*

Definition 2 (Linear game)**.** *A simple game (N, v) is a* linear game *if and only if it is monotonic, and no pair of players in N is incomparable with respect to $\preceq_v$. Thus, for a linear game (N, v), $\preceq_v$ is a total preorder on N. We denote the class of linear games by $\mathcal{G}_{\mathsf{lin}}$.*

All weighted voting games are linear. If (N, v) is a weighted voting game with weighted form $[q; w_1, \ldots, w_n]$, then $i \preceq_v j$ when $w_i \leq w_j$. So, every pair of players is comparable with respect to $\preceq_v$. In fact, the following sequence of strict containments holds: $\mathcal{G}_{\mathsf{wvg}} \subset \mathcal{G}_{\mathsf{lin}} \subset \mathcal{G}_{\mathsf{mon}}$. The following are definitions of two special classes of games used in subsequent sections.

Definition 3 (Canonical weighted voting games and canonical linear games)**.** *A linear game (N, v) is a* canonical linear game *whenever $N = \{1, \ldots, n\}$ for some $n \in \mathbb{N}_{>0}$, and the desirability relation $\succeq$ satisfies $1 \succeq 2 \succeq \cdots \succeq n$. When G is also weighted, then G is a* canonical weighted voting game *(CWVG). The class of canonical linear games is denoted by $\mathcal{G}_{\mathsf{clin}}$, and the class of CWVGs is denoted by $\mathcal{G}_{\mathsf{cwvg}}$. Note that a CWVG always has a weighted representation that is nonincreasing.*

There are two special ways of representing canonical linear games. To introduce these, we need the notions of *left-shift* and *right-shift.*

Definition 4 (Left-shift and right-shift)**.** *Let N be the set of players $\{1, \ldots, n\}$ and let S be a nonempty subset of N. A coalition $S' \subseteq N$ is a* direct left-shift *of S whenever there exists an $i \in S$ and an $i - 1 \notin S$ with $2 \leq i \leq n$ such that $S' = (S \setminus \{i\}) \cup \{i - 1\}$. A coalition $S' \subseteq N$ is a* left-shift *of S whenever for some $k \geq 1$ there exists a sequence $(S_1, \ldots, S_k) \in (2^N)^k$, such that:*

- *$S_1 = S$,*
- *$S_k = S'$,*

- *for all i with $1 \leq i < k$, we have that S_{i+1} is a direct left-shift of S_i.*

We say that a coalition S' is a strict *left-shift of S when S' is a left-shift of S and $S' \neq S$. The definitions of* direct right-shift *and* (strict) right-shift *are obtained when we replace in the above definition $i-1$ with $i+1$ and $i+1$ with $i-1$.*

For example, coalition $\{1,3,5\}$ is a *direct* left-shift of coalition $\{1,4,5\}$ because in the former coalition, player 4 is replaced by player $4-1=3$, and coalition $\{1,2,4\}$ is a left-shift of $\{1,4,5\}$, because the former can be obtained from the latter by a sequence of (three) such direct left shifts.

The notions of left-shift and right-shift make sense for canonical linear games and canonical weighted voting games. Due to the specific desirability order that holds in canonical linear games, a left-shift of a winning coalition is always winning in such a game, because in the left-shift, some player from the winning coalition is replaced by a lower-numbered, and thus more desirable player. Similarly, a right-shift of a losing coalition is always losing in such a game. This allows us to represent a canonical linear game in one of the following two forms.

Definition 5 (Roof/ceiling coalition/form). *Let $G = (N, v)$ be a canonical linear game. Also, let $W_{\min,G}$ be G's set of minimal winning coalitions and let $L_{\max,G}$ be G's set of maximal losing coalitions. A minimal winning coalition $S \in W_{\min,G}$ is a* roof coalition *whenever every right-shift of S is losing. Let $W_{\mathsf{roof},G}$ denote the set of G's roof coalitions. The pair $(N, W_{\mathsf{roof},G})$ is called the* roof form *of G. A maximal losing coalition $S \in L_{\max,G}$ is a* ceiling coalition *whenever every left-shift of S is winning. Let $W_{\mathsf{ceil},G}$ denote the set of G's ceiling coalitions. The pair $(N, W_{\mathsf{ceil},G})$ is called the* ceiling form *of G.*[5]

2.1 Power Indices

If we consider the weighted voting game with weighted form $[100; 98, 1, 1]$, we see that the weight of a player is not necessarily directly proportional to the influence he has in the game. In this game, only the grand coalition is winning. Since all players need to be present in this coalition for it to be winning, they can all be said to have the same influence, despite the fact that there is a huge difference between the weights of the first versus the other players.

A variety of power indices have been proposed to measure players' influence (instead of weight) in a monotonic simple game. Power indices measure players' *a priori* power in a voting game. For this reason, power indices do not rely on (statistical) information about which coalitions are likely to *actually* form due to the preferences of the players. In this paper, we focus on the *normalized Banzhaf index* (also called simply the Banzhaf index, see Banzhaf III, 1965), inasmuch as it is used in the experiments in Section 5. However, for the theoretical part of our work, the particular choice of power index is irrelevant.

Definition 6 (normalized Banzhaf index & raw Banzhaf index). *The* (normalized) Banzhaf index *of a monotonic simple game $(N = \{1, \ldots, n\}, v)$ is defined as $\beta = (\beta_1, \ldots, \beta_n)$, where*

5. The terminology "roof" and "ceiling" is taken from Peled and Simeone (1985), while Taylor and Zwicker (1999) call these coalitions *shift-minimal winning coalitions* and *shift-maximal losing coalitions.*

for $1 \leq i \leq n$,

$$\beta_i = \frac{\beta'_i}{\sum_{j=1}^{n} \beta'_j}, \quad \textit{and} \quad \beta'_i = |\{S \subseteq N \setminus \{i\} : v(S) = 0 \wedge v(S \cup \{i\}) = 1\}|. \tag{1}$$

Here, β'_i *is called the* raw Banzhaf index of player i*, and it counts the number of losing coalitions of other agents that player* i *can turn into winning by joining them.*

The problem of computing power indices and its associated computational complexity has been widely studied. For a survey of complexity results, and exact and approximation algorithms for computing power indices, see the work of De Keijzer (2009b). Prasad and Kelly (1990) prove computation of the raw Banzhaf index to be #P-complete,[6] and the fastest known exponential time algorithm for computing the Banzhaf index is due to Klinz and Woeginger (2005), achieving a runtime in $O((\sqrt{2})^n \cdot n^2)$.

3. Problem Statement and Related Work

In its most general statement, the *voting game design* (VGD) problem is the problem of finding a simple game that optimizes a given requirement. One obtains different variants of the problem by specifying (i) the type of simple game one is interested in, and (ii) the requirement to be optimized. In this paper, we focus on finding a *weighted voting game* (in weighted form), and on the requirement that the normalized Banzhaf power index of the game is as close as possible to a given target power index. We call this variant the *power index weighted voting game design* (PIWVGD) problem. Please note that the approach we propose is not specific to minimizing the distance with a target power index; other optimization criteria can be addressed as well. In fact, most of what follows pertains just to enumerating the games in given classes, and not to the optimization part of the problem, let alone the focus on the (normalized Banzhaf) power index. Only in (some of) our experiments do we focus on the normalized Banzhaf index. In those experiments, we choose to measure closeness in terms of the Euclidean distance in $\mathbb{R}^n$ between the normalized Banzhaf power index of the game and the target power index.

To illustrate the problem, we visualize the 3 player-instance of the problem, because this can be done nicely in two dimensions. The normalized Banzhaf index of each weighted voting game is a vector in the 2-dimensional unit simplex.[7] In our analysis in Section 4, it will turn out to be convenient to restrict ourselves to *canonical* WVGs (see Definition 3). We can do this without loss of generality, because for every WVG there exists an canonical WVG that can be obtained by ordering the players. For three players, these games have normalized Banzhaf power indices only in the shaded area in Figure 1, where the vertices of the simplex are labeled with the players' numbers (1, 2 and 3). corresponding power indices are listed on the left.

The four dark dots in Figure 1 represent the normalized Banzhaf power indices corresponding to all ten existing three-player canonical WVGs. Two of these ten games are

6. #P is the complexity class that contains the counting versions of problems in NP. Problems complete for this class are believed to be hard to solve, and a polynomial-time algorithm for one such problem implies P = NP.

7. Recall that the n-dimensional unit simplex is defined as $\{x \in (\mathbb{R}_{\geq 0})^{n+1} : \sum_{i=1}^{n+1} x_i = 1\}$, so it contains the $(n+1)$-dimensional vectors of non-negative real numbers for which the elements sum to 1.

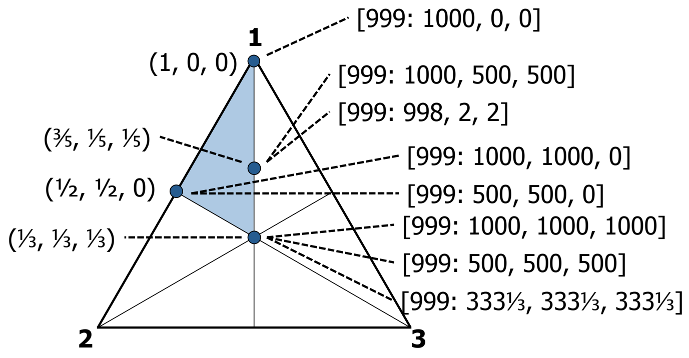

Figure 1: The games with three players and their power indices.

'degenerate,' namely, the two games with no winning and no losing coalitions, respectively. Weighted form representations for the other eight games are given on the right of the figure, which also shows that different games may have the same distribution of power. The PIWVGD problem (for $n = 3$) is now: given a target power index (a point) somewhere in the shaded area of the figure (and in the corresponding part of the $(n-1)$-dimensional unit simplex for general n), return a weighted representation of a game that is closest to this target (in terms of Euclidean distance, for example).

We know of only a couple of studies that propose algorithms for the inverse problem. Fatima, Wooldridge, and Jennings (2008) and Aziz, Paterson, and Leech (2007) present similar algorithms for the inverse problem for a target Shapley-Shubik index (Shapley & Shubik, 1954) and Banzhaf index (Banzhaf III, 1965), respectively. Both algorithms iteratively update a weight vector by using update rules based on the distance from the current weight vector's power index and the target power index. Fatimal et al. use two update rules for which they prove that by applying them, the Shapley-Shubik index of each player cannot get further away from the target. Hence, the proposed algorithm is an anytime algorithm. Aziz et al. do not give such an analysis. Leech (2002a, 2003) proposes an approach that largely resembles the method of Aziz et al., with the exception that a different updating rule is used. Neither algorithm comes with an approximation guarantee.

There are two recent interesting works on the voting game design problem. One is by Kurz (2012b). Kurz proposes an exact method using integer linear programming, for solving the weighted voting game design problem for both the Shapley-Shubik index and the Banzhaf index. The set of linear games is taken as the search space, and branch-and-bound techniques (along with various insights about the set of weighted voting games) are used in order to find in this set a weighted voting game with a power index closest to the target. Kurz does not provide a runtime analysis. The experiments performed show that the algorithm works well for small numbers of players. Our work is independent of the work of Kurz and differs from it in that we are interested in devising an algorithm with provable as-good-as-possible runtime guarantees. Moreover, the approach we take is different from that of Kurz, and the theory necessary to develop our algorithm can be considered interesting in itself.

The other recent work is by De, Diakonikolas, and Servedio (2012b). This paper provides as its main result an algorithm for the inverse power index problem for the case of the Shapley-Shubik index, and has a certain approximation guarantee: in addition to a target power index, the algorithm takes a precision parameter ϵ and guarantees to output a weighted voting game of which the power index is ϵ-close to it, on the precondition that there *exists* an ϵ-close weighted voting game with the property that the quota is not too skewed, in a particular sense. This is, to our knowledge, the only polynomial time algorithm for a power index voting game design problem that provides an approximation guarantee in any sense.

Closely related to our work are two papers that deal with the Chow parameters problem (O'Donnell & Servedio, 2011; De, Diakonikolas, Feldman, & Servedio, 2012a). The results in their paper are stated in terms of boolean function theory and learning theory, but when translated to our setting, these papers can be seen to deal with approximation algorithms for a type of value that can be considered a power index: The Chow parameters of a given player in a given game is defined to be the total number of winning coalitions that the player is in. The authors present in these papers, as a main result, a polynomial time approximation scheme for computing the Chow parameters of a weighted voting game.

