A Survey on the Computation of Power Indices

and Related Topics

Bart de Keijzer

A Survey on the Computation of Power Indices

RESEARCH ASSIGNMENT

by

Bart de Keijzer
born in Amstelveen, the Netherlands

Algorithmics Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

A Survey on the Computation of Power Indices

Author: Bart de Keijzer
Student id: 1221779
Email: B.deKeijzer@student.tudelft.nl

Abstract

A 'power index' is a concept in cooperative game theory. Power indices are used as a measure for the influence that a player has in a certain class of cooperative games called simple games. As the title says, this is a survey on how to compute these power indices. Generally speaking, computing these indices is a tedious task, even for relatively small cooperative games. It is certainly intractable to do this by hand, and therefore as you might have guessed, we need computers to do this for us. In this survey we give first a short introduction to cooperative game theory with a focus on simple games. We discuss the power indices that have been proposed in the literature, and we present algorithms for computing some of these indices. The majority of the algorithms for computing power indices that have been proposed are suited for weighted voting games: an important subclass of simple games. Therefore, the majority of the algorithms that we present here are only suited for weighted voting games. Among the algorithms that we discuss are some of the best that are currently known. In the last chapter, we give some interesting directions for future research. Throughout our survey we try to pay special attention to the relation between computing power indices, computational complexity (one chapter is dedicated to this) and approximability.

Thesis Committee:

Chair: Prof. Dr. C. Witteveen, Faculty EEMCS, TU Delft
University supervisor: Dr. Tomas Klos, Faculty EEMCS, TU Delft
Dr. Yingqian Zhang, Faculty EEMCS, TU Delft

Contents

Chapter 1

Introduction

In this document, we give an overview of the research that has been done in computer science with regard to power indices.

1.1 Topic of this Survey

A power index is a notion in cooperative game theory that arose from the need to measure someone's *a priori* power in certain coalitional games. It turns out to be a challenge to compute power indices, so a lot of research has been done in computer science on how to do this efficiently. Also, it turns out that power indices have an interesting application to computer science domains, namely network reliability, which is a second reason for why power indices are an interesting topic for computer science research. In this survey, we will look at power indices from a computer science perspective. This means that we will talk about algorithms that compute power indices, and we will discuss issues regarding the computational complexity of power index computation.

The reader might be a bit unfamiliar with some of the terminology we just used. This is no problem: all will be explained throughout this text. However, what we do assume is that the reader has a computer science background (e.g. has knowledge of algorithms and complexity theory).

Let us first quickly return to the domain of network reliability. This is an important example of a situation for which power indices can be applied. The use of power indices for measuring network reliability is studied in [9] and [10]. We refer the interested reader to those papers, because —although we ourselves think that this application is interesting— we will not discuss it in this survey. For this survey, we will mostly focus on power index computation for decision making settings. In practical instances of these settings, people mostly work with so-called *weighted voting games*, to be defined formally in the next chapter. In such games each participant is assigned a weight, representing the importance of his vote.

1.2 Applications

Examples of real-life situations of such games include most political decision making bodies. For example, look at the presidential elections of the United States of America, where every state can be seen as a participant, and their weight equals the amount of electors of that state.

Another example is in the European Union, where two decision making rules are used, consisting of three weighted voting game each. The member-nations of the european union are in this case the participants. In order to win, a coalition of nations must win all three weighted voting games simultaneously. We will work out this example in more detail in section 3.2.3.

As a final example, in companies with multiple owners, each holding a certain share of stocks in the company, weighted voting games are often adopted as a decision-making protocol.

1.3 Outline

The outline of this survey is as follows. Chapter 2 serves as an introduction to cooperative game theory. In chapter 3 we explain what power indices are, and we give the definitions of the different power indices that have been invented. In chapter 4 we will introduce some relevant complexity classes (especially #P), and we discuss some computational problems regarding power indices. A lot of the more interesting problems turn out to be complete for the class #P. In chapter 5, we will discuss algorithms for exactly computing power indices, whereas in chapter 6 we will look at approximation methods. Finally, in chapter 7, we will discuss some future research and open problems in the field of power index computation.

Most research that has been done for computing power indices is on the so-called *weighted voting game*. Therefore, the majority of the results that are given in chapters 4, 5 and 6 will be about weighted voting games.

Chapter 2

Cooperative Game Theory

In this chapter we give an introduction to *cooperative game theory*, or *coalitional game theory*. We discuss the most important topics in this field, and especially the topics that are relevant for the subfield of power indices.

2.1 Comparison to Classical Game Theory

Cooperative game theory is very different from it's non-cooperative (classical) counterpart. First we will talk informally about the non-cooperative part of game theory.

Classical *game theory* is centered around the mathematical analysis of behaviour of rational entities in strategic situations. With "the behaviour of rational entities" we mean most of the time just humans of whom the assumption is made that they're completely rational. A strategic situation is called a *game* in game theory, hence the name "Game Theory". The rational entities participating in a game are called the *players*. That should sound pretty logical. Sometimes, but not often, the rationality assumption is dropped, and a different model of behaviour is assumed for the players in a game.

One straightforward mathematical definition of a game for n players is defined to be a set of n *strategy spaces*: one for each player. A strategy space is a set of strategies that a player can adopt. As part of the definition of the game, a payoff scheme is provided that specifies for each combination of strategies what the payoff to each player is. A prominent question that is studied in game theory is which strategy a player should choose in order to maximize his utility. The utility of a player is a real-valued function that depends on the payoff.

In cooperative game theory, we don't study the behaviour of individual players. Instead we are interested in what happens if players work together, and we study how they should work together. For this, we transform games into a more abstract form, called the *coalitional form*. In the coalitional form of a game, there is only a set of players and a function that specifies the gain that can be obtained for each subset of players, if they work together. Hence, in cooperative game theory, we completely disregard strategy spaces, payoff schemes and utility functions.

It is possible to transform games into a coalitional form, although in cooperative game theory often it is the case that we just start analyzing a coalitional form game without considering the non-coalitional form. What we want to say with this is that coalitional form games are regarded as an interesting object of study in itself, without it just being a "tool" for analysis of the underlying non-cooperative game. This might already sound sensible to the reader, but it should certainly become clear later on, where we start studying simple games.

2.2 Coalitional Games

Now we will introduce some important concepts in cooperative game theory. All of this can be looked up in any introductory text on cooperative game theory, for example [47].

2.2.1 General Games

Definition 1 (Coalitional game)**.** A *coalitional game* is a pair (A, v), where $A = \{a_1, \ldots, a_n\}$ is a set of players. And $v : 2^A \to \mathbb{R}$ is a function mapping coalitions of players to a real number, describing how much collective payoff the coalition can gain. Also, $v(\varnothing) = 0$.

Definition 2 (Grand coalition & characteristic function)**.** For a coalitional game (A, v), A is called the *grand coalition*. v is called the *characteristic function* or *gain function*.

Definition 3 (Subgame)**.** Let (A, v) be a coalitional game. (S, v) is the *subgame* of (A, v) *induced on* S if and only if $S \subseteq A$.

Often, a coalition game is denoted by just v if it's clear what the number of players is. For example if we denote by $\mathcal{G}(N)$ the class of coalitional games on N players then we simply say $v \in \mathcal{G}(N)$.

There exist two important subclasses of games: *monotone games* and *superadditive games*. These are games that satisfy respectively the following two constraints:

Definition 4 (Monotonicity)**.** A game (A, v) is *monotone* if and only if $\forall S, T \subseteq A : S \subset T \to v(S) \leq v(T)$.

Definition 5 (Superadditivity)**.** A game (A, v) is *superadditive* if and only if $\forall S, T \subseteq A : S \cap T = \varnothing \to v(S \cup T) \geq v(S) + v(T)$.

Obviously if a game is superadditive it is also monotone, so the superadditive games form a subclass of the monotone games.

2.2.2 Simple Games

Another important subclass of coalitional games, is the class of *simple games*. Informally it consists of the games where a coalition is either winning or losing, and there's nothing in between. For the topic of power indices, we are only interested in this particular class of games. More precisely, power indices are only defined on simple games.

Definition 6 (Simple coalitional game)**.** A *simple coalitional game* or simply *simple game* is a coalitional game (A,v), where the target of v is restricted to $\{0,1\}$. Moreover, we assume $v(\varnothing) = 0$ and $v(A) = 1$. If for a subset S of A, $v(S) = 1$, then S is called a *winning coalition*. If $v(S) = 0$ then S is called a *losing coalition*.

Just as with general coalitional games, we can define analogously the class of *monotone simple games* and *superadditive simple games*. Note that in a lot of literature, the monotonicity constraint is included in the definition of simple games and general games, so simple games as well as general games are often assumed to be monotone in the first place, and monotone games are not regarded as a special case.

There are some other relevant subclasses of simple games. Some of these can be syntactically defined as we will see. We will first start by defining some useful properties of simple games.

Definition 7 (Blocking coalition, proper game strong game & decisive game)**.** In a simple game (A,v), a coalition $S \subset A$ is a *blocking coalition* iff $v(A \backslash S) = 0$. In a *proper simple game*, all winning coalitions are blocking coalitions. In a *strong simple game*, all blocking coalitions are winning coalitions. A *decisive simple game* is a simple game that is both strong and proper.

Decisive simple games are important for settings where it must be guaranteed that a decision between two alternatives is made after voting.

Some of the definitions in the remainder of this section were taken or adapted from [4] and [55]. Next, we turn to the syntactic definitions of certain classes of simple games. There are four important ways to represent simple games.

Definition 8 (Representations of simple games)**.** Consider the following four syntactic classes of games:

Winning coalition form (N,W) is a *winning coalition form* of a simple game (N,v) iff W is the set of all coalitions such that $\forall S \subseteq N : v(S) = 1 \leftrightarrow S \in W$.

Minimal winning coalition form $(N,W_{\min})$ is a *minimal winning coalition form* of a simple game (N,v) iff $W_{\min}$ is the set of *minimal winning coalitions* of the game (N,v), this means that $\forall S \subseteq N : v(S) = 1 \leftrightarrow \exists T \in W_{\min} : S \supseteq T$. This condition implies that a simple game can be represented in minimal winning coalition form if and only if it is mononic.

Weighted voting game $(W,q), W = (w_1,\ldots w_{|N|}), w_1,\ldots w_{|N|}, q \in \mathbb{N}$ is a *weighted voting game* or *weighted majority game* form of a simple game $(N = \{1,\ldots |N|\}, v)$ if $\forall S \subseteq N : v(S) = 1 \leftrightarrow \sum_{i \in N} w_i \geq q$. q is called the *quota* and w_i is called the *weight of player* i. Obviously, any weighted voting game is monotone. However, not any monotone game can be represented as a weighted voting game. Later, we will get to the question of which games can be represented as a weighted voting game. Note that in the literature, not everyone defines that the weights and quota of weighted voting games must be integers.

Mutiple weighted voting game An *m-multiple weighted voting game* or *weighted m-majority game* for a simple game (N,v) is a set

$$\{(W_1 = (w_1^1, \dots w_{|N|}^1), q_1), \dots, (W_m = (w_1^m, \dots w_{|N|}^m), q_m)\}$$

of m weighted voting games such that it holds that $\forall S \subseteq N : v(S) = 1 \leftrightarrow \forall 1 \leq j \leq n : \sum_{i \in N} w_i^j \geq q_j$. In words, a coalition must win every weighted voting game that constitutes the multiple weighted voting game.

Remark 9. We would like to point out that there is a very strong connection between the theory of simple games and the theory of boolean functions, especially *monotone boolean functions* and *boolean threshold functions* A monotone boolean function is the same as a monotone simple game, except that the players who participate in the game are replaced by variables that are set to 0 or 1. A boolean threshold function is a monotone boolean function that can be represented by a weight function, in the same was as that a weighted voting game is a monotone simple game that can be represented by a weight for each player.

Some notes on weighted voting games: a weighted voting game is a very important representation, not only intuitively, but also because it's a compact representation. Also they're important because they are often used in practice, in a lot of decision making protocols. For example: political elections, political decision making bodies and decision making bodies in stock-holder companies. Completely unrelated to this, note that a weighted voting game (W,q) is proper if and only if $q \geq \frac{\sum_{w \in W} w}{2}$.

There is an important theorem regarding m-multiple weighted voting games.

Theorem 10. *Every monotonic simple game is representable as an m-multiple weighted voting game.*

The proof is given in the book of Taylor and Zwicker [55]. This proof is constructive, but the problem is that in the construction m gets very large, while it often is possible to represent the m-multiple weighted voting game with a much smaller value for m.

Definition 11 (Dimension of a monotone simple game)**.** The *dimension* of a monotone simple game G is the smallest m such that there exists an m-multiple weighted voting game representation for G.

Now we will answer the question of which monotone games are representable as a weighted voting game. For this we will have to introduce the following two notions.

Definition 12 (Trade robustness & swap robustness)**.** A simple coalitional game (N,v) is *swap robust* if for any two winning coalitions in that game, say S and T, if we swap one player in S with one player in T, then the resulting two coalitions are not both losing.

A simple game (N,v) is *trade robust* if for any set $\mathcal{S}$ of winning coalitions in that game, any redistribution of players among the coalitions in $\mathcal{S}$ does never result in all coalitions in $\mathcal{S}$ becoming losing.

Clearly, trade robustness implies swap robustness. In [55], the following theorem is proven.

Theorem 13. *A simple coalitional game can be represented as a weighted voting game if and only if it is trade robust.*

Another important concept, related to swap and trade robustness is the *desirability relation* (see [55] and [22]). We define the following desirability relation over the players in a simple game.

Definition 14 (Desirability relation)**.** In a simple game $(N = \{1, \ldots, |N|\}, v)$, $\succeq_D$ is the *desirability relation* defined by:

- For any $i, j \in N$: if $\forall S \subseteq N \backslash \{i, j\} : v(S \cup \{i\}) \geq v(S \cup \{j\})$, then $i \succeq_D j$. In this case we say that i is *more desirable* than j.
- For any $i, j \in N$: if $\forall S \subseteq N \backslash \{i, j\} : v(S \cup \{i\}) = v(S \cup \{j\})$, then $i \sim_D j$. In this case we say that i and j are *equally desirable*.
- For any $i, j \in N$: if $i \succeq_D j$ and not $i \sim_D j$, then $i \succ j$. In this case we say that i is *strictly more desirable* than j.

Moreover, if neither $i \succeq_D j$ nor $j \succeq_D i$ holds for some $i, j \in N$, then we say that i and j are *incomparable*.

Now we can define *linear games*.

Definition 15 (Linear game)**.** A simple game (N, v) is *linear* if and only if no pair of players in N is incomparable. The latter is equivalent to saying that the desirability relation over N is complete. This is also equivalent to saying that the desirability relation is acyclic (or: transitive).

There exist other definitions of desirability relations for which different properties hols [22]. A remarkable result is the following, proved by Taylor and Zwicker in [55]:

Theorem 16. *A game is linear if and only if it is swap robust.*

As a last topic before we finish this exposition of simple games, we want to introduce several different player types that can be distinguished in a simple game.

Definition 17 (Player types in simple games)**.** In a simple coalitional game $(N = \{1, \ldots, |N|\}, v)$ the following special types of players can be distinguished:

- A player $i \in N$ is a *dummy* iff there exists no $S \subseteq N \backslash \{i\}$ such that $v(S \cup \{i\}) = 1$ and $v(S) = 0$.
- A player $i \in N$ is a *passer* iff for all $S \subseteq N : i \in S \rightarrow v(S) = 1$.
- A player $i \in N$ is a *vetoer* iff for all $S \subseteq N : i \notin S \rightarrow v(S) = 0$.
- A player is a *dictator* if he is both a passer and a vetoer.

2.3 Solution Concepts

Now we turn to the general class of coalitional games that are not necessarily simple.

Solution concepts are a central topic in coalitional game theory.

Definition 18 (Solution concepts in cooperative games)**.** For a class of cooperative games $\mathcal{G}(N)$ on players $N = 1,\ldots,|N|$, a *solution concept* $c : \mathcal{G}(N) \to \mathbb{R}^{|n|}$ is a way to divide the gain $v(N)$ from the grand coalition N among the members of that coalition.

If the assumption is that eventually the grand coalition will form in a game, then we need to split up the gains from the grand coalition among the set of all players. This is a standard assumption in cooperative game theory: games are assumed to be monotone, and thus the highest gain is obtained by the grand coalition i.e. the case that everyone cooperates. Solution concepts are usually defined from this point of view.

This assumption is not that restrictive though: one can also see a solution concept as a way to divide the gains of *any* coalition among its members. If we want a method to divide the gain of a smaller coalition among its members, then we can just apply the solution concept on the subgame induced on the players in the smaller coalition.

Below, we will describe the Shapley Value solution concept. We give special attention to the Shapley value, since it is relevant for our discussion about power indices. However, there are also other popular solution concepts, such as the core [28], the nucleolus [51], and the kernel [17].

2.3.1 The Shapley Value

One of the most well-known solution concepts is the Shapley value. This solution concept was introduced by Lloyd S. Shapley in [52].

Definition 19. Given a coalitional game $(A = (a_1,\ldots,a_n), v)$, the Shapley value φ for this game is the payoff vector $(\varphi_1,\ldots,\varphi_n)$. For $1 \leq i \leq n, \varphi_i$ is defined as follows. Let Π be the set of all permutations of A. Let $\text{Prec}_i : \Pi \to 2^A$ be the function that, given a $\pi \in \Pi$, returns the set of players occuring before a_i in pi. Then

$$\varphi_i = \frac{\varphi_i'}{n!},$$

and

$$\varphi_i' = \sum_{\pi \in \Pi} v(\text{Prec}_i(\pi) \cup \{i\}) - v(\text{Prec}_i(\pi)).$$

Remark 20. Whenever we're discussing solution concepts of two or more games simultaneously, we will distinguish them for these games by parametrizing it with the specific game. e.g for two simple games G' and G, we use the notation $\varphi_i(G)$ and $\varphi_i(G')$ to denote the shapley values for player i in game G and game G' respectively. Also we sometimes denote a game $G = (N, v)$ simply game by its characteristic function v, so then we write $\varphi_i(v)$. Actually, $\varphi_i(v)$ is the "correct" way to write it down, so in our definition we actually abused notation and will continue to do so.

In cooperative game-theory research, a common practice is to try to axiomatize a solution concept.

Definition 21 (Axiomatic characterization)**.** An *axiomatization* or *axiomatic characterization* of a solution concept c for a class of games $\mathcal{G}$ is a set of axioms defined using games in $\mathcal{G}$ such that c is the unique solution concept that satisfies all the axioms.

Axiomatic characterizations are useful for assessing the reasonability of solution concepts. For the Shapley value, a lot of axiomatic characterizations have been devised. We now give a clean one in definition 22 and theorem 23, devised by Shapley himself, given for example in [19]:

Definition 22 (Game permutation & game addition)**.** Let v be a coalitional game on the players N; let $\pi : N \to N$ be a permutation of N. Then the game (πv) is defined as

$$\forall S \subseteq N : (\pi v)(S) = v(\{\pi(i) \mid i \in S\}).$$

Let v_1 and v_2 be two games on the players N; then the game $(v_1 + v_2)$ is defined as

$$\forall S \subseteq N : (v_1 + v_2)(S) = v_1(S) + v_2(S).$$

Theorem 23. *Let $\mathcal{G}(N)$ be the class of general coalitional games on any set of players $N = (1, \ldots, |N|)$. For any $v, v_1, v_2 \in \mathcal{G}(N)$, there is a solution concept $c = (c_1, \ldots, c_{|N|})$ that is the unique solution concept satisfying*

A1 (carrier): *If any $S \subseteq N$ satisfies $\forall T \subseteq N : v(T) = v(T \cap S)$, then $\sum_{i \in S} c_i(v) = v(S)$;*

A2 (permutation): *For any permutation π: $c_i(v) = c_{\pi(i)}(\pi v)$;*

A3 (addition): $c_i(v_1 + v_2) = c_i(v_1) + c_i(v_2)$.

Moreover, c is the Shapley value φ.

Chapter 3

Power Indices

We will discuss power indices in this chapter. Mostly, the chapter will be an enumeration of some indices[1] that have been invented over time. But we start with a discussion of why we need power indices.

3.1 Motivation Behind Power Indices

Remark 24. First we would like to note the following, mostly because there's no other point more suitable to mention this: the website [3] by Antti Pajala, from the University of Turku, Finland turned out to be very helpful. This website is dedicated to voting power and power indices, and it proved to be a good resource for general information on power indices. Most importantly, it turned out to be an excellent starting point for our literature research in power indices, containing over 350 references to scientific articles on voting power.

Power Indices originally were introduced because it was observed that in weighted voting games, the weight of a player is not directly proportional to the influence he has in the weighted voting game. This is actually quite easy to see through the following trivial example weighted voting game:

$$\left(W, q = \sum_{w \in W} w\right)$$

Here, each agent is in only one winning coalition: the grand coalition. So no matter what the weights of the agents are, they all have the same power.

Now that we know that the weight of a player is not a good measure for the player's power, the question will be: what *is* a good measure for the player's power in a weighted majority game?

Various answers have been given to this question, by various researchers in the area of coalitional game theory. These answers are in the form of *power indices*, which are simply mathematical formulations for values that try to describe the 'true' influence that a player has in a weighted voting game.

[1] However, not *all* indices. We picked only the most important ones.

Almost all power indices make no assumptions on the true preference of a player in a voting game. So we say power indices measure a player's *a priori* power in a weighted voting game. That is, we attempt to objectively measure the influence that a player has in the outcome of a weighted voting game, without having any statistical information on which votes are likely to be casted by players, or groups of players. To do this, we can not avoid making certain assumptions, but we let these assumptions be as neutral as possible. For example, in the Banzhaf index that we describe below, it is assumed that each coalition will form with equal probability.

While the need for power indices originally arose from weighted voting games, all of the power indices that have been devised up till now are also suitable for any other kind of simple coalitional game. So, for any simple coalitional game, we can use a power index as a measure of a player's a priori power in it.

In computer science, the focus of power index research naturally lies most of the times in finding algoritms that efficiently compute a power index. However, of course power indices have not only been investigated in computer science: even more research has been done in the field of game theory (pretty obvious; power indices are a game-theoretic notion). This purely game-theoretical research has mostly focused on mathematical analysis of the indices: finding mathematical properties and finding *axiomatic characterizations* of the power indices, just as has been done for the solution concepts of the previous chapter. Analogous to axiomatic characterizations of solution concepts, an axiomatic characterization of a power index p is a set of axioms such that the only formula satisfying all axioms can only be p. Axiomatizations are helpful for discussions on topics like the reasonability of a certain power index in a certain situation. And although axiomatizations do give us some mathematical properties of the power indices, those properties are obvious and follow most of the times immediately from the definition. It turns out that they do not really help us for answering most of the computation-related questions that we are interested in, so here we will not state explicitly any axiomatic characterizations for the power indices.

One quick last note before we move on to the actual power indices. Originally, a power index p was defined to be a vector $(p_1, p_2, \ldots, p_n)$ where $p_i, 1 \leq i \leq n$, is supposed to be a measure of the power player i (at least, this is the way most of the classic game theory papers define their power index). Later on, in more and more papers this definition was altered a bit, and researchers began to speak of "the power index p_i", i.e. all of the p_i's themselves were defined to be power indices of individual players. In this document, we use the definitions in both ways, depending on which definition is convenient. We believe no confusion wil arise from this.

3.2 Well-known Power Indices

In this section, we will discuss the two most used power indices: the Banzhaf index and the Shapley-Shubik index. In the computer science literature about power indices, most of the time only these two indices are discussed. After we discuss them, we will give an example.

3.2.1 The Shapley-Shubik index

The very first power index that has been proposed was the Shapley-Shubik index. This is nothing more than the Shapley value of [52] that we introduced in the previous chapter, but now restricted to simple coalitional games. So:

Definition 25 (Shapley-Shubik index & raw Shapley-Shubik index)**.** For a simple coalitional game $(A = \{a_1, \ldots, a_n\}, v)$, let Π be the set of all permutations of all players. For all $1 \leq i \leq n$, let $\text{Prec}_i : \Pi \to 2^W$ be the function such that $\text{Prec}_i(\pi \in \Pi)$ returns the set of players that occur before player $i \in \pi$. Then, the *Shapley-Shubik index* is $\varphi = (\varphi_1, \ldots, \varphi_n)$, where for $1 \leq i \leq n$:

$$\varphi_i = \frac{\varphi'_i}{n!},$$

and

$$\varphi'_i = \sum_{\pi \in \Pi} v(\text{Prec}_i(\pi) \cup \{i\}) - v(\text{Prec}_i(\pi)).$$

φ'_i is called the *raw Shapley-Shubik index*.

We can interpret the raw Shapley-Shubik index for i as the number of different orders of arrivals in which the players can join the coalition, such that the arrival of player i changes a losing coalition into a winning coalition.

Remark 26. For our notational conventions for power indices, our comments in remark 20 also hold.

Definition 27 (Pivot permutations & pivot players for permutations)**.** Let $\pi \in \Pi$ be a permutation such that $v(\text{Prec}_i(\pi) \cup \{i\}) - v(\text{Prec}_i(\pi)) = 1$. We call π a *pivot permutation* for player i. Also, we say that in that case, i is a *pivot player* for π.

The Shapley-Shubik index of player i is the raw Shapley-Shubik index of i divided by $n!$. $n!$ is the total number of possible permutations of all n players, and because in each permutation, only one player is a pivot player, we have

$$\sum_{i=1}^{n} \varphi_i = 1$$

which is a nice property. As a corollary, the Shapley-Shubik index of a player is always a number between 0 and 1. It is easy to see that the Shapley-Shubik index of a player i can be interpreted as the probability that i will be the player that changes a losing coalition into a winning coalition, if all players would join the coalition in a random order.

There is an alternative well-known definition of the raw Shapley-Shubik index (for a player i) that is sometimes convenient to use:

$$\varphi_i = \sum_{S \subseteq A \setminus \{i\}} (|S|!(|A| - |S| - 1)!(v(S \cup \{i\}) - v(S))) \tag{3.1}$$

It is not hard to see why this definition is correct: for any $S \subseteq A$ for which it holds that $(v(S \cup \{i\}) - v(S)) = 1$, there are $|S|!$ ways to permute the players in S, and there are $(|A| - |S| - 1)!$ ways to permute the $|A| - |S| - 1$ players outside S.

The reader may wonder why this power index is called the Shapley-Shubik index if the Shapley-Shubik index is nothing more than the Shapley value, restricted to simple coalitional games. The reason for this is that while Shapley invented the Shapley value, it was the paper [53] written by Shapley and Shubik, where it was pointed out that the Shapley value is an excellent way to measure someone's voting power in a weighted voting game. As we explained, originally the Shapley value was meant as a solution concept for general coalitional games, i.e. to allocate fairly the gain of the grand coalition among the set of players. In this new setting, the Shapley value is used as a way to measure power, so that's for a different purpose as the original purpose of the Shapley value. Hence, the Shapley-Shubik index was born.

The axiomatic characterization for the Shapley value (that we gave in the previous chapter) holds if we restrict it to simple games, but not if we restrict it to simple superadditive games or simple monotone games. In general it is not true that an axiomatic characterization for a certain value on certain set G of games holds on a subset $G' \subset G$: while it is always true that in this case the value satisfies the axioms, it needs no longer be so that the value is the *unique* value that satisfies the axioms. Shubik states in [19] an axiomatization for the Shapley-Shubik value that holds in simple superadditive and simple monotone games.

3.2.2 The Banzhaf index

In 1965, two decades after the Shapley-Shubik index, [31] the *normalized Banzhaf power index* was proposed, named after its inventor John F. Banzhaf III.