The problem of *enumerating* the set of weighted voting games with a fixed number of players is, as we will see, closely related to the approach we take for solving the weighted voting game design problem. This enumeration problem was studied by Kurz (2012a), who uses integer programming techniques to enumerate all canonical weighted voting games up to nine players. Kurz generates integer weighted representations for all of these games and classifies the games that do not have a unique minimum-sum integer weighted representation.

Threshold functions (Hu, 1965; Muroga, 1971) are of fundamental research interest in voting games, circuit complexity and neural networks. The problem of realizing Boolean threshold functions by neural networks has been extensively studied (Parberry, 1994; Siu, Roychowdhury, & Kailath, 1995; Freixas & Molinero, 2008), where upper and lower bounds are derived on the synaptic weights for such realizations. The enumeration of threshold functions is closely related to the enumeration of weighted voting games (see Appendix A): Threshold functions are essentially weighted voting games where negative weights are allowed. The enumeration of threshold functions up to six variables was done by Muroga, Toda, and Kondo (1962). Subsequently, in the work of Winder (1965), and Muroga, Tsuboi, and Baugh (1970), all threshold functions of respectively seven and eight variables were enumerated. Krohn and Sudhölter (1995) enumerated the canonical weighted voting games up to eight players, as well as the class of canonical linear games. Kurz (2012a) was the first to enumerate all nine player canonical weighted voting games, and Freixas and Molinero (2010) were the first to enumerate all nine player canonical linear games. To the best of our knowledge, enumerations for 10 players have not been carried out.

There exists some literature on enumeration of special subclasses of voting games as well: see the work of Freixas, Molinero, and Roura (2012) for linear games with two desirability classes; the work of Freixas and Kurz (2013a) for weighted voting games with one roof; and the work of Freixas and Kurz (2013b) for linear games with certain special types of voters and few desirability classes.

In the work of Krohn and Sudhölter's (1995), the enumeration of canonical linear games and a subclass thereof is studied using various order theoretic concepts. It does not directly address the problem of enumerating weighted voting games, although it does discuss a correspondence between the n-player *proper* weighted voting games and the $(n+1)$-player canonical *decisive* weighted voting games.[8] Because the class of canonical linear games is much bigger than the class of weighted voting games, their algorithms do not imply an efficient enumeration procedure for weighted voting games, as is one of our main contributions in the present work. However, there are some connections between our work and Krohn and Sudhölter: their enumeration procedures work by exploiting graded posets, just like ours; although their posets consist of subsets of winning coalitions together with the set inclusion relation (for the case of decisive canonical linear games, they use a variant of this poset), and not on the subsets of minimal winning coalitions as in our case. Although their idea of using a graded poset corresponds with ours, it seems to us that our results cannot be connected to theirs in any stronger sense. Moreover, the proofs of the properties that we establish for the partially ordered set we propose here, use vastly different ideas, and crucially exploit weightedness.

Alon and Edelman (2010) observe that we need to know *a priori* estimates of what power indices are achievable in simple games to analyze the accuracy of these kinds of iterative algorithms, i.e., there is a need for information about the distribution of power indices in $[0,1]^n$. As a first step in solving this problem, they prove a specific result for the case of the Banzhaf index for monotonic simple games.

In addition, some applied work has been done on the design of voting games. Laruelle and Widgrén (1998), and Sutter (2000) analyze and design the distribution of voting power in the European Union using iterative methods that resemble the algorithm of Aziz et al. (2007). Laruelle and Widgrén's algorithm was systematically analyzed and improved by De Nijs, Wilmer, and Klos (2012). Similar work was done by Leech for the EU (Leech, 2002b), and for the IMF (Leech, 2002c).

Finally, a research direction that is related to our problem is that of studying minimal integer representations for weighted voting games: Bounds on the maximum weight in such a representation provide us with a finite set of weighted representations to search through, as a means of solving our design problem. We explain this in greater detail in the next section. Some classical relevant bounds can be found in the work of Muroga (1971), Section 9.3. See the work of Freixas and Kurz (2011), Freixas and Molinero (2010) for some recent work in this direction.

4. Solving the Power Index Voting Game Design Problem

The most natural representation for a weighted voting game is the weighted representation. However, by the invariance to scaling of weighted representations, there exist an *infinite* number of weighted representations for each individual weighted voting game, even though for every n, the number of weighted voting games is finite: weighted voting games are simple games, and there are 2^{2^n} simple games for n players. This makes it hard to derive an exact algorithm for the weighted voting game design problem that is based on working with

8. A game is called proper if the complement of any winning coalition is losing. A game is called decisive if it is proper and the complement of any losing coalition is winning.

weighted representations alone, since there is not immediately a clear finite set of weight vectors that an algorithm can search through.[9] We resort to working with alternative representations.

We approach voting game design problems by devising an enumeration method that generates every voting game relatively efficiently. First, we discuss a "naive" method that enumerates all monotonic simple games on a given number of players in doubly exponential time (Section 4.1). Subsequently, in Section 4.2, for the case of *weighted voting games*, we improve on this runtime exponentially by showing how to enumerate all weighted voting games on a given number of players within exponential time. Although the runtime of this enumeration method is still exponential, we will see that the algorithm for the PIWVGD problem that results from this enumeration method (trivially) has the *anytime* property: Because we remember the best game found so far, the longer we run our algorithm, the better the result becomes. In addition, we are guaranteed that the algorithm eventually finds the *optimal* answer. The enumeration method exploits a specific (graded) partial order that we prove to exist for the class of weighted voting games.

Because we will be dealing with algorithms that run in exponential time, we make use of the O^*-notation: A function $f : \mathbb{R} \to \mathbb{R}$ is in $O^*(g)$ for some $g : \mathbb{R} \to \mathbb{R}$ if and only if there is a polynomial $p : \mathbb{R} \to \mathbb{R}$ such that $f \in O(g \cdot p)$. This essentially means that we make light of polynomial factors.

4.1 Monotonic Simple Game Design

In this section we consider briefly the power index voting game design problem for the class of monotonic simple games.

A monotonic simple game is represented by either a set of minimal winning coalitions or maximal losing coalitions, with each of those sets always forming an 'antichain' under the $\subseteq$-relation on coalitions of players:[10] No pair of coalitions in a set of minimal winning (maximal losing) coalitions can be comparable with respect to $\subseteq$, because then one of these coalitions would not be *minimal* winning (*maximal* losing). An exact algorithm that solves this problem must therefore search for the antichain that represents a game (as either a list of MWCs or MLCs) that has a power index closest to the target power index. In either case, a simple exact algorithm for this problem would be one that considers every possible antichain, and computes for each antichain the power index for the game that the antichain represents, and its distance to the target power index, to finally return a game that minimizes this distance.

The number of antichains on a set of n elements is known as the *nth Dedekind number D_n*. Because the sequence of Dedekind numbers (D_n) quickly grows very large, this algorithm has high time complexity. Kleitman and Markowski (1975) prove the following bounds on D_n:

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

9. However, the literature does provide us with bounds on the maximum weight necessary in an integer representation of a weighted voting game, and we could utilize this in order to come up with an enumeration algorithm based on generating a finite set of integer weighted representations. We elaborate on this idea in refwvgdesign.

10. A family of sets is an *antichain* with respect to some relation R, if and only if no pair of sets in the family is comparable with respect to R.

where c' and c are constants and E_n is the size of the largest antichain on an n-set.[11] Sperner (1928) proves that $E_n = \binom{n}{\lfloor n/2 \rfloor}$, a result known as *Sperner's theorem.*

From Sperner's theorem and Stirling's approximation, we get

$$E_n \in \Theta\left(\frac{2^n}{\sqrt{n}}\right). \tag{3}$$

We conclude that D_n is doubly exponential in n. Therefore, any algorithm that solves the voting game design problem for monotonic simple games in this way achieves a running time in $\Theta^*(2^{2^n} \cdot h(n))$. The function h is exponential for all popular power indices, e.g., the Shapley-Shubik index and the Banzhaf index (see Aziz, 2008; Deng & Papadimitriou, 1994; Prasad & Kelly, 1990).

4.2 Enumerating Weighted Voting Games

As mentioned in Section 3, the literature on voting game design problems has focused on the weighted voting game variant of the power index voting game design problem where the power index of choice is either the Banzhaf index or the Shapley-Shubik index. Here, we propose an exact algorithm for this problem that runs in exponential time. What will turn out to make this algorithm interesting for practical purposes is that it is an anytime algorithm (trivially): it is guaranteed to output an optimal solution eventually, but we can stop execution of this algorithm at any time, and the answer output will be closer to the optimum, the longer we run it. The advantage of this algorithm over the current local search methods is obviously that we will not get stuck in local optima.

All of the existing local search methods that try to solve the weighted voting game design problem use the weighted representation directly, but as mentioned before, there exist infinitely many weighted representations for even a single weighted voting game, so that such algorithms stall if they move to a different weighted representation that doesn't represent a different game. As we mentioned before, using weighted representations as a basis for an enumeration algorithm is a possibility as well, but it is not immediately clear how to do this, as there is an infinite set of weighted representations for every weighted voting game. Muroga (1971, Thm. 9.3.2.1) provides us with a solution to this problem, as his theorem tells us that for every weighted voting game there exists an integer weighted representation where none of the weights nor the quota exceeds $2^{n \log n}$. This means that a possible enumeration algorithm could work by iterating over all of the $2^{(n+1)n \log n}$ integer weight vectors that have weights that fall within these bounds, and output a weight vector in case it corresponds to a weighted voting game that has not been output before. This yields an improvement over the enumeration algorithm outlined in Section 4.1 for the special case of weighted voting games. But we still do not consider this a satisfactory enumeration procedure because the runtime of this algorithm is still significantly larger than the known upper bounds on the number of weighted voting games (see Appendix A). The enumeration

11. Korshunov (2003) devised an asymptotically equal expression:

$$D_n \sim 2^{C(n)} e^{c(n)2^{-n/2} + n^2 2^{-n-5} - n2^{-n-4}},$$

with $C(n) = \binom{n}{\lfloor n/2 \rfloor}$ and $c(n) = \binom{n}{\lfloor n/2 \rfloor - 1}$. He describes this expression as the number of monotonic boolean functions, which is equal to the nth Dedekind number.

algorithm that we propose below has a better runtime, and indeed has the property that it is also efficient in the sense that it runs in time polynomial in the number of weighted voting games it outputs. Our algorithm does not rely on weighted representations of weighted voting games; instead it works by representing weighted voting games by their sets of minimal winning coalitions.

4.2.1 A New Structural Property for the Class of Weighted Voting Games

The enumeration algorithm we propose enumerates canonical WVGs and exploits the fact that the set of canonical weighted voting games on n players is partially ordered by a specific relation on these games' sets of minimal winning coalitions. In particular, we will show that if we take any canonical weighted voting game on n players with at least one minimal winning coalition (MWC), then there exists an MWC C in this game's set of MWCs such that, if we remove C, the resulting set of MWCs represents another canonical weighted voting game on n players. In our analysis, we will also show how we can check whether the set of MWCs of a given game can be extended with some coalition to form another game's set of MWCs. This way, we can start from the game with zero MWCs, and enumerate all games 'from the bottom up', according to a specific partial order.

We proceed by developing some necessary theory behind the algorithm that we will propose. We will focus only on the class of *canonical* weighted voting games since for each noncanonical weighted voting game there is a canonical one that can be obtained by merely permutating the players.

We use some order-theoretic notions in this section. These are given in the following definition.

Definition 7 (Partial order, (graded) poset, cover, rank function, least element)**.** *For a set S, a* partial order *$\preceq$ is a relation on S that is* reflexive*, so $\forall x \in S : x \preceq x$;* antisymmetric*, so $\forall x, y \in S : ((x \preceq y \wedge y \preceq x) \rightarrow x = y)$; and* transitive*, so $\forall x, y, z \in S : ((x \preceq y \wedge y \preceq z) \rightarrow x \preceq z)$. A* partially ordered set *or* poset *is a set S equipped with a partial order $\preceq$, i.e., a pair $(S, \preceq)$. A* least element *of a poset $(S, \preceq)$ is an element $x \in S$ such that $x \preceq y$ for all $y \in S$. A* minimal element *of $(S, \preceq)$ is an element $x \in S$ such that $y \preceq x$ implies $y = x$ for all $y \in S$. We say that $y \in S$* covers *$x \in S$ in $(S, \preceq)$ when $x \preceq y$ and there is no $z \in S$ such that $x \preceq z \preceq y$. A poset $(S, \preceq)$ is* graded *when there exists a* rank function *$\rho : S \rightarrow \mathbb{N}$ such that: (i) ρ is constant on all minimal elements of $(S, \preceq)$, (ii) $\rho(x) \leq \rho(y)$ for all $x, y \in S$ for which $x \preceq y$ holds, and (iii) for any pair $x, y \in S$ it holds that if y covers x in $(S, \preceq)$, then $\rho(y) = \rho(x) + 1$.*

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

Definition 8 (The relation $\preceq_{\mathsf{MWC}}$)**.** *Let $G, G' \in \mathcal{G}_{\mathsf{cwvg}}(n)$ be any two canonical weighted voting games. We define $G \preceq_{\mathsf{MWC}} G'$ to hold if and only if there exists for some $k \in \mathbb{N}_{\geq 1}$ a sequence $G_1, \ldots, G_k$ of canonical weighted voting games of n players, such that $G_1 = G$, $G_k = G'$, and for all $1 \leq i < k$ it holds that $W_{\min,i} \subset W_{\min,i+1}$ and $|W_{\min,i}| = |W_{\min,i+1}| - 1$, where $W_{\min,i}$ denotes the set of minimal winning coalitions of G_i.*

The following theorem provides the foundation for our enumeration algorithm.

Theorem 1. *For each n, $(\mathcal{G}_{\mathsf{cwvg}}(n), \preceq_{\mathsf{MWC}})$ is a graded poset with rank function*

$$\begin{array}{rccc} \rho & : & \mathcal{G}_{\mathsf{cwvg}}(n) & \to & \mathbb{N} \\ & & G & \mapsto & |W_{\min,G}|. \end{array}$$

Moreover, this graded poset has a least element of rank 0.

Proof of Theorem 1. We need to prove (1) that the pair $(\mathcal{G}_{\mathsf{cwvg}}(n), \preceq_{\mathsf{MWC}})$ is a poset, which requires showing that the relation $\preceq_{\mathsf{MWC}}$ is a partial order on $\mathcal{G}_{\mathsf{cwvg}}(n)$, (2) that the partial order is graded, which requires showing that the function ρ is a rank function, and (3) that the graded poset has a least element of rank 0. As for (1), the relation $\preceq_{\mathsf{MWC}}$ is *reflexive*, because for every game $G \in \mathcal{G}_{\mathsf{cwvg}}(n)$, $G \preceq_{\mathsf{MWC}} G$ holds: we can take $k = 1$ in Definition 8, and the required 'sequence' of games is just G. The relation is also *antisymmetric*, because if $G \preceq_{\mathsf{MWC}} G'$ and $G' \preceq_{\mathsf{MWC}} G$ are true, then there exist two sequences of CWVGs $G_1, \ldots, G_m$ (with $G_1 = G$ and $G_m = G'$), and $G_1, \ldots, G_n$ (with $G_1 = G'$ and $G_n = G$), for which the conditions in Definition 8 hold. Neither sequence can have a length > 1, because that would mean that $W_{\min,G} \subset W_{\min,G'}$ (or vice versa), and then the inclusion wouldn't hold in the other direction. This means we have $m = n = 1$, so that $G = G'$. Finally, the relation is *transitive*, because if $G \preceq_{\mathsf{MWC}} G'$ and $G' \preceq_{\mathsf{MWC}} G''$ hold, there exists sequences of CWVGs $G_1, \ldots, G_m$ (with $G_1 = G$ and $G_m = G'$) and $G_1, \ldots, G_n$ (with $G_1 = G'$ and $G_n = G''$) for which the conditions in Definition 8 hold. Concatenating these sequences establishes that $G \preceq_{\mathsf{MWC}} G''$ holds.

In order to prove (2), that the poset is *graded* under the rank function ρ specified in the theorem, we have to show that the three conditions (i) through (iii) from Definition 7 hold for the function ρ defined in Theorem 1. As for condition (i), the CWVG G with $W_{\min,G} = \varnothing$ is a minimal element: take an arbitrary CWVG G' for which $G' \preceq_{\mathsf{MWC}} G$ holds. We will show that $G' = G$. Because $G' \preceq_{\mathsf{MWC}} G$ holds, there exists a sequence of CWVGs $G_1, \ldots, G_m$ (with $G_1 = G'$ and $G_m = G$) for which the conditions in Definition 8 hold. But because $W_{\min,G} = \varnothing$, the sequence can not have a length > 1, so $G' = G$. Because G' was arbitrary, this holds for all CWVGs, establishing that G is a minimal element. The value of the rank function for this game G is 0. To show that the game G with $W_{\min,G} = \varnothing$ is the *only* minimal element, we will prove the following lemma constructively.

Lemma 1. *For every game $G \in \mathcal{G}_{\mathsf{cwvg}}(n)$ with a nonempty set $W_{\min,G}$ as its set of minimal winning coalitions, there is a coalition $C \in W_{\min,G}$ and a game $G' \in \mathcal{G}_{\mathsf{cwvg}}(n)$ so that $W_{\min,G} \setminus \{C\}$ is the set of minimal winning coalitions of G'.*

This lemma implies that for every game $G \in \mathcal{G}_{\mathsf{cwvg}}$, there exists a sequence of CWVGs that starts at G and ends at the game with no minimal winning coalitions. Each game has a set of MWCs that is one smaller than the previous game in the sequence. To prove Lemma 1, we first prove two preliminary Lemmas (2 and 3).

Lemma 2. *Let $G = (N = \{1, \ldots, n\}, v)$ be a weighted voting game, and let $\ell = [q; w_1, \ldots, w_n]$ be a weighted representation for G. For each player i there exists an $\epsilon > 0$ such that for all positive $\epsilon' < \epsilon$, the vector $\ell' = [q; w_1, \ldots, w_i + \epsilon', w_{i+1}, \ldots, w_n]$ is also a weighted representation for G.*

Informally, this lemma states that it is always possible to increase the weight of a player by some amount without changing the game.

Proof. Recall that W_G and L_G are G's sets of winning and losing coalitions, respectively. Let $w_{\max} = \max_{C \in L_G} w(C)$ and $w_{\min} = \min_{C \in W_G} w(C)$, so $w_{\max} < q$ is the largest weight of a losing coalition, $w_{\min} \geq q$ is the smallest weight of a winning coalition, and $w_{\max} < w_{\min}$. Take $\epsilon = w_{\min} - w_{\max}$ and note that $\epsilon > 0$. Increasing any player's weight by any positive amount ϵ' that is less than ϵ does not turn any losing coalition into a winning coalition. Also, as $\epsilon' > 0$, this change of weight does not turn any winning coalition into a losing coalition, so the weighted representation ℓ' represents the same game G. □

Lemma 3. *Let $G = (N = \{1, \ldots, n\}, v)$ be a weighted voting game, and let $w_\ell(C)$ be the weight of coalition C in the game represented by ℓ. There exists a weighted representation ℓ for G such that for all $C, C' \in 2^N$, $C \neq C'$, for which $v(C) = v(C') = 1$, it holds that $w_\ell(C) \neq w_\ell(C')$.*

Or, informally stated: for every weighted voting game, there exists a weighted representation such that all winning coalitions have a different weight.

Proof. Let $\ell' = [q; w_1, \ldots, w_n]$ be some weighted representation for G. We will construct the required weighted representation ℓ from ℓ'. First fix an arbitrary player i. By Lemma 2, there is an $\epsilon > 0$ such that increasing w_i by any value $\epsilon' \in (0, \epsilon)$ will result in another weighted representation for G. Let $\mathcal{E}$ be the set of choices for ϵ' such that increasing w_i by ϵ' yields a weighted representation ℓ where there are two coalitions $C, C' \in 2^N$, such that $i \in C$ and $i \notin C'$, that have the same weight under ℓ. There are finitely many such pairs (C, C') so $\mathcal{E}$ is finite and therefore $(0, \epsilon) \setminus \mathcal{E}$ is non-empty. By picking for ϵ' any value in $(0, \epsilon) \setminus \mathcal{E}$, and increasing the weight of player i by ϵ', we thus end up with a weighting ℓ in which there is no coalition C containing i such that $w_\ell(C)$ is equal to any coalition not containing i. Furthermore, if C and C' are two arbitrary coalitions that have distinct weights under ℓ', then certainly they will have distinct weights under ℓ.

By sequentially applying the above operation for all $i \in N$, we end up with a weighting ℓ for which it holds for every player i that there is no coalition C containing i such that $w_\ell(C)$ is equal to the weight of any coalition not containing i.

Let $C, C' \in 2^N$ be arbitrary distinct coalitions. Assume without loss of generality that $C \setminus C' \neq \varnothing$ (otherwise, we may swap the names of C and C') and let $i \in C \setminus C'$. Because C contains i and C' does not contain i, we have $w_\ell(C) \neq w_\ell(C')$, and this completes the proof. □

Using Lemma 3, we can prove Lemma 1, which establishes Theorem 1.

Proof of Lemma 1. Let $G = (\{1, \ldots, n\}, v)$ be a canonical weighted voting game. Let $W_{\min,G}$ be its set of minimal winning coalitions and let $\ell = [q; w_1, \ldots, w_n]$ be a weighted representation for which it holds that all minimal winning coalitions have a different weight. By Lemma 3, such a representation exists. We will construct an ℓ'' from ℓ for which it holds that it is a weighted representation of a canonical weighted voting game with $W_{\min,G} \setminus \{C\}$ as its list of minimal winning coalitions, for some $C \in W_{\min,G}$.

Let i be the highest-numbered player that is in a coalition in $W_{\min,G}$, i.e., i is a least desirable nondummy player. We may assume without loss of generality that $w_j = 0$ for all $j > i$ (i.e., the dummy players), as after setting these weights to 0, it still holds that all *minimal* winning coalitions have different weights. Let $C \in W_{\min,G}$ be the minimal winning coalition containing i with the lowest weight among all MWCs in $W_{\min,G}$ that contain i.

Next, define ℓ' as $[q; w_1, \ldots, w_i - (w_\ell(C) - q), \ldots, w_n]$. Note that under ℓ' the weights of the players are still decreasing and nonnegative: Because $w_\ell(C \setminus \{i\}) < q$ (due to $C \setminus \{i\}$ being a losing coalition, because C is a MWC), $w_\ell(C \setminus \{i\}) + w_i = w_\ell(C) < q + w_i$, so $w_i > w_\ell(C) - q$. So player i's weight under ℓ' is indeed nonnegative. Player i's weight is decreased in ℓ', but not by enough to turn even the lightest of all MWCs that contain i into a losing coalition. So now $G_{\ell'} = G_\ell = G$ and $w_{\ell'}(C) = q$. Moreover, the weights of the coalitions in $W_{\min,G}$ that contain player i are still mutually distinct under ℓ'.

We now decrease player i's weight further, by an amount that is so small that the only minimal winning coalition that turns into a losing coalition is C. Note that under ℓ' (representing the same game G), minimal winning coalition C is still the lightest MWC containing player i. Let $C' \in W_{\min,G}$ be the second-lightest minimal winning coalition containing i. Obtain ℓ'' by decreasing i's weight under ℓ' by a positive amount smaller than $w_{\ell'}(C') - w_{\ell'}(C)$. Coalition C will become a losing coalition and all other minimal winning coalitions will stay winning. No new minimal winning coalition is introduced in this process: Suppose there would be such a new minimal winning coalition S in $G_{\ell''}$, then S contains only players that are at least as desirable as i (the other players have weight 0). In case $i \notin S$, we have that S would also be a minimal winning coalition in the original game G because $w_{\ell''}(S) = w_\ell(S) \geq q$, which contradicts the fact that S is a *new* MWC in $G_{\ell''}$. In case $i \in S$, it must be that $S \setminus \{i\}$ is winning in the original game G (because S is winning in the original game and S is not a MWC in the original game, and i was picked to be a least desirable nondummy player). Thus, $w_{\ell''}(S \setminus \{i\}) = w_\ell(S \setminus \{i\}) \geq q$, and this is in contradiction with S being an MWC in $G_{\ell''}$.