Definition 28 (normalized Banzhaf index & raw Banzhaf index)**.** For a simple coalitional game $(A = (a_1, \ldots, a_n), v)$, the *normalized Banzhaf index* is defined as $\beta = (\beta_1, \ldots, \beta_n)$, where for $1 \leq i \leq n$:

$$\beta_i = \frac{\beta'_i}{\sum_{j=1}^{n} \beta'_j},$$

and

$$\beta'_i = |\{S \subseteq A \backslash \{i\} \mid v(S) = 0 \wedge v(S \cup \{i\}) = 1\}|. \tag{3.2}$$

Here, β'_i is called the *raw Banzhaf index* for player i.

So the raw Banzhaf index is simply the number of coalitions where i is critical for the coalition to win the game.

Definition 29 (Swing coalition & critical player)**.** Let $S \subseteq W$ be a coalition such that $v(S) = 0 \wedge v(S \cup \{i\}) = 1$. We will call S a *swing coalition* for i. In a swing coalition for i, i is called a *critical player* or *swing player* (because he "swings" the outcome for the coalition from losing to winning, if he joins).

Note that the definition we use here differs from the definition in some other literature. Originally, for a simple game (N,v) a swing is defined as a pair $(S,S'),S,S' \in N,S' \subset S,|S'| = |S|-1$. However, we think that for our discussion, definition 29 is more convenient.

From the definition we can see that the Banzhaf index for player i is simply the raw Banzhaf index divided by the sum of all raw Banzhaf indices. This way, β_i is always a number between 0 and 1, and we have the elegant mathematical property that $\sum_{i=1}^{n} \beta_i = 1$.

Coleman reinvented the index in 1971 [14], only after which the index became popular. For this reason, the normalized Banzhaf index form is also known as the *Coleman index* or the *Banzhaf-Coleman index*. Coleman has some other power indices though, see the next section for that. Moreover, the normalized Banzhaf index is also known under the name *standardized Banzhaf index*.

Later, in 1979, the index was revised by Dubey and Shapley. It was noted that because of the denominator $(\sum_{j=1}^{n} \beta'_j)$, some useful information was lost. Therefore, Dubey and Shapley proposed in [20] a modification of the banzhaf index where the denominator is replaced by 2^{n-1}.

Definition 30 (Absolute Banzhaf index)**.** So this modified Banzhaf index β''_i looks like

$$\beta''_i = \frac{\beta'_i}{2^{n-1}}.$$

The authors called this index the *absolute Banzhaf index*.

β''_i is precisely the fraction of coalitions containing i, for which i is a critical player. So if each coalition were to form with equal probability, then β''_i is the probability that i is a critical player for the coalition, given that player i is in the coalition.

In [20], an axiomatic characterization of the raw Banzhaf index is given for the general class of simple games.

Actually, as turned out, the absolute Banzhaf index was invented much earlier, even before the normalized Banzhaf index was proposed. In 1946, the index was described by Penrose in [48]. So in the literature, the absolute Banzhaf index is sometimes also referred to as the *Penrose index* or the *Banzhaf-Penrose index*.

Compared to the absolute Banzhaf index, the normalized Banzhaf index not only has undesirable mathematical properties: from a computational point of view, the normalized Banzhaf index is also harder to compute, since (because of the denominator) we need to first calculate the raw Banzhaf indices of all players, before we can calculate the normalized Banzhaf index of one player. This is not the case for the absolute Banzhaf value, where the denominator 2^{n-1} is easy to compute.

For these reasons, nowadays the normalized Banzhaf index has gotten out of fashion.

Definition 31 (Banzhaf index)**.** In the literature, usually when one talks about the *Banzhaf index*, one means the absolute Banzhaf index. We will also use that convention in this survey: when we use the term *Banzhaf index*, we will mean the *absolute Banzhaf index*. Moreover, in the remainder of this text, we will not use the symbol β''_i to denote the absolute Banzhaf index of player i, but instead we use β_i, and we will use β to denote the vector of absolute Banzhaf indices of all players.

Remark 32. If we compare the second definition of the raw Shapley-Shubik index, definition 3.1, with definition 3.2 of the raw Banzhaf index, we see a remarkable resemblance between the two: in the raw Shapley-Shubik index, for each coalition $S \in A \setminus \{i\}$ for which $v(S) = 0 \wedge v(S \cup \{i\})$, we add $|S|!(|A| - |S| - 1)!$ to the index, while in the raw Banzhaf index, for the same coalitions we add simply 1 to the index. This is the only difference between those two indices.

3.2.3 An Example: Voting Power in the European Union

To illustrate the two indices that we just gave, we will use them to analyze the voting power in the European Union.

This example is taken from [2], where also an exact algorithm is explained for computing the Shapley-Shubik and Banzhaf indices in weighted multiple majority games. We will get to this algorithm in the next chapter.

For the European Union, two decision making rules are prescribed in the treaty of Nice. Both of these rules are weighted 3-majority games over four different weighted voting games $\{G_1, G_2, G_3, G_4\}$. The first decision rule is the weighted majority game $G_1 \wedge G_2 \wedge G_3$, the second one is $G_1 \wedge G_4 \wedge G_3$. All of these weighted voting games are over 27 players, each player representing a country in the European Union. The four weighted voting games are:

$$
\begin{aligned}
G_1 &= ((29,29,29,29,27,27,14,13,12,12,12,12,10,10,10,7,7,7,7,7,4,4,4,4,4,3),255),\\
G_2 &= ((1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1),14),\\
G_3 &= ((170,123,122,120,82,80,47,33,22,21,21,21,21,18,17,17,11,11,11,8,8,5,4,3,2,1,1),620),\\
G_4 &= ((1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1),18).
\end{aligned}
$$

The countries corresponding to the players are respectively Germany, United Kingdom, France, Italy, Spain, Poland, Romania, The Netherlands, Greece, Czech Republic, Belgium, Hungary, Portugal, Sweden, Bulgaria, Austria, Slovak Republic, Denmark, Finland, Ireland, Lithuania, Latvia, Slovenia, Estonia, Cyprus, Luxembourg and Malta.

In figure 3.1, for each country the Shapley-Shubik and Banzhaf power indices of countries are given for the first weighted multiple majority game (i.e. $G_1 \wedge G_2 \wedge G_3$), along with the population of each country.

3.3 Less Well-known Indices

There are a lot of power indices other than the Banzhaf and Shapley indices, although they aren't used as often. Nevertheless, they are considered important. We will explain in this section some of these indices.

3.3.1 The Deegan-Packel Index

Perhaps the most popular index among these lesser known indices is the *Deegan-Packel* index, proposed by Deegan and Packel (you guessed it) in [33]. In this index we consider

Countries	Population (millions)	Fraction of total population	Banzhaf index	Shapley-Shubik index
Germany	82.038	0.170	0.0778	0.0871
United Kingdom	59.247	0.123	0.0778	0.0870
France	58.966	0.123	0.0778	0.0870
Italy	57.612	0.120	0.0778	0.0870
Spain	39.394	0.082	0.0742	0.0799
Poland	38.667	0.080	0.0742	0.0799
Romania	22.489	0.047	0.0426	0.0399
The Netherlands	15.760	0.033	0.0397	0.0368
Greece	10.533	0.022	0.0368	0.0340
Czech Republic	10.290	0.021	0.0368	0.0340
Belgium	10.213	0.021	0.0368	0.0340
Hungary	10.092	0.021	0.0368	0.0340
Portugal	9.980	0.021	0.0368	0.0340
Sweden	8.854	0.018	0.0309	0.0281
Bulgaria	8.230	0.017	0.0309	0.0281
Austria	8.082	0.017	0.0309	0.0281
Slovak Republic	5.393	0.011	0.0218	0.0196
Denmark	5.313	0.011	0.0218	0.0196
Finland	5.160	0.011	0.0218	0.0196
Ireland	3.744	0.008	0.0218	0.0196
Lithuania	3.701	0.008	0.0218	0.0196
Latvia	2.439	0.005	0.0125	0.0110
Slovenia	1.978	0.004	0.0125	0.0110
Estonia	1.446	0.003	0.0125	0.0110
Cyprus	0.752	0.002	0.0125	0.0110
Luxembourg	0.429	0.001	0.0125	0.0110
Malta	0.379	0.001	0.0094	0.0082

Figure 3.1: Shapley-Shubik and Banzhaf indices for the first weighted 3-majority game used in the European Union

the set of minimal winning coalitions $W_{\min}$ of a simple game. Informally, for each minimal winning coalition $S \in W_{\min}$, the Deegan-Packel index is obtained by adding $1/|S|$ to each member in S.

The Deegan-Packel index is defined for monotone simple coalitional games, but can also be used for non-monotone simple coalitional games. However, clearly the Deegan-Packel doesn't make sense for coalitional games that are not monotone.

The Deegan Packel index is justified in situations where the *size principle* holds [50]:

> "In social situations similar to n-person, zero sum games with side payments, participants create coalitions just as large as they believe will ensure winning and no larger."

The formal definition of the Deegan-Packel index is:

Definition 33 (Deegan-Packel index)**.** For a simple coalitional game $(A = (a_1, \ldots, a_n), v)$, the *Deegan-Packel index* is defined as $\rho = (\rho_1, \ldots, \rho_n)$, where for $1 \leq i \leq n$:

$$\rho_i = \frac{1}{|W_{\min}|} \sum_{S \in \{S \in 2^A \mid a_i \in S \wedge v(S)=1 \wedge \forall a \in S : v(S \setminus \{a\})=0\}} \frac{1}{|S|}.$$

So, if the assumption is made that players will only form minimal winning coalitions, each minimal winning coalition has an equal probability of forming, and the players in a coalition divide the gains equally, then the Deegan-Packel index of a player represents the player's expected gain.

3.3.2 The Holler Index

The *Holler index* or *public good index* was first proposed in [29]. For player i, it is defined as the number of minimal winning coalitions that i appears in, divided by the sum of the sizes of all minimal winning coalitions.

Definition 34 (Holler index)**.** For a simple coalitional game $(A = (a_1, \ldots, a_n), v)$, the *Holler index* is defined as $h = (h_1, \ldots, h_n)$, where for $1 \leq i \leq n$:

$$h_i = \frac{|\{S \in 2^A \mid a_i \in S \wedge v(S) = 1 \wedge \forall a \in S : v(S \backslash \{a\}) = 0\}|}{\sum_{S \in \{S \in 2^A \mid v(S)=1 \wedge \forall a \in S : v(S \backslash \{a\})=0\}} 1}.$$

3.3.3 The Coleman Indices

As we said, the Banzhaf index is sometimes also called the Coleman index because Coleman was the first who came up with it. This is confusing, because Coleman also defined the following three other indices in [15].

Definition 35 (Coleman initiative power)**.** For a simple coalitional game $(A = (a_1, \ldots, a_n), v)$, the *Coleman initiative power index* is defined as $I = (I_1, \ldots, I_n)$, where for $1 \leq i \leq n$:

$$I_i = \frac{\sum_{S \in 2^A} v(S \cup \{a_i\}) - v(S)}{\sum_{S \in 2^A} 1 - v(S)}$$

So the Coleman initiative index for player i represents the fraction of losing coalitions that player i can turn into a winning coalition.

Definition 36 (Coleman preventive power)**.** For a simple coalitional game $(A = (a_1, \ldots, a_n), v)$, the *Coleman preventive power index* is defined as $P = (P_1, \ldots, P_n)$, where for $1 \leq i \leq n$:

$$P_i = \frac{\sum_{S \in 2^A} v(S \cup \{a_i\}) - v(S)}{\sum_{S \in 2^A} v(S)} \tag{3.3}$$

The Coleman preventive index for player i represents the fraction of winning coalitions that player i can turn into a losing coalition.

The last of Coleman's indices is not really an index: it is only defined on the grand coalition, and not on its individual players. It is simply equal to the fraction of all coalitions (including the empty one) that is a winning coalition.

Definition 37 (Power of a collectivity to act)**.** For a simple coalitional game $(A = (a_1, \ldots, a_n), v)$, the *power of a collectivity to act* is defined as:

$$A = \frac{\sum_{S \in 2^A} v(S)}{2^n} \tag{3.4}$$

3.3.4 The Johnston Index

The Johnston index was defined in [32]. The idea is here to look for all winning coalitions that a player occurs in, count per winning coalition the number of swings s in that coalition, and add $\frac{1}{s}$ to the index. Finally, the result is divided by the total number of winning coalitions that have swings, in order to normalize the index.

Definition 38 (Johnston index)**.** For a simple coalitional game $(A = (a_1, \ldots, a_n), v)$, the *Johnston index* is defined as $J = (J_1, \ldots, J_n)$, where for $1 \leq i \leq n$:

$$\begin{aligned} J_i &= \frac{J'_i}{\sum_{j=1}^{n} J'_j}, \\ J'_i &= \sum_{S \in \{S \in 2^A \mid a_i \in S \wedge v(S)=1 \wedge v(S \setminus \{a_i\})=0\}} \frac{1}{|S|}. \end{aligned}$$

3.3.5 The Chow Parameters

Chow parameters stem from the research done in boolean threshold functions by Chow [13]. A good discussion on Chow parameters from the viewpoint of simple coalitional games can be found in [55]. We are not sure though when Chow parameters was first incorporated into the theory of simple coalitional games.

It seems that the Chow parameters of a coalitional game are not considered as a real power index. Nevertheless, we mention Chow parameters here because obviously they *do* give us some useful information about the power of a player in a simple coalitional game, and certainly they are closely related to some power indices, such as the Banzhaf index.

Definition 39 (Chow parameters)**.** Given a simple coalitional game $(A = \{a_1, \ldots, a_n\}, v)$, the elements of the vector $(W_1, \ldots, W_n, W)$ are the *Chow parameters* of (A, v) iff W is the total number of winning coalitions of (A, v), and for all $1 \leq i \leq n$, W_i is the number of winning coalitions that player a_i occurs in.

The following theorem makes is what makes Chow parameters so interesting

Theorem 40 (Chow's theorem)**.** *The Chow parameters of a boolean threshold function (i.e. weighted voting game) uniquely determine that threshold function.*

Such results are not known for other power indices.

Chapter 4

Complexity of Power Index Computation

In this chapter we give complexity results for various problems regarding the computation of power indices. We first describe in section 4.1 some important complexity classes and we will introduce some of the relevant complexity theory related to these classes. Subsequently we describe some interesting problems in section 4.2, and we give the complexity-theoretic results for these problems. These will mostly be completeness results. Most of the time, we will refrain from giving proofs, but we refer to the literature instead. However, we might give a proof if the proof is short enough. Also, we might give an intuition instead of a proof.

4.1 Complexity Classes

A lot of the computational problems we will encounter are in the complexity class #P.

Definition 41 (#P)**.** Let $\Sigma = \{0,1\}$. A function $f : \Sigma^* \to \mathbb{N}$ is in #P iff there is a polynomial-time Turing machine M such that

$$f(x) = \{y \in \Sigma^* \mid M(x,y) \text{ accepts}\}.$$

Moreover, we require that the size of y is polynomial in x. Any $f \in$ #P can be seen as the problem of counting the number of accepting paths in the computation graph of a non-deterministic polynomial time turing machine.

With an *instance* of $f \in$ #P we mean an input $\in \Sigma^*$ for f.

Most "counting versions" of NP-complete problems are in #P. Consider for example the satisfiability problem (*SAT*). SAT is the set of propositional formulas that are satisfiable, that is, the set of propositional formulas φ for which there exists some truth assignment to the variables occuring in φ that makes φ true. In the decision version of this problem we are asked whether there is such an assignment, given a propositional formula. In the counting version of this problem, called *#SAT* we are asked how many satisfying truth-assignments exist. The function that counts this is in #P. This is not hard to see: A non-deterministic polynomial time Turing machine M for the decision problem could non-deterministically

choose a truth-assignment, and accept if the truth-assignment satisfies the formula. In this case, every distinct computation path corresponds to a different truth-assignment, hence counting the number of computation paths of M is the same as counting the number of different truth-assignments that satisfy the formula. It follows that #SAT $\in$ #P.

Remark 42. Note that not all counting versions of *NP*-complete problems are in #P. Consider the problem #CIRCUMSCRIPTION: Given a propositional formula φ over the boolean variables $(x_1,\ldots,x_n)$, determine the number of minimal models for φ. A minimal model is a truth-assignment to $(x_1,\ldots,x_n)$ that satisfies φ and that has as few as possible variables set to true, i.e. there exists no other satisfying truth-assignment that has less variables set to true. The decision version of this problem is clearly NP-complete: if φ is satisfiable, only then it has a minimal model. However, the number of accepting paths on this Turing machine of the decision problem is unequal to the number of minimal models. Instead, it is equal to the number of satisfying assignments. In fact, it turns out that there is no non-deterministic polynomial time Turing machine for which we can count the number of accepting paths in order to obtain the number of minimal models. The problem #CIRCUMSCRIPTION turns out to not be #P-complete. Instead it is complete for the harder class #NP. For a definition and a proof, see [21].

There exist complete problems for #P, however, in the literature there are different notions of completeness for this class. We will see that, according to a reasonable definition of completeness #P, #SAT is #P-complete.

The class #P was originally defined by Valiant in [57], where he proved that the complexity of computing the permanent is #P-complete under Cook reductions (i.e. polynomial-time Turing reductions):

Definition 43 (Cook reduction)**.** Given two problems Π_1 and Π_2, a *Cook reduction* from Π_1 to Π_2 is a Turing machine which solves instances of Π_1 in polynomial time and makes use of an oracle for Π_2.

Computing the permanent of a matrix is equivalent to the problem of finding the number of matchings in a bipartite graph. The decision variant of this problem lies in P. It turns out that there are a lot of problems in P for which the counting version is P-complete. So, interestingly, there are NP-complete problems for which the counting version is P-complete, as well as problems in P for which the counting version is P-complete.

Cook reductions seem like a reasonable notion for completeness of #P-problems: If we have a Cook reduction from $f_1 \in$ #P to $f_2 \in$ #P, then we are be able to compute f_1 in polynomial time if there is an algorithm that computes f_2 in polynomial time. Nevertheless, stronger notions of completeness have been defined for #P. This is partly due to the fact that a #P-problem f that is complete under Cook-reductions, is not necessarily "rich" enough to "represent" all other problems in #P. By this we mean that it is not necessarily possible to encode the instances of any #P-problem g into one single instance x of f. Also it's partly due to the fact that stronger notions of completeness have some interesting properties (e.g. closure-properties for certain sets, or interesting relations to other classes), but we will not go into these details here. We will now define formally some of the stronger notions of #P-completeness.

Definition 44 (Reductions for #P problems)**.**

- A *metric reduction* from $f \in \#\mathsf{P}$ to $f' \in \#\mathsf{P}$ is a pair of polynomial-time computable functions (φ, ψ) such that for any $x \in \Sigma^*$, $f(x) = \psi(x, f'(\phi(x)))$. Defined by Krentel in [35].

- A *many-one reduction* from $f \in \#\mathsf{P}$ to $f' \in \#\mathsf{P}$ is a pair of polynomial-time computable functions (φ, ψ) such that for any $x \in \Sigma^*$, $f(x) = \psi(f'(\phi(x)))$. Defined by Zankó in [59].

- A *parsimonious reduction* from $f \in \#\mathsf{P}$ to $f' \in \#\mathsf{P}$ is a polynomial-time computable function ψ such that for any $x \in \Sigma^*$, $f(x) = \psi(x)$. Defined by Simon in [54].

Please note that a parsimonious reduction is simply a special case of a many-one reduction, and a many-one reduction is simply a special case of a metric reduction. Intuitively, we can interpret these three types of reductions as follows: a parsimonious reduction is a polynomial-time algorithm that converts one problem instance to another problem instance, while preserving the number of reductions. A many-one reduction is a pair of polynomial-time algorithms, one of which converts a problem instance A to another problem instance B; and one of which computes the number of solutions for A, given the number of solutions for B. Finally, a metric reduction is the same as a many-one reduction, except for that we now require information from the original instance A in order to compute the number of solutions for A, given the number of solutions for B.

Given these types of reductions, we can define the following four notions of #P-completeness.

Definition 45 (#P-completeness)**.** A function $f \in \#\mathsf{P}$ is #P-*Cook-complete* if there is a Cook reduction from any $f' \in \#\mathsf{P}$ to f. #P-*metric-completeness*, #P-*many-one-completeness* and #P-*parsimonious-completeness* are defined analogously.

Clearly, if a #P-problem is parsimonious-complete then it is also many-one-complete, and if a #P-problem is many-one-complete, then it is also metric-complete.

The class of decision problems that is most closely related to #P is the class PP.

Definition 46. PP is the class that contains all languages L for which there is a probabilistic polynomial-time turing machine M such that

$$x \in L \Rightarrow \mathbf{Pr}(M(x) \text{ accepts }) > 1/2,$$
$$x \notin L \Rightarrow \mathbf{Pr}(M(x) \text{ accepts }) < 1/2.$$

The abbreviation PP stands for *Probabilistic Polynomial* time. PP is also frequently pronounced as *majority-P*. This is because PP is exactly the class that contains all languages L that have a non-deterministic polynomial-time Turing machine where more than half of the computation paths are accepting iff $x \in L$.

4.2 Computational Problems

In the literature, there are various complexity results related to the power indices. We will give them here. Most of these results are concerned with weighted majority games.

There are some natural decision problems that arise immediately from the definition of weighted majority games and power indices. In [41] and [49], NP-completeness is established for a variety of these decision problems.

Definition 47 (PIVOT)**.** The language PIVOT is defined as

$$\{\langle W = (w_1, \ldots, w_n), q, i\rangle \mid \beta'_i > 0 \text{ in the weighted majority game } (W, q)\}.$$

where β'_i is the number of swings (called 'pivots' in [49]), or equivalently, the raw Banzhaf index for player i.

Theorem 48. *PIVOT is* NP*-complete.*

Proof. The first proof was by Prasad and Kelly in 1990, in [49]. We refer to this paper for a formal proof. Intuitively, NP-completeness of this problem gets obvious if you realize the following: for any $\langle W = (w_1, \ldots, w_n), q, i\rangle \in$ PIVOT, there must be a $S \subseteq W \backslash w_i$ such that

$$q - w \leq \left(\sum_{w \in S} w \right) < q.$$

Deciding whether there exists such an S is equivalent to deciding a generalized version of the SUBSET-SUM problem. □

Matsui and Matsui proved in [41] something slightly stronger:

Theorem 49. *PIVOT remains* NP*-complete if i is fixed at n, and $\forall i, j : i > j \Rightarrow w_i > w_j$. In other words, PIVOT remains* NP*-complete if i is restricted to be the player with lightest weight.*

Also, in the same paper it is showed that theorem 49 holds if we replace β'_i is replaced by φ'_i: the raw Shapley-Shubik index.

It is easy to generalize the results of theorems 48 and 49 into the following two theorems (these are not taken from any existing literature). The first of the two shows that PIVOT is fixed-parameter tractable if the agents are ordered ascendingly on their weights, and if the fixed parameter is the agent number. The second of the two shows that PIVOT is NP-complete if the agents are ordered descendingly, and if the agent number is fixed.

Theorem 50. *The languages PIVOT-iTH-HEAVIEST are in* P *for all $i \in \mathbb{N}$. They are defined as:*

$$\{\langle W = (w_1, \ldots, w_n), q\rangle \mid \beta'_i > 0 \textit{ in the weighted voting game } (W, q) \wedge \forall i, j : i < j \rightarrow w_i \geq w_j\}.$$

Here, β'_i is the number of coalitions that i is critical in a.k.a. the raw Banzhaf index for player i.

Proof. Let's first consider the language PIVOT-1TH-HEAVIEST. Clearly, in a weighted voting game $(W = (w_1, \ldots, w_n), q)$ where $w_1 \geq w_2 \geq \cdots \geq w_n$, clearly w_1 there is always a coalition in which w_1 is critical as long as $0 < q \leq \sum_{w \in W} w$. So there is an $O(n)$-time algorithm for PIVOT-1TH-HEAVIEST. Actually, it is even possible to find a coalition for which player i is critical: consider the grand coalition and keep removing the lowest-weighted players until the sum of the weights of the players in the coalition drops below q. If that happens, add the last player w_j that was removed. The resulting coalition is a swing coalition for player 1, because $w_1 \geq w_j$.

For the problems PIVOT-iTH-HEAVIEST, with $i > 1$ we can generalize the idea: In a weighted voting game $(W = (w_1, \ldots, w_n), q)$ where $w_1 \geq w_2 \geq \cdots \geq w_n$, player i has a pivot coalition if and only if at least one of the following coalitions is a swing coalition for i :

$$C = \{s_{<i} \cup s_{\geq i} \mid s_{<i} \in S_{<i} \wedge s_{\geq i} \in S_{\geq i}\},$$

where $S_{<i}$ is the collection of subsets $s \subseteq W$ that contain only weights heavier than w_i, and $S_{\geq i}$ consists of the subsets s of W that contain w_i, and if $w_j \in s$ for any $j > i$, then $w_{j-1} \in S_{\leq i}$.

This is true because we can transform any arbitrary swing coalition for i into a swing coalition for i in C: Consider a swing coalition D for i that is not in C. Remove all the weights lower than w_i from D to obtain D'. If the resulting coalition is a swing coalition, then we're done. Otherwise, we have $\sum_{d \in D'} d < q - w_i$, so we iteratively add the highest weight lower than w_i that's not in D'. Clearly at some iteration we will have $q - w_i \leq \sum_{d \in D'} < q$. At that point, D' is in C.

We can enumerate all of the coalitions in C, and check whether each coalition is a swing coalition for i. The cardinality of C is $2^{i-1}(n-i)$, so we can decide PIVOT-iTH-HEAVIEST in linear time. □

Theorem 51. *The languages PIVOT-iTH-LIGHTEST are* NP*-complete for all* $i \in \mathbb{N}$*. They are defined as:*

$$\{\langle W = (w_1, \ldots, w_n), q\rangle \mid \beta'_i > 0 \text{ in the weighted voting game } (W, q) \wedge \forall i, j : i < j \rightarrow w_i \leq w_j\}.$$

Here, β'_i *is the number of swings a.k.a. the raw Banzhaf index for player i.*

Proof. For $i = 1$, it has already been proved in [41], by a polynomial-time reduction (a.k.a. Karp reduction) from PARTITION. It is very easy to extend it to reductions for the cases that $i > 1$. We give a family of Karp reductions from PIVOT-1TH-LIGHTEST to PIVOT-iTH-LIGHTEST for any i. Given an instance $\langle W, q\rangle$ of PIVOT-1TH-LIGHTEST, append $i-1$ weights with value $\frac{1}{i-2}$ to the beginning of W to get W'. Because (W, q) is a weighted voting game, q and all weights in W are integer. The newly added weights are not integer, but if we would allow the weights in weighted voting games to be rational numbers, then obviously the new players in (W', q) would all be dummy players. We eliminate the rationals from (W', q) by multiplying all weights by $i-2$ and multiplying q by $i-2$. In the resulting game we still have that the first $i-1$ players are dummy players. So if $\beta'_1 > 0$ in the PIVOT-1TH-LIGHTEST-instance, then $\beta'_i > 0$ in the PIVOT-iTH-LIGHTEST-instance, and vice versa. Therefore, all of the problems PIVOT-iTH-LIGHTEST are NP-complete. □

The following modifications of PIVOT are also NP-complete.