$G_{\ell''}$ is therefore an n-player canonical weighted voting game whose set of MWCs forms a subset of the MWCs of G, and the cardinalities of these two sets differ by exactly 1. This proves the claim. □

Continuing the proof of Theorem 1, it follows from Lemma 1 that the game with no minimal winning coalitions is the *unique* minimal element. The fact that properties (*ii*) and (*iii*) hold for the rank function ρ follows immediately from the definitions of the relation $\preceq_{\mathsf{MWC}}$ and the function ρ. Regarding claim (3), that the poset has a least element with rank 0, consider the game G with no minimal winning coalitions. For this game G, $G \preceq_{\mathsf{MWC}} G'$ holds for all games $G' \in \mathcal{G}_{\mathsf{cwvg}}(n)$, so G is a least element, and it has rank 0. □

In Figure 2, $(\mathcal{G}_{\mathsf{cwvg}}(4), \preceq_{\mathsf{MWC}})$ is depicted graphically. Note that for convenience of display, this figure is *not* exactly the Hasse diagram of the poset $(\mathcal{G}_{\mathsf{cwvg}}(4), \preceq_{\mathsf{MWC}})$. It is clear from Figure 2 that $(\mathcal{G}_{\mathsf{cwvg}}(4), \preceq_{\mathsf{MWC}})$ is not a tree since there is one game that covers multiple games: it is the game labelled '0110' that has two edges coming into it, one of which is dashed. From the set of minimal winning coalitions representing this game, there are *two* minimal winning coalitions (1001 and 0110) for each of which it holds that if it is removed, the remaining set of coalitions is the set of minimal winning coalitions representing another

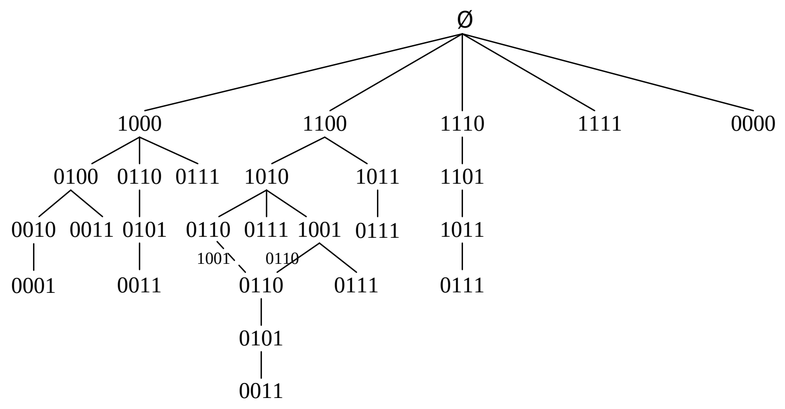

Figure 2: Graphical depiction of $(\mathcal{G}_{\mathsf{cwvg}}(4), \preceq_{\mathsf{MWC}})$. Each node in this graph represents a canonical weighted voting game of four players. The figure should be read as follows: each node has the 'characteristic vector' of a minimal winning coalition as a label; a binary sequence indicating for each player whether it is (1) or is not (0) a member of the coalition. The set of minimal winning coalitions of a game that corresponds to a certain node n in the graph, has as its elements those coalitions that are described by the elements of the set V_n of characteristic vectors on the path from the top node to n along the solid edges. The top node corresponds to the canonical weighted voting game with zero minimal winning coalitions (i.e., every coalition loses). The actual Hasse diagram of this poset can be obtained by changing the label of each node n to V_n and including the solid edges as well as the dashed edge in the diagram.

canonical weighted voting game. Since one could add any number of dummy players to this game (i.e., players that do not occur in any winning coalition), we conclude the following.

Proposition 1. *For every $n \geq 4$, $(\mathcal{G}_{\mathsf{cwvg}}(n), \preceq_{\mathsf{MWC}})$ is not a tree.*

When we develop our algorithm in the next section, it will turn out that Proposition 1 makes things significantly more complicated.

4.2.2 The Algorithm

We use the results from the previous section to develop an exponential-time exact algorithm for the power index voting game design problem. The way this algorithm works is straightforward. Just as in the naive algorithm suggested in Section 4.1, we enumerate the complete class of games (weighted voting games in this case), and we compute for each game that is output by the enumeration algorithm its power index and the distance of this power index to the target power index. We keep track of the game that minimizes this distance.

The key to efficient enumeration is that by utilizing Theorem 1, it is possible to efficiently generate the minimal winning coalition listings of the canonical weighted games of rank i (according to the graded poset just defined) from the minimal winning coalition listings of the canonical weighted voting games of rank $i - 1$.

The following two theorems show us how to do this. Firstly, we need a result from the work of Peled and Simeone (1985).

Theorem 2 (Peled & Simeone, 1985)**.** *There exists a polynomial time algorithm for testing whether a game given as a list of minimal winning coalitions is a weighted voting game. Moreover, if it is a weighted voting game, then its list of maximal losing coalitions and a weighted representation can be found in polynomial time.*

The algorithm proposed by Peled and Simeone (1985) that has the above-named characteristics is called the *Hop-Skip-and-Jump* algorithm. The algorithm was originally designed for applications related to solving problems in the fields of threshold logic and set covering, but it is only a matter of changing terminology in a straightforward way to see that the Hop-Skip-and-Jump algorithm fulfills our purposes as well.

To state the next theorem, we first define the *truncation operation.*

Definition 9 (Right-truncation)**.** *Let $S \subseteq N = \{1, \dots, n\}$ be a coalition of players. Using $P(S, i)$ to denote the ith highest numbered player among the players in S, the i*th right-truncation *of S, denoted $\mathsf{rtrunc}(S, i)$, is defined as*

$$\mathsf{rtrunc}(S, i) = \begin{cases} S & \text{if } i = 0, \\ S \setminus \{P(S, i), \dots, n\} & \text{if } 0 < i \leq |S|, \\ \textit{undefined} & \textit{otherwise.} \end{cases}$$

In canonical linear games, the ith right-truncation of a coalition S (for $i \leq |S|$) is the coalition that remains when the i least desirable players are removed from S. For example, the 2nd right-truncation of the coalition $\{1, 2, 4, 6, 8\}$ is the coalition $\{1, 2, 4\}$.

Theorem 3. *For any n, let $G, G' \in \mathcal{G}_{\mathsf{clin}}(n)$ be a pair of canonical linear games with respective sets of minimal winning coalitions $W_{\min,G}$ and $W_{\min,G'}$, such that $W_{\min,G} \subseteq W_{\min,G'}$ and $|W_{\min,G'}| = |W_{\min,G}| + 1$. 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 $W_{\min,G'} = W_{\min,G} \cup \{\mathsf{rtrunc}(C, i)\}$.*

Proof. Let C' be the coalition such that $W_{\min,G'} = W_{\min,G} \cup \{C'\}$. Coalition C' cannot be a superset of a coalition in $W_{\min,G}$ because then it would not be a *minimal* winning coalition in G'. Therefore, C' is a losing coalition in 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 $C \in L_{\max,G}$. So there is a $C \in L_{\max,G}$ such that C' is a subset of C, but not a right-truncation of C. This means that in C', some player j from C is not present, while at least one less desirable player $k > j$ from C is in C'. This implies that there is a left-shift C'' of C' such that C'' is a subset of a coalition in $L_{\max,G}$: C'' is obtained from C' by replacing j by k. Because C'' is a subset of C, C'' is still a losing coalition in G. So C'' is not a superset of any coalition in $W_{\min,G}$, and 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 linear game, so by the desirability relation $1 \succeq \cdots \succeq n$, C'' *is a winning coalition in* G' because it is a left-shift of the winning coalition C'. This is a contradiction. □

From Theorems 2 and 3, it now becomes apparent how to use $(\mathcal{G}_{\mathsf{cwvg}}(n), \preceq_{\mathsf{MWC}})$ for enumerating the class of n-player canonical weighted voting games. We start by outputting the n-player weighted voting game with zero minimal winning coalitions. After that, we repeat the following process until we find no further games: We use Theorem 3 to generate the minimal winning coalition lists of all canonical weighted voting games with i minimal winning coalitions, using the set of canonical weighted voting games games with $i-1$ minimal winning coalitions (also represented as a list of minimal winning coalitions), starting at $i = 1$. Once generated, we have the choice to output the games either as a list of minimal winning coalitions or in a weighted representation, by using the Hop-Skip-and-Jump algorithm.

Generating the set of games of i minimal winning coalitions works as follows. For each game G with $i-1$ minimal winning coalitions, we obtain the set of maximal losing coalitions by using the Hop-Skip-and-Jump algorithm. Next, we check for each right-truncation of each maximal losing coalition $C \in L_{\max,G}$ whether, when we add it to G's set of minimal winning coalitions, the resulting set represents another weighted voting game. Again, testing whether a game is a weighted voting game is done by using the Hop-Skip-and-Jump algorithm. If a game turns out to be weighted, we store and output it.

There is one remaining problem with this approach: it outputs duplicate games. If $(\mathcal{G}_{\mathsf{cwvg}}(n), \preceq_{\mathsf{MWC}})$ were a tree, then this would not be the case. However, by Proposition 1, $(\mathcal{G}_{\mathsf{cwvg}}(n), \preceq_{\mathsf{MWC}})$ is not a tree for any $n \geq 4$. Thus, we have to do a *duplicates check* for each weighted voting game we find. In principle, this seems not to be so difficult. For each game we find, we can sort its list of minimal winning coalitions, and check if this list of coalitions already occurs in the array of listings of minimal winning coalitions that correspond to games that we already found. The problem with this solution is that the list can grow very large, making these checks very time- and space-consuming operations.

We will therefore use a different method for doing this "duplicates check": Suppose we have found an n-player canonical weighted voting game G of i minimal winning coalitions by adding a coalition C to a minimal winning coalition listing of a canonical weighted voting game that we have already found. We first sort G's list of minimal winning coalitions according to some fixed total order. Then we check for each coalition C' that occurs before C in this sorted list, whether C''s removal from the list results in a list of minimal winning coalitions of a canonical weighted voting game. If there is such a C', then we discard G, and otherwise, we keep it. This way, it is certain that each canonical weighted voting game will be generated only once. In terms of orderly generation of combinatorial objects (McKay, 1998), our method thus provides a canonical construction path for CWVG's.

Algorithm 1 gives the pseudocode for this enumeration method. The array element $\mathsf{games}[i]$ will be the list of canonical weighted voting games that have i minimal winning coalitions; the rank-i games. The value of i can not exceed $\binom{n}{\lfloor n/2 \rfloor}$ by Sperner's Theorem. The games are represented as lists of minimal winning coalitions. The algorithm iterates from every new game found, starting with the game in $\mathsf{games}[0]$, with zero minimal winning coalitions.

Algorithm 1 An enumeration algorithm for the class of n player canonical weighted voting games. hopskipjump refers to the Hop-Skip-and-Jump algorithm.

```
1: Output $[1; 0, \dots, 0]$                                  ▷ Output the game with no MWCs
2: games$[0] \leftarrow \{\varnothing\}$                       ▷ Add this game to the list of rank-0 games
3: for $i = 1$ to $\binom{n}{\lfloor n/2 \rfloor}$ do
4:     for all $G \in$ games$[i-1]$ do                          ▷ Evaluate each game $G$ of rank $i-1$
5:         $L_{\max} \leftarrow$ hopskipjump$(W_{\min,G})$      ▷ Obtain game $G$'s set of MLCs
6:         for all $C \in L_{\max}$ do                          ▷ Evaluate each MLC $C$
7:             for $j = 0$ to $n$ do                            ▷ Evaluate each of $C$'s right-truncations
8:                 $G' \leftarrow W_{\min,G} \cup$ rtrunc$(C, j)$   ▷ Call the game to be evaluated $G'$
9:                 if isweighted$(G')$ then                     ▷ This requires solving an LP
10:                    if $G'$ passes the duplicates check then
11:                        Output a weighted representation of $G'$.
12:                        Append $G'$ to games$[i]$.
13:                    end if
14:                end if
15:            end for
16:        end for
17:    end for
18: end for
```

Correctness of the algorithm follows from our discussion above. Next, we analyze the time-complexity of the algorithm.