Theorem 52. *If, in the definition of PIVOT, we replace β_i' by β_i (the Banzhaf index of player i), β_i^n (the normalized Banzhaf index of player i) or φ_i (the Shapley-Shubik index of player i), then the problem remains* NP*-complete.*

Proof. Given in [49]. It follows immediately from the fact that β_i', β_i and φ_i are greater than 0 iff $\beta_i' > 0$. □

[49] gives another NP-completeness result:

Theorem 53. *If, in the definition of PIVOT, we replace $\beta_i' > 0$ by $\beta_i' > r$ for any fixed r between 0 and $\sum_{w \in W} w$, then the resulting problem remains NP-complete.*

As a final NP-completeness result, we want to mention that in [41] it is proved that:

Theorem 54. *It is* NP*-complete to decide for any two players i and j with $w_i > w_j$ whether $f_i' > f_j'$ when we choose β (the Banzhaf index) or φ (the Shapley-Shubik index) for f[1]. This holds even if we require that i and j are the two heaviest-weighted players.*

Remark 55. Of course, for restrictions of the problem in theorem 54, NP-completeness need not hold. In [40] it is pointed out that there is a linear-time algorithm for deciding the problem in the case that i and j are the heaviest-weighted and the lightest-weighted player respectively. More precisely two players in a weighted voting game are symmetric iff

$$\forall S \subseteq W \setminus \{w_i, w_j\}: \qquad q - w_i \leq \left(\sum_{w \in S} w \right) < q$$
$$\leftrightarrow \quad q - w_j \leq \left(\sum_{w \in S} w \right) < q.$$

The authors show NP-completeness for deciding symmetricity between two players, but give a polynomial time algorithm for the special case that i is the player with the heaviest weight and j is the player with the lightest weight.

Let's now consider some function problems, instead of decision problems. Of course, in the case of a weighted majority game, the most interesting question that concerns us for each power index is: what is the complexity of computing the power index in a weighted majority game? Again, in the literature, this question is exclusively covered for the Shapley-Shubik and Banzhaf indices.

Let us look again at the complexity of the decision problems we just described. We see that it is already NP-complete to decide whether the (raw) Banzhaf and (raw) Shapley-Shubik indices equal zero. Loosely stated, this means that it will be at least NP-hard to decide what the precise value of the power index is.

The first result for this question is for the raw Banzhaf index, given implicitly in [49]:

[1] *Abusing notation a little bit.*

Theorem 56. *Let $(W = (w_1, \ldots, w_n), q)$ be any weighted voting game. Let #PIVOT-WVG $: \Sigma^* \to \mathbb{N}$ be the function that returns the raw Banzhaf index β_i' for game (W, q) when it is given input $\langle W, q, i \rangle$. #PIVOT-WVG is* #P*-parsimonious-complete.*

Proof. The proof is roughly as follows. The authors point out that the reduction for theorem 48 is a reduction from KNAPSACK that preserves the number of solutions. The counting version of this problem, #KNAPSACK, is #P-parsimonious-complete. Hence, #PIVOT is also #P-parsimonious-complete. □

Aziz gives in [4] a theorem that is in a sense stronger:

Theorem 57. *Let $(A, W_{\min})$ be a monotone simple game in minimal winning coalition form. Let #PIVOT-MWC $: \Sigma^* \to \mathbb{N}$ be the function that returns the raw Banzhaf index β_i' for game $(A, W_{\min})$ when it is given input $\langle A, W_{\min}, i \rangle$. #PIVOT-MWC is* #P*-metric-complete.*

This is proven by a reduction from the following problem, which is also #P-metric-complete by a fairly straightforward reduction from vertex cover.

Definition 58 (NUMWINNING-MWC)**.** Let $(A, W_{\min})$ be a monotone simple game in minimal winning coalition form. #NUMWINNING-MWC $: \Sigma^* \to \mathbb{N}$ is the function that returns the number of winning coalitions for game $(A, W_{\min})$ when it is given input $\langle A, W_{\min}, i \rangle$. #PIVOT-MWC is #P-metric-complete.

Because NUMWINNING-MWC is #P-complete, we get the following corollaries:

Corollary 59. *Computing the Coleman collectivity index of a weighted voting game in minimal winning coalition representation is* #P*-complete.*

Corollary 60. *Determining the Chow parameters of a weighted voting game in minimal winning coalition representation is* #P*-complete.*

The author strengthens this last corollary and proves that it also holds for weighted voting games.

In a theorem in [49] that is subsequent to theorem 56, the authors point out that the function that returns the (non-raw) Banzhaf index instead of the raw Banzhaf index, is also #P-complete. Indeed, the Banzhaf index is equal to the raw Banzhaf index divided by 2^{n-1}, where n is the number of players in the weighted majority game. This means that we could compute the raw Banzhaf index in polynomial time if we are given an oracle that returns the Banzhaf index. However, the problem with the claim is that the Banzhaf index is a rational number, and a function in #P must have $\mathbb{N}$ as a target. So, strictly speaking, the function that returns the Banzhaf index for a certain player in a weighted majority game is not in #P.

In [18], such an issue again arises. In the same way, the authors again slightly abuse the definition of #P, and prove that computing the Shapley-Shubik index in weighted majority games is #P-complete. The Shapley-Shubik index is a rational, so it can't be the case. If we look at it strictly, they show the following implicitly:

Theorem 61. *Let $(W = (w_1, \ldots, w_n), q)$ be any weighted voting game. Let #RAWSHAPLEY $: \Sigma^* \to \mathbb{N}$ be the function that returns the raw Shapley-Shubik index β_i' for game (W, q) when it is given input $\langle W, q, i \rangle$. #RAWSHAPLEY is* #P*-metric-complete.*

Faliszewski and (L.) Hemaspaandra sharpen this result. They show in [23] that:

Theorem 62. *#RAWSHAPLEY is* #P*-many-one-complete.*

Theorem 63. *#RAWSHAPLEY is not* #P*-parsimonious-complete.*

The proofs of these two theorems are interesting but complicated. In the same paper, the complexity of comparing the power-indices of a player in two different weighted majority games is investigated. Through the #P-parsimonious-completeness, with relative ease the authors are able to show that this problem is PP-complete. For the Shapley-Shubik index they also manage to prove PP-completeness, although with a more complicated proof because it is not #P-parsimonious-complete.

Theorem 64. *The languages SHAPLEYCOMPARE and BANZHAFCOMPARE are:*

$$\begin{aligned} \textit{SHAPLEYCOMPARE} &= \{\langle W_1, q_1, W_2, q_2, i\rangle \mid |W_1| = |W_2| \wedge \varphi_i((W_1, q_1)) > \varphi_i((W_2, q_2))\}, \\ \textit{BANZHAFCOMPARE} &= \{\langle W_1, q_1, W_2, q_2, i\rangle \mid |W_1| = |W_2| \wedge \beta_i((W_1, q_1)) > \beta_i((W_2, q_2))\}. \end{aligned}$$

SHAPLEYCOMPARE and BANZHAFCOMPARE are both PP*-complete.*

These problems are interesting for the scenario of weighted voting games in open anonymous environments, where the players can choose to cooperate by merging their weight into a single false-name identity, in order to increase their power. As we mentioned in chapter 1, Bachrach and Elkind investigate this problem in [7].

There are of course domains other than weighted voting games for which we can calculate power indices. Unfortunately, for those domains the computational complexity has not been studied very deeply. There is one exception to this rule: power indices in games related to network reliability. Bachrach et al study this kind of games in [9] and [10].

As a last result we want to give, in [4], the complexity for two less standard indexes, the Holler and Deegan-Packell indices, is determined for games in minimal winning coalition form.

Theorem 65. *Computing the Holler index for a game G can be done in linear time if G is represented in minimal winning coalition form. Computing the Deegan-Packel index for a game G can be done in linear time if G is represented in minimal winning coalition form.*

Chapter 5

Exact Methods for Calculating Power Indices

Looking at the complexities of the problems in the previous chapter, we see that the problems of computing the raw Shapley-Shubik index and raw Banzhaf index are both #P-complete. Therefore, we can not expect to find an algorithm that computes these indices within polynomial time. Most of the algorithms that have up till now been proposed are meant for computing the Banzhaf or Shapley-Shubik indices in weighted voting games. However, some algorithms for more general and more specific cases exist as well.

In this chapter, we discuss only exact algorithms for weighted voting games. Other types of games for which algorithms have been proposed include:

- The network reliability games that we mentioned in the first chapter. A method for computing the Banzhaf index in various kinds of network games is given in [9] and [10].

- Decomposed characteristic function games. In [16], Conitzer and Sandholm propose a (fairly basic) method for computing the Shapley value for games in which the characteristic function is decomposed into several "subfunctions".

- Games represented as marginal contribution nets. There is a polynomial time algorithm for computing the Shapley value of such games, and every coalitional game can be represented as such a net. However, it takes exponential time to convert a coalitional game into a marginal contribution net. See [30].

For this chapter, we will start with discussing the naive enumeration algorithm, and then move on to discuss more sophisticated algorithms: the improvement of Klinz and Woeginger, the generating function method, and an extention and improvement on the method. This list of power index algorithms for weighted voting games is not exhaustive: we have omited the enumeration algorithms of Matsui & Matsui [40] for the Banzhaf, Shapley-Shubik, and Deegan-Packel indices.

5.1 The Direct Enumeration Algorithm

In this section and the following, we will make use the O^*-notation to disregard sub-exponential factors. For example $O(n^2 \log(n) 2^n) = O^*(2^n)$.

Let's say that we want to compute the raw Banzhaf index of the players in a weighted voting game. An exact algorithm that directly comes to mind when thinking about the problem, is the following.

1. We are given the weighted voting game (W, q) on n players ($|W| = n$).
2. Initialize for $1 \leq i \leq n$ the raw Banzhaf index β'_i of player i to 0.
3. Enumerate all subsets C of the set of players $\{1, \ldots n\}$.
4. If $w(C)$ is greater than or equal to q, then check for each player i in C whether $w(C \backslash \{i\})$ is smaller than q.
5. If so, add 1 to β'_i.

This algorithm runs in time $O(n^2 2^n)$. In the O^*-notation this comes down to $O^*(2^n)$. It is easily seen that for all the other power indices, a similar method of $O^*(2^n)$ time exists.

5.2 Klinz & Woeginger's Improvement

Klinz and Woeginger improve in [34] the algorithm of the previous section. They consider algorithms for computing the *normalized* Banzhaf index of a single player and the Shapley-Shubik index of a single player. Clearly, if we use the trivial method then this will cost us $O(n^2 2^n)$ time for the normalized Banzhaf index (because we need to compute the raw Banzhaf index of all players in order to find out what the denominator of the normalized Banzhaf index should be), and $O(n2^n)$ time for the Shapley-Shubik index (directly implied by the alternative definition of the Shapley-Shubik index: equation 3.1).

By using a partitioning trick, Klinz and Woeginger manage to reduce these $O^*(2^n)$-time algoritms into $O^*((\sqrt{2})^n)$-time algorithms (on the tradeoff of space).

First, they consider the following problem. We are given three integers p, L, U, $L \leq U$, and a cost function $C : \{1, \ldots, 2p\} \to \mathbb{N}$. We are also given four sequences of non-negative integers:

- $x_1, \ldots x_M$,
- $y_1, \ldots y_N$,
- $\alpha_1, \ldots, \alpha_M$,
- $\gamma_1, \ldots, \gamma_N$,

with the restriction that all integers in the last two sequences are not larger than p. Now define the characteristic function $\chi : \mathbb{N} \to \{0,1\}$ as follows.

$$\chi(k) = \begin{cases} 1 & \text{if } L \leq k \leq U, \\ 0 & \text{otherwise} \end{cases}$$

The problem now, is to compute

$$\sum_{1 \leq i \leq M, 1 \leq j \leq N} \chi(x_i + y_j) C(\alpha_i + \gamma_j).$$

There exists an algorithm for this problem that runs in time $O(M \log M + N \log N + nM)$. Although this algorithm is not that complicated, we will not give it here; we refer the interested reader to [34] instead.

Using this algorithm, which we will call the *auxiliary algorithm*, we can compute the Shapley-Shubik index for a player i as follows (the algorithm for the Normalized Banzhaf index is very similar).

1. Partition the set of players $\{1, \ldots, i-1, i+1, \ldots, n\}$ into disjoint sets A and B with $|A| = \lfloor (n-1)/2 \rfloor$ and $|B| = \lceil (n-1)/2 \rceil$.

2. A swing coalition S for i contributes $|S|!(n-|S|-1)$ to the raw Shapley-Shubik index of player i. Lets consider an arbitrary coalition $S \subseteq A \cup B$ and let S_A and S_B be the parts of S that are in A and B respectively. Now we have that $S \cup \{i\}$ is a swing coalition for i if and only if $w(S_A) + w(S_B) \geq q - w_i$ and $w(S_A) + w(S_B) \leq q - 1$.

3. So what we do next is: we convert the entire situation into a large instance of the auxiliary problem and solve it. We set L to $q - w_i$ and we set U to $q - 1$. For all subsets S_i of A, set x_i to $w(S_i)$ and set α_i to $|S_i|$. In the same way, for all subsets S_i of B, set y_i to $w(B_i)$ and set γ_i to $|S_i|$. Finally, we define the cost function as follows:

$$C(n) = \begin{cases} k!(n-k-1)! & \text{if } 0 \leq k \leq n-1, \\ 0 & \text{otherwise.} \end{cases}$$

4. If we solve this instance of the auxiliary problem, we obtain the raw Shapley-Shubik index of player i. Hence, the last thing we need to do is to divide by $n!$.

Solving an instance of the auxiliary problem can be done in polynomial time. However, the instance size of the auxiliary problem that we generate is exponential in the size of the weighted voting game. The size of M is $2^{|A|} = 2^{\lfloor (n-1)/2 \rfloor}$: we generate an x and an α for each subset of A. Similarly, the size of N is $2^{|}B| = 2^{\lceil (n-1)/2 \rceil}$. Substituting these values in the description of the time-complexity of the auxiliary problem, we get a time-complexity of $O(2^{\lfloor (n-1)/2 \rfloor} \log 2^{\lfloor (n-1)/2 \rfloor} + 2^{\lceil (n-1)/2 \rceil} \log 2^{\lceil (n-1)/2 \rceil} + n2^{\lfloor (n-1)/2 \rfloor}) = O(n(\sqrt{2})^n) = O^*((\sqrt{2})^n)$.

5.3 The Generating Function Approach

In [39], Mann and Shapley use a different algorithm for computing Shapley-Shubik indices. This algorithm based on a generating function by Cantor, given in [37]. Mann and Shapley use this method to evaluate the power indices of the US electoral college exactly (previously, they had to use a Monte Carlo approximation methods; see the next chapter).

Later, Brams and Affuso find the generating function for the Banzhaf index [12], yielding a similar method for computing the Banzhaf index in weighted voting games.

In [11], these methods are used to compute the Banzhaf and Shapley-Shubik indices for the Council of Ministers of The European Union. Moreover, in that paper an algorithm is proposed to compute the power indices in weighted 2-majority games, also based on generating functions. In [2], this idea is extended to work for any weighted n-majority game, see section 5.4.

The actual algorithm that resulted from obtaining these generating functions is a dynamic programming algorithm that can easily be explained without even mentioning generating functions. This is done in [40], where a slightly adapted version of the algorithm is presented and also the algorithm is adapted to compute the Deegan-Packel index.

For this survey, we do give the algorithms for the Shapley-Shubik and Banzhaf indices from the viewpoint of generating functions. After that we will give the related algorithm for the Deegan-Packel index.

Definition 66 (Generating function)**.** The *(ordinary) generating function* of a function $f(k)$: $\mathbb{N} \to \mathbb{R}$ is the formal power series

$$\sum_{k=0}^{\infty} f(k)x^k.$$

For such a power series we are not interested in questions related to convergence, and we are also not interested in the evaluation of the series for a particular x, that's why the power series is called formal. We actually only care about the coefficients of the powers series.

5.3.1 A Generating Hunction for the Shapley-Shubik Index

To obtain a generating function that's suitable for computing the Shapley-Shubik index φ_i of player i in some simple weighted voting game v on n players, we must realize the following (follows from equation 3.1):

Proposition 67. *Let $A_i(k,j)$ be the number of coalitions of j players that do not contain player i and have a total weight equal to k. Then the Shapley-Shubik index is given by:*

$$\varphi_i = \sum_{j=0}^{n-1} \frac{j!(n-j-1)!}{n!} \left(\sum_{k=q-w_i}^{q-1} A_i(k,j) \right).$$

There is a generating function for $A_i(k,j)$.

Theorem 68. *Let (W,q) be a weighted voting game. The generating function of $A_i(k,j)$ for this game is $G_{A_i}(z,x) = \prod_{j \neq i}(1+zx^{w_j})$.*

Proof. Consider the function:

$$\prod_{j\geq 1}(1+zx^{w_j}) = \sum_{T\subseteq W} z^{|T|}x^{\sum_{w\in T} w} = \sum_{k\geq 0, j\geq 0} A(k,j)x^k z^j.$$

$A(k,j)$ is here the number of coalitions of size j with weight k. If we drop the factor $(1+zx_i^w)$, then we get the coefficients $A_i(k,j)$ instead. □

Now that we have a generating function for A_i, how do we use it to obtain an algorithm for computing the Shapley-Shubik index? Clearly, what we have to do is obtain the coefficients of the generating function. We use dynamic programming for this. First, we observe that

$$\begin{aligned}
\prod_{j\geq 1}(1+zx^{w_j}) &= (1+zx^{w_1})\prod_{j\geq 2}(1+zx^{w_j}) \\
&= (1+zx^{w_1}+zx^{w_2}+z^2x^{w_1+w_2})\prod_{j\geq 3}(1+zx^{w_j}) \\
&= \cdots \\
&= \sum_{k\geq 0, j\geq 0} A(k,j)x^k z^j.
\end{aligned}$$

We can see the "building up" of this polynomial form as an iterative algorithmic process. We start in iteration 0 with the polynomial form "1", and in each iteration j we multiply the polynomial from iteration $j-1$ with the factor $(1+zx^{w_j})$ and convert it into polynomial form again. Per iteration, we only need to keep track of the two dimensional array of coefficients of the polynomial. We use the array from iteration $j-1$ to build up the array of coefficients for iteration j. After iteration n, the array will contain the coefficients A_i, and by a single scan through the array we can then compute the Shapley-Shubik index for player i. For iteration 0, this array is initialized to all zero's, except for element $(0,0)$ which is initialized to 1.

As for the complexity of this algorithm: firstly, we use the assumption that basic arithmetic takes unit time. In iteration 0 we initialize the array. The size of this array is $n \times q$. In each subsequent iteration, we execute at most $n \times q$ steps that take a constant amount of time. There are n iterations. After the last iteration, we scan the array of the nth iteration again, which takes nq steps again (but this can also be done simultaneously with the last iteration). Altogether, the time complexity is $O(n^2q)$ (please note that this is *pseudo*polynomial). For the space complexity: we can use two $n \times q$ arrays that we overwrite each two iterations. Therefore, the space complexity is $O(n^2q)$.

5.3.2 A Generating Function for the Banzhaf Index

For the Banzhaf index, we use a similar approach, but a different generating function.

Proposition 69. *Let $b_i(k)$ be the number of coalitions that do not contain player i and have a total weight equal to k. Then the Banzhaf index is given by:*

$$\beta_i = \frac{1}{2^{n-1}} \sum_{k=q-w_i}^{q-1} b_i(k).$$

There is a generating function for $b_i(k)$.

Theorem 70. *Let (W,q) be a weighted voting game. The generating function of $b_i(k)$ for this game is $G_{b_i}(x) = \prod_{j\neq i}(1+x^{w_j})$.*

Proof. Consider the function:

$$\prod_{j\geq 1}(1+x^{w_j}) = \sum_{T\subseteq W}\prod_{w\in T}x^w = \sum_{T\subseteq W}x^{\sum_{w\in T}w} = \sum_{k\geq 0}b(k)x^k.$$

$b(k)$ is here the number of coalitionswith weight k. If we drop the factor $(1+x_i^w)$, then we get the coefficients $b_i(k)$ instead. □

With this generating function in mind, we can use the same dynamic programming approach as we did for the Shapley-Shubik index. The only difference is that we can use a one-dimensional array instead of a two-dimensional one, because this is a generating function of only one parameter. The size of this array is q. So our time complexity for this algorithm is $O(nq)$ and our space complexity is $O(n+q)$.

5.3.3 Computing the Deegan-Packel Index

For the deegan packel index, a generating function is not (yet) known, but a related dynamic programming algorithm has been discovered in [40].

Given a weighted voting game $(W,q), |W| = n$, we partition the agents in weight-equality classes $N_1,\ldots,N_m$. Such that if the weight of any two players is equal, they're placed in the same class, and if the weight of any two players is unequal, then the player with the higher weight is in a class with a lower index.

Now we define the numbers $c_i(w,t,x)$ as

$$\#\{S\subseteq W\backslash\{i\} \mid \sum_{w_j\in S}w_j = w \wedge |S| = t \wedge N_x\cap S\neq\varnothing \wedge \forall z > x : N_z\cap S = \varnothing\}.$$

Let's say that player i is in the weight-equality class N_y. Then, the Deegan-Packel index ρ_i of player i is equal to

$$\left(\left(\sum_{x=1}^{y}\sum_{w=q-w_i}^{q-1}\sum_{t=1}^{n-1}\frac{c_i(w,t,x)}{t+1}\right)+\left(\sum_{x=y+1}^{m}\sum_{w=q-w_i}^{q-1-w_i+w_x}\sum_{t=1}^{n-1}\frac{c_i(w,t,x)}{t+1}\right)\right)\frac{1}{|W_{\min}|}. \tag{5.1}$$

The first part of the expression between the brackets counts the number of coalitions for which i is a swing player and for which all the other players have a higher weight than i. If that's the case then we automatically know that all players in the coalition are swings, hence the coalition is a minimal winning coalition. The second part of the expression between the brackets counts all coalitions for which i is a swing player and for which the total weight of the coalition is no larger than $q-1$ plus the lightest weighted player in the coalition. If that is the case, then clearly we're sure that the coalition is a minimal winning coalition. The

factor outside of the brackets divides everything by the total number of winning coalitions, in order to normalize the index.

We can compute the expression inside the brackets by adapting our existing algorithm for the Shapley-Shubik index. The key is to realize that

$$c_i(w,t,m) = A_{w,t}. \tag{5.2}$$

Remember for equation 5.2 that we defined m to be the number of weight-equality classes, so $m \leq n$. Unfortunately, having the values for all $A(w,t)$ is not enough for computing the Deegan-Packel index. We need to know all values of all the individual $c_i(w,t,x)$. Naively, one would say that we need a dynamic programming method that generates an $n \times n \times q$ array in each iteration, yielding an algorithm that runs in time $O(n^3q)$ and space $O(n^2q)$. This is not the case. We simply use our original method to compute $A(w,t)$, but prior to that we do the following.

Let's say we are given a weighted voting game $(W,q), |W| = n$. We first sort the weights (and implicitly: the players) descendingly to obtain the game $(W^{\downarrow} = \{w_1,\ldots,w_n\},q)$. Assume for ease of explanation that no two players have the same weight, so that there are exactly n weight-equality classes, and each weight-equality class N_i contains the single agent with weight w_i.

Now consider the dynamic programming algorithm on $(W^{\downarrow},q)$ to compute the $A_i(w,t)$. If we run this algorithm on $(W^{\downarrow},q)$, then in each iteration i of the dynamic programming process, we actually generate the numbers $c_i(w,t,i)$. To see this, realize that under our assumption that our weights are all unequal and are sorted descendingly :

1. The number $c_i(w,t,i)$ is nothing more than the computation of the numbers $A(w,t)$ on the subgame induced on the first i agents of $(W^{\downarrow},q)$.

2. The dynamic programming algorithm computes $A_i(w,t)$ for game $(W^{\downarrow},q)$, by starting with computing the $A(w,t)$ values for the subgame of $(W^{\downarrow},q)$ induced on the empty set of agents (all 0, except $A_i(0,0)$. Subsequently the algorithm computes in iteration $j \geq 1$ the values of $A_i(w,t)$ for the subgame of $(W^{\downarrow},q)$ induced on the highest i agents, *given* the values of $A_i(w,t)$ for the subgame of $(W^{\downarrow},q)$ induced on the highest $i-1$ agents.

Hence, throughout the dynamic programming process we actually encounter all of the $c_i(w,t,i)$, and once we encounter such a value that is used in equation 5.1, we can make computations accordingly.

If we drop our assumption that all weights must be unique, things get a little bit more complicated, but not that much. We will not go into these details here.

So, using this method, we can compute the expression within the brackets in time $O(n^2q)$ and in space $O(nq)$. The only problem that's left for the computation of the Deegan-Packel index is dividing by $|W_{\min}|$.

Again, we can use a straightforward dynamic programming approach. Consider again a weighted voting game $(W,q), |W| = n$ and its weight-equality classes $N_1,\ldots,N_m$. Let

$w,x \in \mathbb{N}$ and let $c(w,x)$ denote

$$\#\{S \subseteq W \mid \sum_{w_j \in S} w_j = w \wedge N_x \cap S \neq \varnothing \wedge \forall z > x : N_z \cap S = \varnothing\}.$$

Let w_{N_x} denote the weight of a player in weight-equality class N_x. Now $|W_{\min}|$ equals

$$\sum_{x=1}^{m} \sum_{w=q}^{q+w_{N_x}-1} c(w,x).$$

We will not go into detail on this any further, the approach is similar to what we have discussed already. We refer to [40] for details.

5.4 Generating Functions for Weighted Multiple Majority Games

Algaba et al. extend the method of generating functions for use in weighted m-multiple majority games [2]. They propose generating functions for computing the Banzhaf index and the Shapley-Shubik index. They use their resulting algorithms for computation of these power indices in the European Union (see the example in section 3.2.3).

The generating functions resemble a lot the generating functions for $A_i(k,j)$ and $b_i(k)$ that we have seen in the previous sections.

Proposition 71. *Let $\{(W_1 = (w_1^1, \ldots w_{|N|}^1), q_1), \ldots, (W_m = (w_1^m, \ldots w_{|N|}^m), q_m)\}$ be a weighted m-majority game on the set of players $N = \{1, \ldots n\}$. Let $b_i(k_1, \ldots, k_m)$ be the number of coalitions $S \subseteq N \backslash \{i\}$ such that $\sum_{j \in S} w_i^k = q^k$ for all $1 \leq k \leq m$.*

The banzhaf index β_i of player i is equal to

$$\sum_{k_1=q_1-w_i^1}^{q_1-1} \sum_{k_2=q_2-w_i^2}^{q_2-1} \cdots \sum_{k_m=q_m-w_i^m}^{q_m-1} b_i(k_1, \ldots, k_m).$$

The generating function for the numbers $b_i(k_1, \ldots, k_m)$ is

$$G_{b_i,m}(x_1, \ldots, x_m) = \prod_{j \neq i} 1 + x_1^{w_j^1} \cdots x_m^{w_j^m}$$

We will not proof this here. See [2] instead, but at least please compare this generating function with the generating function for b_i and notice how the functions seems like a natural extension. For the Shapley-Shubik index, a very similar extension can be made.

A dynamic programming algorithm can again be used to compute the coefficients of the generating functions. This works completely analogous to the dynamic programming algorithms that we already discussed. In the end, we end up with an algorithm of time complexity $O(nq_1 \ldots q_m)$ for the Banzhaf index, and an algorithm of time complexity $O(n^2 q_1 \cdots q_m)$ for the Shapley-Shubik index. Likewise, the space complexities will be $O(nq_1 \ldots q_m)$ and $O(n^2 q_1 \cdots q_m)$ respectively. Note that both the time and space complexities of these algorithms are exponential in m.