Theorem 4. *Algorithm 1 runs in $O^*(2^{n^2+2n})$ time.*

Proof. Lines 5 to 16 are executed at most once for every canonical weighted voting game. From Sperner's Theorem, we know that any list of minimal winning coalitions has fewer than $\binom{n}{\lfloor n/2 \rfloor}$ elements. So by the runtime of the Hop-Skip-and-Jump algorithm, line 5 runs in time $O\left(n\binom{n}{\lfloor n/2 \rfloor}^2 + n^3\binom{n}{\lfloor n/2 \rfloor}\right) = O(n^2\sqrt{n}2^{2n})$. Within an iteration of the outer loop (line 4), lines 9 to 14 are executed at most $n\binom{n}{\lfloor n/2 \rfloor} = O(\sqrt{n}2^n)$ times (because $L_{\max}$ is also an antichain, Sperner's Theorem also applies for maximal losing coalitions). The time-complexity of one execution of lines 9 to 14 is as follows.

- At line 9, we must solve a linear program, taking time $O\left(n^{4.5}\binom{n}{\lfloor n/2 \rfloor}\right) = O(n^4 2^n)$ using Karmarkar's interior point algorithm (Karmarkar, 1984).
- At line 10, we must execute the duplicates check. This consists of checking for at most $\binom{n}{\lfloor n/2 \rfloor}$ sets of minimal winning coalitions whether they are weighted. This involves running the Hop-Skip-and-Jump algorithm, followed by solving a linear program. In total, this takes $O(n^3\sqrt{n}2^{2n})$ time.
- Lines 11 and 12 take linear time.

Bringing everything together, we see that a single pass from lines 5 to 16 costs us $O(n^4 2^{3n})$ time. As mentioned earlier, these lines are executed at most $|\mathcal{G}_{\mathsf{cwvg}}(n)|$ times. From Corollary 1 in Appendix A we know that $|\mathcal{G}_{\mathsf{wvg}}(n)| \in O(2^{n^2-n})$, and of course $|\mathcal{G}_{\mathsf{cwvg}}(n)| < |\mathcal{G}_{\mathsf{wvg}}(n)|$, hence lines 5 to 16 are executed at most $O(2^{n^2-n})$ times, and therefore the runtime of the algorithm is $O(2^{n^2+2n}n^4) = O^*(2^{n^2+2n})$. □

Although the runtime analysis of Algorithm 1 is not very precise, we want to emphasize that this method runs in exponential time instead of doubly exponential time. We can also show that the runtime of this algorithm is only polynomially greater than the amount of data output. This implies that Algorithm 1 is essentially the fastest possible enumeration algorithm for canonical weighted voting games, up to a polynomial factor.

Theorem 5. *Algorithm 1 runs in output-polynomial time, i.e., a polynomial in the number of bits that Algorithm 1 outputs.*

Proof. Lines 5 to 16 are executed less than $|\mathcal{G}_{\mathsf{cwvg}}(n)|$ times. From (5), 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 5 to 16 costs $O(n^4 2^{3n})$ time, and thus one iteration runs in

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

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

Remark 1. *We cannot give a very sharp bound on the space complexity of Algorithm 1 because we do not know much about the maximum cardinality of an antichain in $(\mathcal{G}_{\mathsf{cwvg}}(n), \preceq_{\mathsf{MWC}})$. (Nonetheless, we did obtain the maximum sizes of the antichains for $n \leq 8$; see Figure 3 in the next section.) However, it can be seen that it is also possible to generate the games in this poset in a depth-first manner, instead of in a breadth-first manner like we do now. In that case, the amount of space that needs to be used is bounded by the maximum length of a chain in $(\mathcal{G}_{\mathsf{cwvg}}(n), \preceq_{\mathsf{MWC}})$. This is a total amount of $O(\frac{2^n}{\sqrt{n}})$ space.*

We will now briefly illustrate how the algorithm enumerates (four-player) CWVG's by referring to Figure 2, and afterwards indicate the application to solving the power index weighted voting game design problem. The algorithm starts in line 1 by outputting a weighted representation of the game at the root node. This is the game that has no winning coalitions. (By our definition of simple game, it exists.) This root game's list of MWC's is $\varnothing$, and it is added to the array element games[0] in line 2. Now we loop through all possible values i the rank function ρ defined in Theorem 1 can take: $1 \leq i \leq \binom{n}{\lfloor n/2 \rfloor}$ (see Section 4.1). For four players, these values for $1 \leq i \leq 6$ correspond to the six horizontal rank-levels below the root node in the graph in Figure 2. For every game G in the set of games one level higher in the graph (line 4), we consider the set of maximal losing coalitions $L_{\max,G}$ (line 6), and for each of those (line 7), we evaluate whether the game obtained by adding each of its (at most) n right truncations (line 8) to $W_{\min,G}$ yields a weighted voting game (line 9). In the four player case, when $i = 1$, we look at the single root game. Since this game has only losing coalitions, the grand coalition is the only element of this game's set of MLC's. Now we evaluate whether, for each of this grand coalition's five right truncations ($0 \leq j \leq 4$), adding it to $\varnothing$ yields a CWVG. It turns out that this is true in all five cases, yielding the five CWVG's G with $\rho(G) = 1$ that we see at rank-level 1 (the second level from the top of the graph). These games are added to games[1]. Now we increase i by one, and iteratively consider these five games in turn.

To apply this enumeration algorithm to the power index weighted voting game design problem, we just compute the Banzhaf index for every generated game, and compute its distance to the target power index (according to a chosen distance function). We store and update the weighted representation for the game that minimizes this distance function, and output that representation after all games have been generated.

4.3 Improvements and Optimizations

Algorithm 1 is in its current state not quite suitable for solving the weighted voting game design problem in practice. In this section, we will describe several improvements to the algorithm. These improvements will result in a version of the enumeration algorithm which we expect to output canonical weighted voting games at a steady rate, and give us a practically applicable anytime-algorithm for voting game design problems (defined on small numbers of players).

The Hop-Skip-and-Jump algorithm of Peled and Simeone (1985) works by first generating the list of maximal losing coalitions of a game, given the list of minimal winning coalitions of a game, and subsequently solving a linear program to check whether the game is a weighted voting game. Peled and Simeone give a first improvement by showing several ways to improve and compactify the linear program in question. This smaller linear program requires only the lists of roof coalitions and ceiling coalitions of the game.

In the next section, we present a strengthened version of Theorem 3. This strengthened version implies that it is only necessary to know a game's ceiling coalitions (instead of all maximal losing coalitions) to generate the weighted voting games that cover it. In Section 4.3.2 we give an output-polynomial time algorithm for enumerating all ceiling coalitions, given a set of roof coalitions.

By using these improvements, combined with the more compact linear program, we eliminate the need to compute the complete set of maximal losing coalitions of the weighted voting games that we enumerate. Instead, it suffices to keep track only of the sets of minimal winning coalitions and ceiling coalitions of the weighted voting games enumerated, and this will speed up our algorithm significantly.[12]

4.3.1 A Better Way of Finding New Minimal Winning Coalitions

Theorem 3 allows us to find potential minimal winning coalitions with which we can extend canonical weighted voting games to generate new ones. We will now see that we do not really need to consider every right-truncation of every maximal losing coalition. In fact, we only need to look at ceiling coalitions.

Theorem 6. *For any n, let $G, G' \in \mathcal{G}_{\mathsf{clin}}(n)$ be a pair of canonical linear games such that $W_{\min,G} \subseteq W_{\min,G'}$ and $|W_{\min,G'}| = |W_{\min,G}| + 1$. Let $W_{\min,G}$ and $W_{\min,G'}$ be the sets of minimal winning coalitions of G and G', respectively, and let $L_{\mathsf{ceil},G}$ and $L_{\mathsf{ceil},G'}$ be the sets of ceiling coalitions of G and G', respectively. There is a $C \in L_{\mathsf{ceil},G}$ and an $i \in \mathbb{N}$ with $0 \leq i \leq n$ such that $W_{\min,G'} = W_{\min,G} \cup \mathsf{rtrunc}(C, i)$.*

Proof. Let C' be the coalition such that $W_{\min,G'} = W_{\min,G} \cup \{C'\}$. By Theorem 3, C' is a right-truncation of a coalition $S \in L_{\max,G}$. Suppose for contradiction that C' is not a right-truncation of a ceiling in $L_{\mathsf{ceil},G}$. Then there is a ceiling $C'' \in L_{\mathsf{ceil},G}$ such that C' is a subset of a strict right-shift of C''. (This is because S is by definition a subset of a right-shift of some ceiling, and C' is a subset of S.) This in turn means that there is a strict left-shift T of C' such that T is a subset of a right-shift of C''. Coalition T is not a superset of any MWC in $W_{\min,G}$ because T is losing in G, and T is not a superset of C' either, because T is a strict left-shift of C'. So it follows that T *is a losing coalition in* G'. But G' is a canonical linear game, so the desirability relation $1 \succeq \cdots \succeq n$ is satisfied. Because T is a left-shift of C', and C' is winning in G', it follows that T *is a winning coalition in* G'. This is a contradiction. □

4.3.2 An Algorithm for Obtaining the Ceiling-List of a Canonical Weighted Voting Game

Using the Hop-Skip-and-Jump algorithm, we can output in polynomial time the list of maximal losing coalitions from the list of minimal winning coalitions, when given a weighted voting game. Given this, an interesting question at this point is whether we can also output in polynomial time the list of ceiling coalitions from the list of roof coalitions.

In Appendix B, we prove that this is impossible in general because the output may be exponentially sized in the input. Nevertheless, it certainly still makes sense to try to come up with as efficient as possible an algorithm for this problem, considering that such an algorithm can be used in combination with the improvements of the previous section for finding weight vectors for weighted voting games. Using such an algorithm is certainly preferred to using the Hop-Skip-and-Jump algorithm, because in a canonical linear game there are always fewer roof coalitions than minimal winning coalitions, and fewer ceiling coalitions than maximal losing coalitions.

However, a recent construction by Polyméris and Riquelme (2013) shows that an output-polynomial time algorithm for this problem would have sensational consequences, as it would

12. Related is Proposition 2.3 of Krohn and Sudhölter (1995), which gives insight in the maximum number of ceiling coalitions that a canonical linear game can have.

imply a polynomial time algorithm for the *monotone boolean duality* problem (Fredman & Khachiyan, 1996): a well-known problem that can be solved in sub-exponential time, but of which it is not known whether it admits a polynomial time algorithm.

Finding an output-polynomial algorithm for generating the ceiling set of a game from its roof set, is thus a very interesting open problem, but due to its alleged difficulty[13] we instead resort to studying the problem of outputting the ceiling set of a game from its set of minimal winning coalitions.

Of course, one could simply solve the latter problem by using the Hop-Skip-and-Jump algorithm. This would provide us with a list of MLCs of the input game, after which we could filter out the ceilings. This algorithm would run in $O(nm^2 + n^3m)$ time, where m is the number of MWCs. Below, we will provide an alternative simpler algorithm for the special case where we only need to output the ceilings of a given game. In the remainder of this section, we will use the following definitions.

Definition 10. *Let N be the set of players $\{1, \ldots, n\}$ and let $S \subseteq N$ be a coalition. The functions a and b and* trunc *are defined as follows.*

- *$b(S)$ is the highest-numbered player in S.*
- *$a(S)$ is the highest-numbered player j in $N \setminus S$ such that $j + 1 \in S$ (if such a j does not exist, then we define $a(S) = 0$).*
- *The* truncation *of S, defined by $\mathsf{trunc}(S) = S \setminus \{a(S) + 1, \ldots, n\}$.*

For example, if $N = \{1, 2, 3, 4, 5\}$ and $S = \{1, 2, 4\}$, then the highest numbered player in S is $b(S) = 4$, $N \setminus S = \{3, 5\}$, so the highest numbered player $j \in N \setminus S$ for which $j + 1 \in S$ is $a(S) = 3$, and $\mathsf{trunc}(S) = S \setminus \{4, 5\} = \{1, 2\}$. If $S = \{1, 2, 3\}$ then $N \setminus S = \{4, 5\}$, there does not exists a $j \in N \setminus S$ for which $j + 1 \in S$, so $a(S) = 0$.