5.5 Another Improvement on the Generating Function Algorithm

Let us consider again the generating function algorithms for weighted voting games. If we use these algorithms to compute the Banzhaf index, Shapley-Shubik index and Deegan-Packel index for a single agent, then this costs us respectively $O(nq)$, $O(n^2q)$ and $O(n^2q)$ time. If we want to compute these indices for all players, then we can do this by simply running these algorithm n times. This results in runtimes of $O(n^2q)$, $O(n^3q)$ and $O(n^3q)$. Takeaki Uno presents in [56] a more efficient way to do this. He shows that it is possible to compute the indices of all players without exceeding the time complexity of the algorithms for single players.

We will explain this method for the Banzhaf index for a weighted voting game $(W = (w_1,\ldots w_n),q)$. The methods for the Shapley-Shubik and Deegan-Packel indices are more complex, we refer to [56] for those.

The first thing that we have to do in order to design this algorithm is to look at the generating function algorithm (the basic one, for a single player) in a different way. Let's look at the way that the values $b_n(k)$ (i.e. for the last player) from section 5.3.2 are computed. $b_n(k)$ equals $f(n,k)$ where

$$f(i,y) = |\{S \mid S \subseteq W_{\leq i}, w(S) = y)\}|.$$

In this equation $W_{\leq i}$ is the set of weights $\{w_1,\ldots,w_i\}$.

The dynamic programming that actually takes place during the computation of the values $f(n,k)$ is directly implied by the function

$$f(i,y) = \begin{cases} f(i-1,y) + f(i-1,y-w_i) & \text{if } y \geq w_i \\ f(i-1,y) & \text{if } y < w_i \end{cases}$$

If we compute the Banzhaf index for a player j other than the n'th player, then we basically do the same, except that we switch the weights of the j'th and the n'th player. This method is essentially the same as the generating function algorithm.

Next, we make the observation that if we compare the computations of the $f(n,k)$ for different players, then a lot of computations are the same. This means that there is room for improvement.

We can improve the algorithm by using dynamic programming "in both directions". The values $f(n,k)$ are computed by using the values of $f(n-1,\cdot)$. Consider now the function $b(n,k)$ that does exactly the opposite:

$$b(i,y) = |\{S \mid S \subseteq W_{\geq i}, w(S) = y)\}|.$$

$$b(i,y) = \begin{cases} b(i+1,y) + b(i+1,y-w_i) & \text{if } y \geq w_i \\ f(i+1,y) & \text{if } y < w_i \end{cases}.$$

We can compute in $O(nq)$ time the two-dimensional array that contains all the values $f(i,y)$ for $0 \leq i \leq n-1, 0 \leq y \leq q$. We can also compute in $O(nq)$ time the two-dimensional array that contains all the values $f(i,y)$ for $0 \leq i \leq n-1, 0 \leq y \leq q$.

The Banzhaf index β_i of the i'th player is now equal to

$$\begin{aligned}\beta_i &= \{(S_1,S_2) \mid S_1 \subseteq W_{\leq i-1} \wedge S_2 \subseteq W_{\geq i+1} \wedge q - w_i \leq w(S_1) + w(S_2) \leq q-1\} \\ &= \sum_{y=0}^{q-1} f(i-1,y) \left(\sum_{z=\max\{0,q-w_i-y\}}^{q-1-y} b(i+1,z) \right)\end{aligned}$$

Computing this sum can be done in $O(q)$ time, given that we already calculated all the values $f(\cdot,\cdot)$ and $b(\cdot,\cdot)$. Hence, computing all banzhaf indices takes $O(nq)$ time and $O(nq)$ space.

In [56], in addition several optimizations are given, one of which even reduces the space complexity to $O(n+q)$.

Chapter 6

Approximating Power Indices

Here we will discuss approximation algoritms for power indices, again mostly for the domain of weighted voting games. We start in section 6.1 by establishing facts about the hardness of approximating power indices in various settings. To do this, we also introduce some necessary theory of approximation algorithms. After that, in section 6.2 we describe the approximation algorithms that have been proposed in the literature.

6.1 Hardness of Approximation

According to the strict definition, *approximation algorithms* are algorithms for which it is guaranteed that the outcome is within a certain error bound from the optimal solution. Moreover, there must provably be a certain asymptotic bound on the runtime. For a good read on approximation algorithms, see [58]. The theory introduced in this section can also be found there.

Sometimes, researchers who say to have devised an approximation algorithm do not use this strict definition, and they simply propose an algorithm without giving any guarantee on the solution quality, or runtime. Strictly, such an algorithm should not be called an approximation algorithm; the term *'heuristic'* is sometimes used instead. The algorithms we give in the next section are not all approximation algorithms in the very strict sense, although in all cases there are actually interesting properties that can be stated about the runtime and solution quality.

For this section however, we will make use of the strictest definition of approximation algorithms, and we analyze the approximability of computing the two most popular indices: the Banzhaf index and the Shapley-Shubik index. In the literature, we have not found any explicitly stated results on approximability for these problems.

Most approximation algorithms for counting problems are in the form of *fully polynomial time (randomized) approximation schemes (FPTAS & FPRAS)*.

Definition 72 (FPTAS for counting)**.** Let $f : \Sigma^* \to \mathbb{N}$ be a counting function. Algorithm A is an *FPTAS* for f iff for all $0 < \varepsilon \leq 1$

$$\forall x \in \Sigma^* : |A(x) - f(x)| \leq \varepsilon f(x),$$

and A runs in time polynomial in both $|x|$ and $\frac{1}{\varepsilon}$.

Definition 73 (FPRAS for counting)**.** Let $f : \Sigma^* \to \mathbb{N}$ be a counting function. Algorithm A is an *FPRAS* for f iff for all $0 < \varepsilon \leq 1$

$$\forall x \in \Sigma^* : \mathbf{Pr}[|A(x) - f(x)| \leq \varepsilon f(x)] \geq p,$$

where $p \in \mathbb{Q}, \frac{1}{2} < p \leq 1$, and A runs in time polynomial in both $|x|$ and $\frac{1}{\varepsilon}$.

If we don't restrict the algorithms to run in time polynomial in $\frac{1}{\varepsilon}$, then the algorithms are called *PTAS* and *PRAS* respectively. The majority of approximation algorithms that have been found for counting problems are FPRAS's, although some FPTAS's are also known for some #P problems. An interesting fact about these approximation algorithms for #P problems, is that no FPRAS exists for counting problems that correspond to an NP-complete problem. We will now make this fact explicit (and even stronger) for the approximation of the Banzhaf and Shapley-Shubik indices.

From the above two definitions, we can extract a less strict notion for approximation algorithms for counting problems:

Definition 74 (Approximation algorithm for counting)**.** Let $f : \Sigma^* \to \mathbb{N}$ be a counting function. An algorithm A is an ε-*approximation algorithm for* f iff $0 < \varepsilon \leq 1$ and

$$\forall x \in \Sigma^* : |A(x) - f(x)| \leq \varepsilon f(x)$$

Notice that it makes no sense to allow values greater than 1 for ε, since this would allow for a very simple approximation algorithm for any counting problem: always output the number 0. Of course, in the above definition we are also allowed to choose for ε any function f that maps the input size to the range $(0,1]$. Using this definition, the following theorems show that the raw Banzhaf and raw Shapley-Shubik indices are strictly not approximable at all: not within any constant, any polynomial factor, any exponential factor, and even not within any other type of factor.

Theorem 75. *For any choice of f such that*

$$f : \mathbb{Z}^+ \to (0,1],$$

there exists no deterministic polynomial-time algorithm that f-approximates the raw Banzhaf index or the raw Shapley-Shubik index for weighted voting games under the extension that $\mathsf{P} \neq \mathsf{NP}$.

Proof. We proof this for the case of the raw Banzhaf index. The proof for the raw Shapley-Shubik index is analogous.

For the sake of contradiction, suppose that there exists a polynomial-time deterministic algorithm A that f-approximates the raw Banzhaf index for weighted voting games, where f is of course any function with domain Z^+ and target $(0,1]$. We can now decide the NP-complete problem PIVOT (see chapter 4) in polynomial time. Suppose we're given a PIVOT-instance $\langle W, q, i \rangle$, we simply run A on it. By definition, if the number of pivots is

zero, then $\langle W, q, i\rangle$ is a YES-instance, and the raw banzhaf index is 0. By the definition of an approximation algorithm for a counting problem, A will output the number 0. If the number of pivots is greater than 0, then $\langle W, q, i\rangle$ is a NO-instance, and the raw Banzhaf index is greater than 0. By the definition of an approximation algorithm for a counting problem, A will never output the number 0. □

Corollary 76. *Under* $\mathsf{P} \neq \mathsf{NP}$*, there exists no (F)PTAS for computing the raw Banzhaf index or the raw Shapley-Shubik index in weighted voting games.*

Our next theorem requires the definition of the two randomized complexity classes RP and BPP.

Definition 77 (BPP)**.** BPP is the class that contains all languages L for which there is a probabilistic polynomial-time Turing machine M such that

$$x \in L \Rightarrow \mathbf{Pr}(M(x) \text{ accepts }) \geq p,$$
$$x \notin L \Rightarrow \mathbf{Pr}(M(x) \text{ accepts }) \leq p,$$

where $p \in \mathbb{Q}$ and $\frac{1}{2} < p \leq 1$. The abbreviation BPP stands for *Bounded Probabilistic Polynomial* time.

Definition 78 (RP)**.** RP is the class that contains all languages L for which there is a probabilistic polynomial-time turing machine M such that

$$x \in L \Rightarrow \mathbf{Pr}(M(x) \text{ accepts }) \geq p,$$
$$x \notin L \Rightarrow \mathbf{Pr}(M(x) \text{ accepts }) \leq 0,$$

where $p \in \mathbb{Q}$ and $\frac{1}{2} \leq p \leq 1$. The abbreviation RP stands for *Randomized Polynomial* time.

Under the assumption that $\mathsf{NP} \neq \mathsf{RP}$, we can extend our last two theorems to randomized algorithms. This assumption is weaker than the assumption that $\mathsf{NP} \neq \mathsf{P}$, but nevertheless it is widely believed to be true.

Theorem 79. *For any choice of* f *such that*

$$f : \mathbb{Z}^+ \to (0, 1],$$

there exists no randomized polynomial algorithm that f*-approximates the raw Banzhaf index for weighted voting games with a probability* $p \in \mathbb{Q}$ *greater than* $\frac{1}{2}$*, under the extension that* $\mathsf{NP} \neq \mathsf{RP}$*. The same holds for the raw Shapley-Shubik index.*

Proof. Let β_i be the raw Banzhaf index of a player i in a weighted voting game. Let A be the randomized algorithm that outputs β_i' such that $\mathbf{Pr}[|\beta_i - \beta_i'| \leq f_k(n)\beta_i] \geq p, p \in \mathbb{Q}, \frac{1}{2} < p \leq 1$.

As in the previous two proofs, we obtain an algorithm A' that decides PIVOT. The difference is that this algorithm A' is not deterministic, but randomized, and it has an error-probability. A' decides PIVOT correctly with probability greater than p. The error-probability of this algorithm is two-sided. This means that for NO-instances, the probability that A' outputs NO is greater than p, and for YES-instances, the probability that A' outputs YES is greater than p. In other words, A' is a BPP-algorithm for PIVOT, hence we have established that $\mathsf{NP} \subseteq \mathsf{BPP}$. From exercise 11.5.18 of [46] it follows that if $\mathsf{NP} \subseteq \mathsf{BPP}$, then $\mathsf{NP} = \mathsf{RP}$. □

Corollary 80. *Under* NP $\neq$ RP*, there exists no (F)PRAS for computing the raw Banzhaf index or the raw Shapley-Shubik index.*

Luckily, these results only hold for the raw versions of the indices and only for the specific type of approximation algorithm that we defined above in definition 74. It is possible to define an alternative kind of approximation algorithm that seems quite natural.

Definition 81 (Additive error approximation algorithm for counting problems)**.** Let $f : \Sigma^* \to \mathbb{N}$ be a counting function. An algorithm A is an α*-additive error approximation algorithm for* f iff $\alpha : \mathbb{Z}^+ \to \mathbb{R}^+$ and

$$\forall x \in \Sigma^* : f(x) - \alpha(|x|) \leq A(x) \leq f(x) + \alpha(|x|)$$

Note that the name we use for this type of algorithm is not standard, but we will use it throughout this chapter.

We will see in the next section a randomized algorithm that computes the (non-raw) Banzhaf and (non-raw) Shapley-Shubik indices within any *fixed additive* error ε with high probability. Moreover, this algorithm runs in a time that's both polynomial in the input size and in $\frac{1}{\varepsilon}$. As a consequence, this means that for the raw Banzhaf index and raw Shapley-Shubik index there *does* exist for any $\varepsilon > 0$ respectively an *($\varepsilon 2^{n-1}$)-additive error randomized approximation algorithm* and an *($\varepsilon n!$)-additive error randomized approximation algorithm* that runs in time polynomial in the input size and in $\frac{1}{\varepsilon}$.

This poses the question for which functions f, the raw Banzhaf and raw Shapley-Shubik indices are f-additive error inapproximable. The answer is given in the following theorem.

Theorem 82. *If* P $\neq$ NP*, then for any polynomial function $f : \mathbb{N} \to \mathbb{N}$ of, there is no f-additive error approximation algorithm for the raw Banzhaf index and the raw Shapley-Shubik index.*

Proof. Here is the proof for the case of the raw Banzhaf index. The problem of computing the raw Banzhaf index is #P-parsimonious-complete, so automatically the same holds for the raw Shapley-Shubik index.

We will show how to decide PIVOT in polynomial time, if there would be such an f-additive error approximation algorithm.

Assume w.l.o.g. that there is an a such that $\forall n \geq a : f(n) < n^c$ for some $c \geq 1$. Suppose algorithm A is such an approximation algorithm and suppose we have the instance $\langle W, q, i \rangle$ of PIVOT. Assume w.l.o.g. that $|W|$ is greater than a and is also great enough such that $2^{|W|^c} > 4|W|^{2c}$. We now add $|W|^c$ dummy players to W, so that we obtain the game (W', q). We run algorithm A on $\langle W', q, i \rangle$.