Theorem 7. *Let $G \in \mathcal{G}_{\mathsf{clin}}(n)$ be a canonical linear game on players $N = \{1, \ldots, n\}$, let $W_{\min,G}$ be G's set of MWCs, let $\mathcal{C}$ be the set of ceilings of G, and let $C \in \mathcal{C}$ such that $a(C) > 0$. Then there exists an $i \in \mathbb{N}_{\geq 0}$, $1 < i \leq |C| - |\mathsf{trunc}(C)|$, such that $\mathsf{trunc}(C) \cup \{a(C)\} \cup \{a(C) + j : 2 \leq j \leq i\}$ is a minimal winning coalition.*

Proof. The coalition C is a ceiling, so it is losing, which implies that $\mathsf{trunc}(C)$ is also losing, and $\mathsf{trunc}(C) \cup \{b(C)\} \cup \{b(C) + j : 2 \leq j \leq |C| - |\mathsf{trunc}(C)|\}$ is a left-shift of C, and hence winning. Therefore there exists an $i \in \mathbb{N}_{\geq 0}, 2 \leq i \leq |C| - |\mathsf{trunc}(C)|$ such that

$$(C' := \mathsf{trunc}(C) \cup \{a(C)\} \cup \{a(C) + j : 0 \leq j \leq i\} \text{ is winning) AND}$$
$$(C'' := \mathsf{trunc}(C) \cup \{a(C)\} \cup \{a(C) + j : 2 \leq j \leq i - 1\} \text{ is losing) OR } (C'' = C').$$

By canonicity, this means that C' is a minimal winning coalition. □

From the above theorem it becomes clear how to efficiently generate the set of ceilings from the set of minimal winning coalitions: For each MWC S, it suffices to check for all $k \in \mathbb{N}_{\geq 0}, k \leq n - b(S)$ whether:

13. We thank Andreas Polyméris and Fabián Riquelme for pointing out to us the connection to the monotone boolean duality problem, as well as for pointing out an error in a preliminary version of the present paper.

- $(S \setminus \{a(S)-1\}) \cup \{a(S)\} \cup \{b(S)+j : 1 \leq j \leq k\}$ is a ceiling (in case $a(S)-1 \in S$),
- $(S \setminus \{b(S)\}) \cup \{b(S)+j : 1 \leq j \leq k\}$ is a ceiling.

This would generate all ceilings C with the property that $b(C) > 0$. There are furthermore at most n ceilings C for which it holds that $b(C) = 0$, and it is clear that such coalitions can be generated and checked straightforwardly.

The runtime of this implied algorithm is theoretically no better than the Hop-Skip-and-Jump algorithm. However, due to the simplicity of this algorithm, and due to the fact that this algorithm only finds the ceilings (of which there are in general far less than there are MWCs), we expect this algorithm to run much faster in practice, in most cases.

5. Experiments

We have implemented Algorithm 1 together with all of the optimization strategies described in Section 4.3, that allow us to work with just the ceiling coalitions, rather than all maximal losing coaltions. The programming language we used is *C*. Execution of the algorithm involves solving a large number of linear programs. To to this, we made use of the *GNU Linear Programming Toolkit* (see Makhorin, 2004). Our implementation solves the normalized Banzhaf index weighted voting game design problem. This means that for each weighted voting game that is output by our enumeration algorithm, we must invoke a procedure for computing the normalized Banzhaf index. The algorithm used for this is simply the naive brute-force approach. (In our experiments, the runtime with Banzhaf computation (to three decimal places) is only different from the runtime without it for four players or more. Up to eight players, including computation of the Banzhaf indices never increases the runtime by more than 0.6%.)

The purpose of the experiments is to gain insight on the average optimal attainable error in random instances (for small n), as well as on the number of weighted voting games as a function of the number of players and the number of minimal winning coalitions.

Experiment 1. We used the enumeration algorithm to compute for all n with $1 \leq n \leq 8$ and all m with $0 \leq m \leq \binom{n}{\lfloor n/2 \rfloor}$, the *exact* number of canonical weighted voting games of n players with m minimal winning coalitions. The results are displayed in Figure 3. Note that on the vertical axis we have a log-scale. We see that for each of these choices of n, most of the canonical weighted voting games have a relatively small number of minimal winning coalitions relative to the maximum possible number of winning coalitions, which is $\binom{n}{\lfloor n/2 \rfloor}$.

Experiment 2. For n between 1 and 7, we computed for 1000 random instances the *average optimal error*. That is, the average error that is attained out of 1000 random instances (i.e., uniform random vectors in the $(n-1)$-dimensional unit-simplex restricted to the non-increasingness constraint) when the algorithm is allowed to run to completion on these instances. We also report the worst error that is attained among these 1000 instances. The error function we used is the square root of the sum of squared errors. The reason for using this specific error measure is because it has a nice geometric interpretation: it is the Euclidean distance between the target (input) vector and the closest point in the unit simplex that is a normalized Banzhaf index of a weighted voting game.

From Figure 4, we see that the errors decrease as n gets larger. This confirms the observation by Alon and Edelman (2010) that "when the number of voters is small, it is clear that one can not closely approximate every power vector." The main result in that paper states that if the target vector has a lot of its weight concentrated in a strict subset of the players, there exists a game with a Banzhaf vector close to this target. Here we study random vectors, but we still see the general phenomenon that as the number of players increases, there exist more games and so the probability of one of those being close to the target increases. We also see that the worst case optimal error can be much worse than the average case. This is expected, as Alon and Edelman show that for all n, there exist vectors in the standard n-simplex that can not be approximated well. Our results are computed over only 1000 random instances. Therefore, these worst case optimal errors serve only as a lower bound for the worst case optimal error over all possible instances.

It should be noted that in preliminary work (De Keijzer, Klos, & Zhang, 2012), and even earlier preliminary work (De Keijzer, 2009a), additional experiments are reported: Firstly, we used our implementation (respectively an early version of our implementation) to find the number of weighted voting games on n players, for $1 \leq n \leq 8$. However, in independent work (Kurz, 2012b) that is by now already published, the authors compute the same numbers for $1 \leq n \leq 9$ (using incomparable techniques). Secondly, some experiments were performed in order to study the time performance of the algorithm and the error-convergence of the algorithm for larger numbers of players. We omit these experiments here (see De Keijzer et al., 2012, for more results) and give a short summary instead: the outcome of these experiments is not surprising and in line with the theoretical runtime analysis of the previous section. The runtime graph indeed looks like a quadratic function, when shown on a log-scale. As mentioned, computing the Banzhaf index constitutes only a fraction of the total runtime. The error-convergence of the algorithm is slow already for 15 players, and therefore the algorithm is only practical when the number of players is not too big, as expected.[14]

6. Conclusions and Future Work

In this paper, we developed an *exact* algorithm for solving power index weighted voting game design problems. First, we derived a doubly exponential algorithm for the large class of monotonic simple games. Subsequently, we showed that it is possible to obtain a (singly) exponential algorithm for the important special case of weighted voting games.

At the core of these algorithms are methods for enumerating weighted voting games based on a new partial order on the class of weighted voting games with some specific interesting properties. The enumeration algorithm resulting from this, is to the best of our knowledge the first enumeration algorithm for weighted voting games that provably runs in output-polynomial time.

The algorithm works with families of minimal winning coalitions. We demonstrated how it is possible to improve the runtime of this algorithm by showing that it suffices to work only with a subset of these minimal winning coalitions: i.e., the roof coalitions. Using this idea, we provided various techniques to improve our algorithm. Among these improvements

14. For a more detailed discussion of the results of these experiments, see the work of De Keijzer et al. (2012).

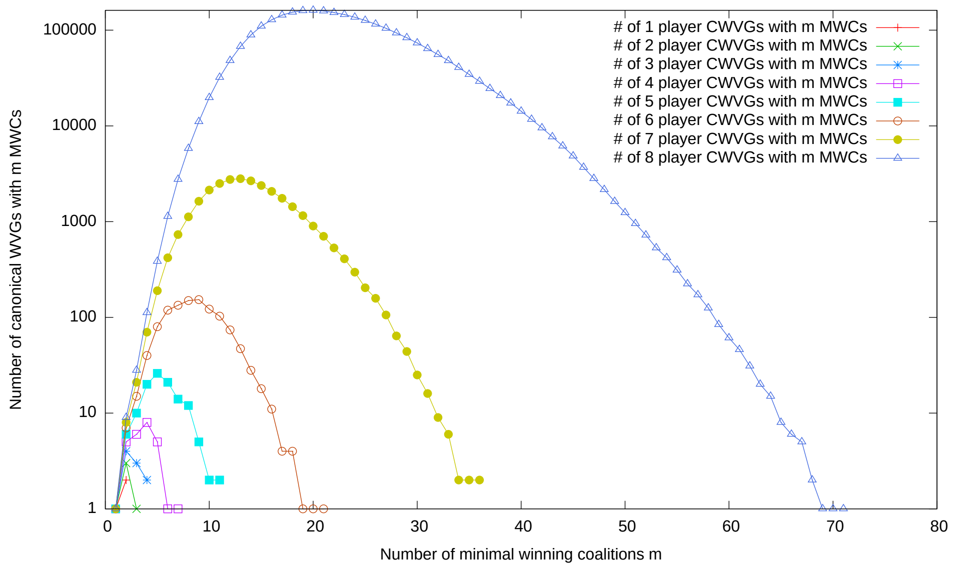

Figure 3: The number of canonical weighted voting games (y-axis) of n players, for $1 \leq n \leq 8$, with m minimal winning coalitions (x-axis).

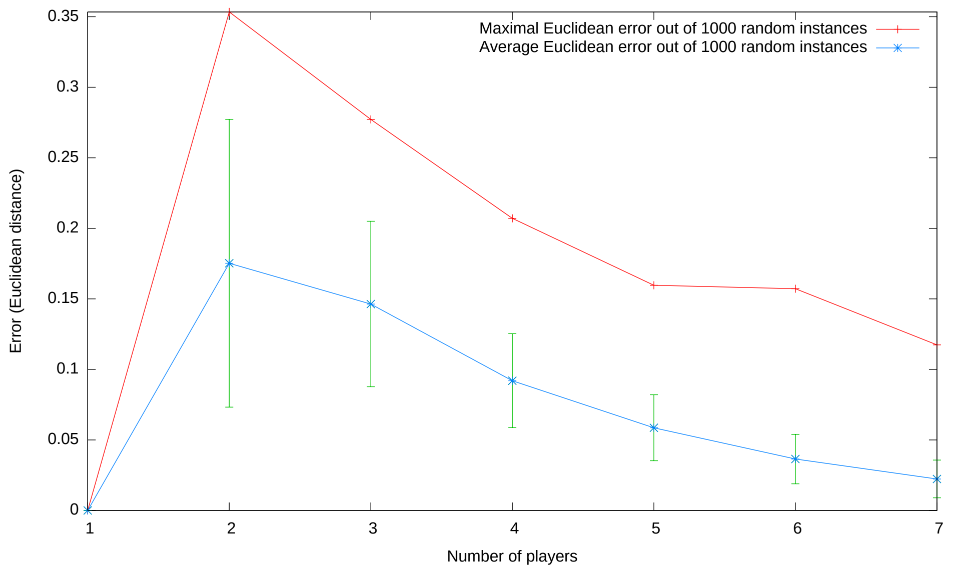

Figure 4: Optimal Euclidean error of 1000 random n player instances, for $1 \leq n \leq 7$. The error bars indicate one standard deviation.

is an output-polynomial time algorithm for outputting the list of ceiling coalitions of a linear game, given the list of roof coalitions.

In addition, we implemented the aforementioned enumeration algorithm for weighted voting games to measure its performance, obtained some interesting data about the class of weighted voting games, and validated some theoretical results related to weighted voting games.

This algorithm is based on an enumeration procedure for the class of weighted voting games: it works by simply enumerating every game, and verifying for each game whether it lies closer to the target power index than the games that we encountered up until that point. For this reason, the algorithm has the anytime-property: as we run this algorithm for a longer period of time, the algorithm enumerates more games and the quality of the solution will improve, and the algorithm is guaranteed to output the optimal solution in the long run.

The choice of the normalized Banzhaf index in our implementation in the previous section is arbitrary: the algorithm works for any choice of power index. Moreover, due to the genericity of enumeration, we can use our algorithm to solve not only power index voting game design problems, but also any other voting game design problem. The only thing we have to adapt is the error-function of the algorithm (i.e., the part of the algorithm that checks the property in question for each of the games that the enumeration procedure outputs); the enumeration procedure does not need to be changed.

As for the future work, note that in most real-life examples, the number of players in a weighted voting game is rather small: usually 10 to 50 players are involved. Thus, one of our goals is to get the proposed algorithm to yield good results within a reasonable amount of time when the number of players is somewhere in this range. It would already be interesting to be able to solve the problem for ten players, as we are not aware of any enumerations of ten player canonical weighted voting games. However, we concluded that the current implementation of the algorithm is not yet fast enough to be able to handle ten players. Optimistic extrapolation tells us that this would take tens of years; pessimistic extrapolation gives us thousands of years. However, the current implementation is not really efficient, and we have hope that with some future insights, together with careful computer programming, enumerating weighted voting games for ten players or more will be within scope. One way to improve the current algorithm is to study in more depth the partial order we introduced in this paper. One possible prospect is the following. With regard to weighted voting game design problems, we suspect that it is possible to prune a lot of "areas" in this partial order: Careful analysis of the partial order and its properties might lead to results that allow us to construct an enumeration algorithm that *a priori* discards certain (hopefully large) subsets of weighted voting games.

We are moreover interested 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 (possibly heuristic) more intelligent methods to search through the partial order. First steps in this direction are taken by Kleijn (2012). We wonder if it is possible to use such a search method while still having an optimality guarantee or approximation guarantee on the quality of the solution. In addition, we can also consider the ideas presented here as a postprocessing step to existing algorithms. In other words, it might be a good idea to first run the algorithm of Fatima et al. (2008) or Aziz et al. (2007) in order to obtain a good initial game. Subsequently, we can try to search through the "neighborhood" of the game to find improvements, according to the partial order introduced in this paper.

Lastly, some related questions for which it would be interesting to obtain an answer are about the computational complexity of the power index voting game design problem, and also about the polynomial-time-approximability of the problem. It is quite straightforward to see that the decision version of this problem is in most cases in $\mathsf{NP}^{\#\mathsf{P}}$ (and therefore in PSPACE), as one could nondeterministically guess a weight vector, and subsequently use a #P-oracle to obtain the particular power index of interest.[15] On the other hand, at the moment we do not have any ideas on how to prove hardness for this problem for any complexity class whatsoever. It seems a challenge to come up with a polynomial-time reduction from any known computational problem that is hard for any nontrivial complexity class.

Acknowledgments

We thank Fabián Riquelme and Andreas Polyméris for pointing out a problem in a preliminary version of this paper (see Section 4.3.2). We thank the anonymous referees for very extensive feedback, comments and suggestions. This is an extended version of a paper published in the proceedings of the 2010 International Conference on Autonomous Agents and Multi-Agent Systems (De Keijzer, Klos, & Zhang, 2010). This research is partially supported by the EU FET project MULTIPLEX 317532, the ERC StG Project PAAI 259515, and the Google Research Award for Economics and Market Algorithms. The majority of this research has been carried out while the first author was a master's student at Delft University of Technology. A part of this research has been carried out while the first author was a Ph.D. student at Centrum Wiskunde & Informatica (CWI), Amsterdam.

Appendix A. The Number of Weighted Voting Games

To our knowledge, the existing game theory literature does not provide us with any insights in the number of weighted voting games with n players, beyond $n = 5$. Fortunately there is a closely related field of research, called *threshold logic* (see for example Muroga, 1971), that has some relevant results.

Definition 11 (Boolean threshold function, realization, **LT**). *Let f be a boolean function on n boolean variables. The function 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 $r_1x_1 + \cdots + r_nx_n \geq r_0$ if and only if $f(x_1, \dots, x_n) = 1$. We say that r* realizes *f. We denote the set of threshold functions of n variables $\{x_1, \dots, x_n\}$ by $\mathbf{LT}(n)$.*[16]

Threshold functions resemble weighted voting games, except we talk about *boolean variables* instead of *players* now. Also, an important difference between threshold functions and weighted voting games 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 weighted voting games.

15. All power indices that have been proposed and that we have encountered are known to be in #P.
16. "LT" stands for "Linear Threshold function".

(Žunić, 2004) presents an upper bound on the number of threshold functions of n variables: $|\mathbf{LT}(n)| \leq 2^{n^2-n+1}$. Also, the following asymptotic lower bound is known (Zuev, 1989): For large enough n, we have

$$|\mathbf{LT}(n)| \geq 2^{n^2(1-\frac{10}{\log n})}. \tag{4}$$

From these bounds, we can deduce some easy upper and lower bounds for $|\mathcal{G}_{\mathsf{wvg}}|$. First we observe the following property of the set of threshold functions on n variables. Let $\mathbf{LT}^+(n)$ be the set of *non-negative threshold functions* of variables $(x_1, \ldots, x_n)$, containing those threshold functions $f \in \mathbf{LT}(n)$ for which there exists a *non-negative weight vector* r that realizes f. Then, it is clear that $|\mathcal{G}_{\mathsf{wvg}}(n)| = |\mathbf{LT}^+(n)| \leq 2^{n^2-n+1}$ for all n.

We will proceed by obtaining a lower bound on the number of weighted voting games.

Corollary 1. *For large enough n, it holds that*

$$|\mathcal{G}_{\mathsf{wvg}}(n)| \geq 2^{n^2(1-\frac{10}{\log n})-n-1}$$

Proof. Let f be a non-negative threshold function and let r be a non-negative weight vector that realizes f. There are 2^{n+1} possible ways to negate the elements of r, so there are at most $2^{n+1}-1$ threshold functions $f' \in \mathbf{LT}(n) \setminus \mathbf{LT}^+(n)$ such that f' has a realization that is obtained by negating some of the elements of r. From this, it follows that $|\mathbf{LT}^+(n)| \geq \frac{|\mathbf{LT}(n)|}{2^{n+1}}$, and thus also $|\mathcal{G}_{\mathsf{wvg}}(n)| \geq |\frac{\mathbf{LT}(n)}{2^{n+1}}|$. Now by using (4) we get $|\mathcal{G}_{\mathsf{wvg}}(n)| \geq \frac{2^{n^2(1-\frac{10}{\log n})}}{2^{n+1}} = 2^{n^2(1-\frac{10}{\log n})-n-1}$. □

Our next question is: what about the canonical case, $\mathcal{G}_{\mathsf{cwvg}}(n)$? The set $\mathcal{G}_{\mathsf{cwvg}}(n)$ is a subset of $\mathcal{G}_{\mathsf{wvg}}(n)$, and for each noncanonical weighted voting game there exists a permutation of the players that makes it a canonical one. Since there are $n!$ possible permutations, it must be that $|\mathcal{G}_{\mathsf{cwvg}}(n)| \geq \frac{|\mathcal{G}_{\mathsf{wvg}}(n)|}{n!}$, and thus we obtain that

$$|\mathcal{G}_{\mathsf{cwvg}}(n)| \geq \frac{2^{n^2(1-\frac{10}{\log n})-n-1}}{n!} \tag{5}$$

for large enough n.

Appendix B. Generating Roofs from Ceilings

In this section, we answer the question whether it is possible to generate in polynomial time the list of ceiling coalitions of a linear game from its list of roof coalitions: This turns out to not be the case. We give a family of examples of canonical linear games in which the number of roof coalitions is exponential in n, while the number of ceiling coalitions is only polynomial in n. As a consequence, any algorithm that generates the list of roofs from the list of ceilings will run in exponential time in the worst case. By symmetry it also follows that generating the list of roof coalitions from the list of ceiling coalitions is not possible in polynomial time.

Let us first define the following specific type of coalition.

Definition 12 ((k,i)-encoding coalition)**.** *Let $N = \{1, \dots, n\}$ be a set of players such that $n = 4i$ for some $i \in \mathbb{N}$. For any k satisfying $0 \le k < 2^i - 1$, the* (k,i)-encoding coalition *$S_{k,i} \subseteq N$ is then defined as*

$$\{4(j-1)+2, 4(j-1)+3 : \textit{the } j\textit{th bit in the binary representation of } k \textit{ is } 0\} \quad \cup$$
$$\{4(j-1)+1, 4(j-1)+4 : \textit{the } j\textit{th bit in the binary representation of } k \textit{ is } 1\}$$

For example, $S_{2,2} = \{1,4,6,7\}$, and $S_{5,3} = \{1,4,6,7,9,12\}$. We can then define canonical linear games in which the roof coalitions are (k,i)-encoding coalitions.

Definition 13 (i-bit roof game)**.** *Let $N = \{1, \dots, n\}$ be a set of players such that $n = 4i$ for some $i \in \mathbb{N}$. The* i-bit roof game *with N, denoted $G_{i-\mathsf{bit}}$, is the canonical linear game such that the set of roof coalitions of G is $\{S_{0,i}, \dots, S_{2^i-1,i}\}$.*

For example, the 2-bit roof game, $G_{2-\mathsf{bit}}$, consists of the roofs $\{\{2,3,6,7\}, \{2,3,5,8\}, \{1,4,6,7\}, \{1,4,5,8\}\}$. $G_{i-\mathsf{bit}}$ is well-defined for all i because the binary representations of two arbitrary i-bit numbers k and k' differ in at least one bit. Therefore, $S_{i,k}$ is not a superset of a left-shift of $S_{i,k'}$ and hence the set of roofs that we have defined for $G_{i-\mathsf{bit}}$ is indeed a valid set of roofs (i.e., there are no two roofs such that one is a left-shift of another).

The game $G_{i-\mathsf{bit}}$ has $2^i = 2^{\frac{n}{4}}$ roofs, i.e., an exponential number in n. We will show that the number of ceilings in $G_{i-\mathsf{bit}}$ is only polynomially bounded. First let us use the following definitions for convenience.

Definition 14 (Accepting roof set)**.** *Let $G \in \mathcal{G}_{\mathsf{clin}}(n)$ be a canonical linear game with players $N = \{1, \dots, n\}$. Let $C \subseteq N$ be a coalition, let x be a natural number such that $1 \le x \le |C|$, and let $D(C,x)$ be the x-th most desirable player in C. The* accepting set of roofs of the x-th most desirable player in C*, denoted $A(C,x)$, is the set consisting of those roof coalitions R for which either the x-th most desirable player in R is greater than or equal to $D(c,x)$, or $|R| < x$.*

It is important to now observe that the following fact holds.

Proposition 2. *In a canonical linear game, a coalition C is winning if and only if $\bigcap_{a=1}^{|C|} A(C,a) \neq \varnothing$.*

Proof. This lemma is in fact an equivalent statement of the fact that C is winning in a canonical linear game if and only if it is a superset of a left-shift of a roof. If $R \in \bigcap_{a=1}^{|C|} A(C,a)$ then it means that replacing the a-th most desirable player in R by the a-th most desirable player in C for all a,$1 \le a \le R$ would result in a left-shift of R that is a subset of C, so C must be winning.

Conversely, suppose C is winning. Then there must be a roof R that is a right-shift of a subset of C. By removing from C the players with a higher number than $D(C,|R|)$, we obtain a subset C' of C with $|R|$ players. By replacing the a-th most desirable player of C by the a-th most desirable player of R for $1 \le a \le R$, we obtain a right-shift of C that is R. Because in this last step we replaced each player in C' by a higher-numbered player, we get that $R \in \bigcap_{a=1}^{|R|} A(C,a)$. R is also in $\bigcap_{a=|R|+1}^{|C|} A(C,a)$ by definition. □

Using the notion of an accepting roof set, we can prove the following technical lemma. The reader should recall the definition of a *direct* left-shift (Definition 4).

Lemma 4. *Let C be a ceiling of $G_{i-\mathsf{bit}}$ with two or more distinct coalitions that are direct left-shifts of C, and let p be an arbitrary player that we can apply the direct left-shift operation on, i.e., let p be a player such that $C_1 = C \cup \{p-1\} \setminus \{p\}$ is a direct left-shift of C. Also, let a be the number such that $p = D(C, a)$. Then $p = 2a$.*

Proof. Observe that for all b it holds that every roof R of $G_{i-\mathsf{bit}}$ has either $D(R, b) = 2b-1$ or $D(R, b) = 2b$. By construction of $G_{i-\mathsf{bit}}$, the number of roofs of $G_{i-\mathsf{bit}}$ that contain player $2b-1$ is $\frac{2^i}{2}$, and the number of roofs that contain player $2b$ is also $\frac{2^i}{2}$.

C has at least two distinct direct left-shifts, so there must be another player p', $p' \neq p$, such that $C_2 = C \cup \{p'-1\} \setminus \{p'\}$ is a direct left-shift of C.

First we will show that $p \leq 2a$. Assume therefore that $p > 2a$. Now we have that $|A(C, a)| = 0$, so then $|A(C_2, a)| = 0$ and hence $\bigcap_a A(C_2, a) = \varnothing$. We see that C_2 is losing, but C_2 is a direct left-shift of C, which is a ceiling, so C_2 is winning. This is a contradiction, so $p \leq 2a$.

Now we will show that $p \geq 2a$. Assume therefore that $p < 2a$. Now we have that $|A(C, a)| = 2^i$, so then $A(C_1, a) = 2^i$. Now it must be that $\bigcap_a A(C_1, a) = \bigcap_a A(C, a)$. But $\bigcap_a A(C, a) = \varnothing$ because C is losing, and therefore $\bigcap_a A(C_1, a) = \varnothing$ so C_1 is losing. C_1 is also winning, because it is a left-shift of ceiling C. This is a contradiction, so $p \geq 2a$.

Because $p \geq 2a$ and $p \leq 2a$, we conclude that $p = 2a$. □

Lemma 5. *In $G_{i-\mathsf{bit}}$, a ceiling does not have more than two direct left-shifts.*

Proof. For contradiction, let C be a ceiling with more than two direct left-shifts. Let k be the number of direct left-shifts of C, and let $P = \{p_1, \ldots, p_k\}$ be the set containing the players of C that we can apply the direct left-shift operation on (we say that we can apply the direct left-shift operation on a player q when $C \cup \{q-1\} \setminus \{q\}$ is a left-shift of C). Let $\mathcal{A} = \{a_1, \ldots, a_k\}$ then be the numbers such that p_j is the a_j-th most desirable player in C, for all i with $1 \leq j \leq k$. For any $j \in \{1, \ldots, i\}$ and any $b \in \{0, 1\}$, let $R(j, b)$ denote the following set of roofs of $G_{i-\mathsf{bit}}$:

$$R(j, b) = \{S_{k,i} : \text{The } j\text{-th bit of the binary representation of } k \text{ is } b.\ \}$$

Observe that by the previous lemma, there is a k-tuple of bits $(b_1, \ldots, b_k) \in \{0, 1\}^k$ such that for all j with $1 \leq j \leq k$:

$$A(C, a_j) = R(\lceil p_j/4 \rceil, k_j).$$

There are now two cases:

Case 1: *All of the players $\{p_1, \ldots, p_k\}$ are in different multiples of 4, i.e., $\lceil p_1/4 \rceil \neq \lceil p_2/4 \rceil \neq \cdots \neq \lceil p_k/4 \rceil$.* Then by the properties of the binary numbers, the intersection $\bigcap_{a \in \mathcal{A}} A(C, a) = \bigcap_{p \in P} R(\lceil p/4 \rceil, b)$ is not empty, therefore C must be winning, which is in contradiction with C being a ceiling. So this case is impossible.

Case 2: *There are two distinct players p and p', both in P, that are in the same multiple of 4, i.e.,* $\lceil p/4 \rceil = \lceil p'/4 \rceil$. Assume without loss of generality that $p < p'$. Then $A(C, a) \cap A(C, a') = \varnothing$. But then we would be able to apply a direct left-shift on player p'' without turning C into a winning coalition, i.e., $C \cup \{p'' - 1\} \setminus \{p''\}$ is winning. But C is a ceiling, so that is a contradiction.

From the previous lemma it follows that there can not be more than two players that are the same multiple of 4, so the above two cases are indeed exhaustive. Both cases are impossible, so we must reject the assumption that there exists a ceiling C with more than two left-shifts. □

It is easy to see that there exist no more than $O(n^5)$ coalitions with exactly two left-shifts, there are no more than $O(n^3)$ coalitions with one left-shift, and there are no more than $O(n)$ coalitions with no left-shifts. So we get the following corollary.

Corollary 2. *The game $G_{i-\mathsf{bit}}$ (on $n = 4i$ players) has $O(n^5)$ ceilings.*

We can now conclude that $\{G_{i-\mathsf{bit}} : i \in \mathbb{N}\}$ is an infinite family of examples in which there are exponentially many more roofs than ceilings. Hence, finally we obtain:

Corollary 3. *There is no polynomial time algorithm that generates the list of ceilings of a linear game from its list of roofs. There is no polynomial time algorithm that generates the list of roofs of a linear game from its list of ceilings.*

References

Algaba, E., Bilbao, J. M., García, J. R. F., & López, J. J. (2003). Computing power indices in weighted multiple majority games. *Mathematical Social Sciences*, *46*, 63–80.

Alon, N., & Edelman, P. H. (2010). The inverse Banzhaf problem. *Social Choice and Welfare*, *34*(3), 371–377.

Aziz, H. (2008). Complexity of comparison of influence of players in simple games. In *Proceedings of the 2nd International Workshop on Computational Social Choice (COMSOC)*, pp. 61–72.

Aziz, H., Paterson, M., & Leech, D. (2007). Efficient algorithm for designing weighted voting games. In *Proceedings IEEE International Multitopic Conference*, pp. 1–6.

Banzhaf III, J. (1965). Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review*, *19*(2), 317–343.

De, A., Diakonikolas, I., Feldman, V., & Servedio, R. A. (2012a). Nearly optimal solutions for the Chow parameters problem and low-weight approximation of halfspaces. In *Proceedings of the 44th Symposium on Theory of Computing*, pp. 729–746.

De, A., Diakonikolas, I., & Servedio, R. (2012b). The inverse Shapley value problem. In Czumaj, A., Mehlhorn, K., Pitts, A., & Wattenhofer, R. (Eds.), *Automata, Languages, and Programming*, Vol. 7391 of *Lecture Notes in Computer Science*, pp. 266–277. Springer.

Deng, X., & Papadimitriou, C. H. (1994). On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, *19*(2), 257–266.

Dubey, P., & Shapley, L. S. (1979). Mathematical properties of the Banzhaf power index. *Mathematics of Operations Research*, *4*(2), 99–131.

Fatima, S. S., Wooldridge, M., & Jennings, N. R. (2008). An anytime approximation method for the inverse Shapley value problem. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 935–942.

Fredman, M. L., & Khachiyan, L. (1996). On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, *21*(3), 618–628.

Freixas, J., & Kurz, S. (2011). On minimal integer representations of weighted games. *CoRR*, *abs/1303.0868*.

Freixas, J., & Kurz, S. (2013a). Enumeration of weighted games with minimum and an analysis of voting power for bipartite complete games with minimum. *Annals of Operations Research*, *Online first*.

Freixas, J., & Kurz, S. (2013b). The golden number and Fibonacci sequences in the design of voting structures. *European Journal of Operational Research*, *226*(2), 246–257.

Freixas, J., & Molinero, X. (2008). The greatest allowed relative error in weights and threshold of strict separating systems. *IEEE Transactions on Neural Networks*, *19*(5), 770–781.

Freixas, J., & Molinero, X. (2010). Weighted games without a unique minimal representation in integers. *Optimization Methods and Software*, *25*(2), 203–215.

Freixas, J., Molinero, X., & Roura, S. (2012). Complete voting systems with two classes of voters: weightedness and counting. *Annals of Operations Research*, *193*(1), 273–289.

Hu, S. (1965). *Threshold logic*. University of California Press.

Isbell, J. R. (1958). A class of simple games. *Duke Mathematical Journal*, *25*(3), 423–439.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM Symposium on Theory of computing (STOC)*, pp. 302–311.

De Keijzer, B. (2009a). On the design and synthesis of voting games: exact solutions for the inverse problem. Master's thesis, Delft University of Technology.

De Keijzer, B. (2009b). A survey on the computation of power indices. Tech. rep., Delft University of Technology.

De Keijzer, B., Klos, T., & Zhang, Y. (2010). Enumeration and exact design of weighted voting games. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 391–398.

De Keijzer, B., Klos, T., & Zhang, Y. (2012). Solving weighted voting game design problems optimally: Representations, synthesis, and enumeration. *CoRR*, *abs/1204.5213*.

Kleijn, A. (2012). Weighted voting game design heuristics. Master's thesis, Erasmus School of Economics, Erasmus University Rotterdam.

Kleitman, D., & Markowski, M. (1975). On Dedekind's problem: The number of isotone boolean functions II. In *Transactions of the American Mathematical Society*, Vol. 213, pp. 373–390.

Klinz, B., & Woeginger, G. J. (2005). Faster algorithms for computing power indices in weighted voting games. *Mathematical Social Sciences*, *49*, 111–116.

Korshunov, A. D. (2003). Monotone boolean functions. *Russian Mathematical Surveys*, *58*(5), 198–162.

Krohn, I., & Sudhölter, P. (1995). Directed and weighted majority games. *Mathematical Methods of Operations Research*, *42*(2), 189–216.

Kurz, S. (2012a). On minimum sum representations for weighted voting games. *Annals of Operations Research*, *196*(1), 361–369.

Kurz, S. (2012b). On the inverse power index problem. *Optimization*, *61*(8), 989–1011.

Laruelle, A., & Widgrén, M. (1998). Is the allocation of voting power among EU states fair?. *Public Choice*, *94*, 317–339.

Leech, D. (2002a). Computation of power indices. Tech. rep. 664, Warwick Economic Research Papers.

Leech, D. (2002b). Designing the voting system for the EU Council of Ministers. *Public Choice*, *113*(3–4), 437–464.

Leech, D. (2002c). Voting power in the governance of the International Monetary Fund. *Annals of Operations Research*, *109*, 373–395.

Leech, D. (2003). Power indices as an aid to institutional design: the generalised apportionment problem. In Holler, M., Kliemt, H., Schmidtchen, D., & Streit, M. (Eds.), *European Governance*, No. 22 in Jahrbuch für Neue Politische Ökonomie, pp. 107–121. Mohr Siebeck.

Makhorin, A. (2004). GNU linear programming toolkit..

Matsui, Y., & Matsui, T. (2001). NP-completeness for calculating power indices of weighted majority games. *Theoretical Computer Science*, *263*(1–2), 305–310.

McKay, B. D. (1998). Isomorph-free exhaustive generation. *Journal of Algorithms*, *26*, 306–324.

Muroga, S. (1971). *Threshold logic and its applications*. Wiley-Interscience.

Muroga, S., Toda, I., & Kondo, M. (1962). Majority functions of up to six variables. *Mathematics of Computation*, *60*(80), 459–472.

Muroga, S., Tsuboi, T., & Baugh, C. R. (1970). Enumeration of threshold functions of eight variables. *IEEE Transactions on Computers*, *C-19*(9), 818–825.

De Nijs, F., Wilmer, D., & Klos, T. (2012). Evaluation and improvement of Laruelle-Widgrén inverse Banzhaf approximation. In *Proceedings Benelux AI Conference (BNAIC)*, pp. 194–201.

O'Donnell, R., & Servedio, R. A. (2011). The Chow parameters problem. *SIAM Journal on Computing*, *40*(1), 165–199.

Parberry, I. (1994). *Circuit complexity and neural networks*. Mit Press.

Peled, U. M., & Simeone, B. (1985). Polynomial-time algorithms for regular set-covering and threshold synthesis. *Discrete Applied Mathematics*, *12*, 57–69.

Peleg, B., & Sudhölter, P. (2003). *Introduction to the Theory of Cooperative Games*. Springer.

Polyméris, A., & Riquelme, F. (2013). On the complexity of the decisive problem in simple, regular and weighted games. *CoRR*, *abs/1303.7122*.

Prasad, K., & Kelly, J. S. (1990). NP-completeness of some problems concerning voting games. *International Journal of Game Theory*, *19*(1), 1–9.

Shapley, L. S., & Shubik, M. (1954). A method for evaluating the distribution of power in a committee system. *American Political Science Review*, *48*(3), 787–792.

Siu, K., Roychowdhury, V., & Kailath, T. (1995). *Discrete Neural Computation: A Theoretical Foundation*. Prentice Hall.

Sperner, E. (1928). Ein Satz über Untermengen einer endlichen Menge. *Mathematische Zeitschrift*, *27*(1), 544–548.

Sutter, M. (2000). Fair allocation and re-weighting of votes and voting power in the EU before and after the next enlargement. *Journal of Theoretical Politics*, *12*, 433–449.

Taylor, A. D., & Zwicker, W. S. (1999). *Simple Games: Desirability Relations, Trading, Pseudoweightings*. Princeton University Press.

Winder, R. O. (1965). Enumeration of seven-argument threshold functions. *IEEE Transactions on Electronic Computers*, *EC-14*(3), 315–325.

Zuev, Y. A. (1989). Asymptotics of the logarithm of the number of threshold functions of the algebra of logic. *Soviet Math. Dokl.*, *39*, 512–513.

Žunić, J. (2004). On encoding and enumerating threshold functions. *IEEE Transactions on Neural Networks*, *15*(2), 261–267.