If $\langle W, q, i \rangle \in$ PIVOT, then the number of swing coalitions for player i in (W, q) is 0, so the number of swing coalitions for player i in (W', q) is $0 \cdot 2^{|W|^c} = 0$. Algorithm A will in this case output a number between 0 and $(|W| + |W|^c)^c \leq 2|W|^{2c}$. If $\langle W, q, i \rangle \notin$ PIVOT, then the number of swing coalitions for player i in (W, q) is at least 1, so the number of swing coalitions for player i in (W', q) is at least $1 \cdot 2^{|W|^c} = 2^{|W|^c}$. Algorithm A will in this case output a number that is at least $2^{|W|^c} - (|W| + |W|^c)^c \geq 2^{|W|^c} - 2|W|^{2c} > 4|W|^{2c} - 2|W|^{2c} = 2|W|^{2c}$.

So if $\langle W, q, i\rangle \in$ PIVOT, then A will output a number lower than or equal to $2|W|^{2c}$. And if $\langle W, q, i\rangle \notin$ PIVOT, then A will output a number greater than $2|W|^{2c}$. □

Theorem 83. *If* RP $\neq$ NP*, then for any polynomial function* $f : \mathbb{N} \to \mathbb{N}$ *of, there is no randomized algorithm that* f*-additive error approximates the raw Banzhaf index and the raw Shapley-Shubik index of a weighted voting game with probability* $p \in \mathbb{Q}, p > \frac{1}{2}$.

Proof. By the same arguments as in the proof for theorem 79. □

In section 6.2.3, we give two more inapproximability results (as presented in [8]) about the amount of information that needs to be extracted from the characteristic function of a game, in order to reach a specific desired approximation guarantee.

6.2 Approximation Algorithms

In this section, we will cover three algorithms that approximate the Shapley-Shubik index and/or Banzhaf index. These algorithms have in common that they all use statistical theory. Just as in the last chapter, these three algorithms do not form an exhaustive list. We picked the algorithms based on the algorithmic ideas used, and on the domain of weighted voting games on which they are applicable. The algorithms we omit are the following.

- In [40], a Monte-Carlo algorithm is given for the Shapley-Shubik and Banzhaf indices in weighted voting games where the players are ordered descendingly according to their weights. The maximum-likelihood estimator is deduced, and no confidence interval is given (as a function of the number of samples).
- In [38], another Monte-Carlo approach is suggested. The algorithm we discuss in 6.2.3 is similar, and has been analyzed more rigorously.
- The apprimation algorithm for weighted voting games due to Dennis Leech, proposed in [1] and [36]. This algorithm is a combination of the multi-linear extension method that we explain in section 6.2.1, and the direct enumeration method of section 5.1. Basically, on the agents with heavier weights, the direct enumeration method is used, and on the agents with lighter weights, the multi-linear extension method is used.

We start with discussing a classic algorithm due to Owen ([45] [43], [44]), based on the multilinear extension of a game. After that, we move on to a more advanced algorithm by Fatima et al. ([24], [26]) that's suited for weighted m-multiple majority games. Finally, we will give an approximation algorithm by Bachrach et al. ([8]) that can be used for all voting games, even non-monotone ones.

6.2.1 Multi-linear extensions

Owen's method of multi-linear extensions [43], [44] is based on two concepts. The first one is the use of the central limit theorem, which states that the averaged sum of a large number of identically distributed variables will approximately be a normal distribution. The second one is writing the Shapley-Shubik function as an Euler Beta function.

The Euler Beta function B can be written as

$$B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}\,dt,$$

but also as

$$B(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)},$$

where Γ is the Gamma function (the extension of the factorial function to reals, $\Gamma(n) = (n-1)!$).

Hence, if we let (W,q) be a simple game and if we let $S_i \in 2^W$ be the set of swing coalitions for player i in that game, then the Shapley value φ_i of player i is given by

$$\varphi_i = \sum_{S\in S_i} \frac{|S|!(|W|-|S|-1)}{|W|!} = \sum_{S\in S_i} B(|S|+1,|W|-|S|) = \sum_{S\in S_i} \int_0^1 x^{|S|}(1-x)^{|W|-|S|-1}\,dx.$$

If we now look at the integrand alone in the above equality, then if we fix x between 0 and 1, we can regard this integrand as the probability that coalition S will form, if every player joins S with probability x.

So if we fix x and we simply sum the integrand over all coalitions of S_i, then we obtain the probability $\text{swing}_i(x)$ of a swing for i. So

$$\text{swing}_i(x) = \sum_{S\in S_i} x^{|S|}(1-x)^{|W|-|S|-1}.$$

This expression is called the *multilinear extension of* (W,q) The Banzhaf index is then equal to $\text{swing}_i(\frac{1}{2})$. Moreover, the Shapley-Shubik index can be written as

$$\int_0^1 \text{swing}_i(x).$$

We can approximate $\text{swing}_i(x)$ as follows. Suppose player j votes with probability x in the same way that i does, we can define the stochastic variables $v_{j,x}$ for each $i \neq j$ as

$$\mathbf{Pr}[v_{j,x} = w_j] = x, \mathbf{Pr}[v_{j,x} \neq w_j] = 1-x$$

Then the total weight that is on the same side as the weight of player i is $v_{i,x} = \sum_{j\neq i} v_{j,x}$. Now we have

$$\text{swing}_i(x) = \mathbf{Pr}[v_{i,x} < q].$$

$\mu_{i,x} = x\sum_{w\in W\setminus\{i\}} w$ is the mean of $v_{i,x}$ and $\sigma_{i,x}^2 = x(1-x)\sum_{w\in W\setminus\{i\}} w^2$ the variance of $v_{i,x}$. By the central limit theorem, we can approximate $\text{swing}_i(x)$ as follows

$$\text{cost}_i(x) = \mathbf{Pr}[v_i < q] = \phi(\frac{q-\mu_{i,x}}{\sigma_{i,x}}) - \phi(\frac{q-w_i-\mu_{i,x}}{\sigma_{i,x}}),$$

where ϕ denotes the standard cumulative normal distribution function. Altogether, this approximation method runs in linear time, but in general this method only works well on voting games with a large number of small weighted agents. If the majority of the weights is only in the hands of a few, the central limit theorem fails [36].

6.2.2 Approximating the Shapley-Shubik index

Fatima et al. present in [24] a method to approximate the Shapley-Shubik index of a weighted voting game. This method runs in time linear in the number of agents. Later, they extend this method to also work for weighted m-multiple majority games [26]. This method runs in time $O(m^2 n)$, with n being the number of agents. We will explain the method for weighted voting games only. This is for the reason that the theory of statistics involved for the extension to m-multiple majority games, is too complex to discuss here.

The method is based on the following theorem, cited from [26]:

Theorem 84. *If $w_1, \ldots w_X$ is a sample of size X drawn from 'any distribution' with mean μ and variance σ^2, then the sample mean (i.e. $\frac{1}{X}\sum_{i=1}^{X} w_i$) has an approximate normal distribution, $\mathcal{N}$, with mean μ and variance $\frac{\sigma^2}{X}$.*

Proof. See [27]. □

We use this distribution from theorem 84 to compute a player i's expected contribution in a coalition of size $X+1$. Let the weighted voting game be $(W = (w_1, \ldots, w_n), q)$. Let μ be the mean of the weights and let σ^2 be the variance of the weights. The weight of a player in a coalition of size X that doesn't contain player i has the approximate normal distribution $\mathcal{N}(\mu, \frac{\sigma^2}{X})$. This means that the size of the area under the curve $\mathcal{N}(\mu, \frac{\sigma^2}{X})$ between $\frac{q-w_i}{X}$ and $\frac{q}{X}$ is the approximate probability that the total weight of a coalition of size X lies between $q - w_i$ and q. Hence, the approximate probability $E\Delta_i^X$ that player i can turn a coalition of size X into a winning coalition is the size of the aforementioned area and is given by

$$E\Delta_i^X = \frac{1}{\sqrt{(2\pi\sigma^2/X)}} \int_{q-w_i}^{q} e^{-X\frac{(x-\mu)^2}{2v}}. \tag{6.1}$$

Now, by taking the average over all coalition sizes smaller than n, we get the approximate Shapley-Shubik index $\bar{\varphi}_i$ of player i:

$$\bar{\varphi}_i = \frac{1}{n}\sum_{X=0}^{n-1} E\Delta_i^X.$$

In [26], the approximation error of this method is analysed. The analysis is too extensive to give here, but it turns out that this method yields a $O(\frac{1}{\sqrt{n}})$-additive error approximation algorithm. The extension of this method to weighted m-majority games yields a $O(\frac{k^2}{\sqrt{n}})$-additive error approximation algorithm.

6.2.3 A general approximation method for simple games

Bachrach et al. give in [8] an approximation algorithm for the Shapley-Shubik and Banzhaf indices that works for any type of simple coalitional game; even the general, non-monotonic class of simple coalitional games.

Besides the fact that this algorithm works on any simple game, another difference from the two algorithms we discussed above is that this algorithm is randomized. It works by randomly sampling coalitions.

Suppose the input of the algorithm is a game (A, v) and a number $i, 1 \leq i \leq |A|$. The algorithm computes the Banzhaf index for player i. First we sample uniformly at random a set of k coalitions $C_1, \ldots, C_k$ containing player i such that every player $j \neq i$ has a probability of $\frac{1}{2}$ of occuring in a coalition.

Next, compute the number c of these coalitions in which i is a swing player. The unbiased maximum likelihood estimator for the Banzhaf index β_i for player i is

$$\bar{\beta}_i = \frac{c}{k}.$$

This does not give us any confidence interval, but fortunately we can establish one through Hoeffding's inequality (cited from [8]):

Theorem 85. *Let $X_1, \ldots, X_n$ be independent random variables, where all X_i are bounded so that $X_i \in [a_i, b_i]$, and let $X = \sum_{i=1}^{n} X_i$. Then the following inequality holds.*

$$\mathbf{Pr}[|X - E[X]| \geq n\varepsilon] \leq 2exp\left(-\frac{2n^2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right)$$

c can be regarded as the sum of k independent Bernouilli variables, so all of these variables are bounded between 0 and 1. Moreover, the expected value of c is equal to $k\beta_i$. Therefore we have

$$\mathbf{Pr}[|c - k\beta_i| \geq k\varepsilon] \leq 2e^{-2k\varepsilon^2},$$

and it follows that

$$\mathbf{Pr}[|\bar{\beta}_i - \beta_i| \geq \varepsilon] \leq 2e^{-2k\varepsilon^2}.$$

Using this equation, it is fairly straightforward to deduce the number of samples k that we have to make in order to compute $\bar{\beta}_i$ with a given accuracy ε and a given confidence level $1 - \delta$. Such a convidence interval is guaranteed when $k = \frac{\ln \frac{2}{\delta}}{2\varepsilon^2}$.

So computing in this way the Banzhaf index that is probably approximately correct in the interval $[\bar{\beta}_i - \varepsilon, \bar{\beta}_i + \varepsilon]$ with probability $1 - \delta$, requires us to take $O(\ln \frac{1}{\delta} \cdot \frac{1}{\varepsilon^2})$ samples, and taking a sample requires $O(|A|)$ time, therefore the total runtime is $O(\ln \frac{1}{\delta} \cdot |A| \cdot \frac{1}{\varepsilon^2})$. So as you can see, the runtime and approximation guarantee of this algorithm is indipendent from the length of the encoding of the characteristic function v.

This algorithm is easily adapted to work for the Shapley-Shubik index instead of the Banzhaf index. The only adaptation that needs to be made is in the random sampling of coalitions: instead of sampling coalitions, now we should sample permutations.

Lastly, in [8], some interesting approximability results are given. They are remarkable because they do not rely on any complexity-theoretic assumtions such as $\mathsf{P} \neq \mathsf{NP}$. Their

proofs are not that short, so we won't give the proofs here. However, we will give the actual results:

Theorem 86. *Given a player and a simple coalitional game v:*

- *A deterministic algorithm that has query access to v and gets as input the number of players and i, that computes the Banzhaf index for player i with accuracy* $O(\frac{1}{\sqrt{n}})$ *must query v* $\Omega(\frac{2^n}{\sqrt{n}})$ *times.*
- *There exists no randomized algorithm with query access to v that constructs a confidence interval for the Banzhaf index (given a player and a simple coalitional game v) with a confidence level of* $1-\delta, \delta < \frac{1}{2}$ *and an accuracy of* $o(\frac{1}{2^n})$*, querying v only polynomially many times.*

Chapter 7

Future Work

Concerning the research of power indices in computer science, there are several interesting directions for future work.

Other Power Indices A lot of research has been put into algorithms for computing the Banzhaf and Shapley-Shubik indices in weighted voting games, but as you might have guessed, other power indices such as the Deegan-Packel index and even more so the Holler index and the Johnston index received little attention. Efficient exact algorithms[1] and approximation algorithms for computing these power indices still need to be designed.

Other Types of Games Secondly, there is the problem of computing the power indices in simple games other than weighted voting games. Of course, the algorithm of section 6.2.3 is suitable for all types of coalitional games, but specialized algorithms for these games might run faster or approximate better. For example, algorithms for computing power indices in coalitional games that are represented by their minimal winning coalitions remain unknown. It could be that for this representation faster algorithms exist than for the case of games in weighted voting game representation, because a minimal winning coalition representation of a weighted voting game can be exponentially much larger than the corresponding weighted voting game representation (although theorem 57 is some evidence that such algorithms might not be *that* much faster).

As we briefly mentioned in the introduction, Bachrach and Rosenschein already investigated the problem for a specific different type of coalitional game. In [9] and [10], they give some algorithms for computing the Banzhaf index in specific coalitional classes of games on graphs. This naturally gives rise to the idea of using power indices as a network reliability metric. Therefore it would be interesting to investigate the problem of computing power indices in coalitional graph games some more.

Manipulation Recently, in [7] Bachrach and Elkind started studying the effect on a player's power index in a weighted voting game when the player splits his total weight into two

[1] An exception is probably the Deegan-Packel index, see [40] and [56] and chapter 5 of this survey. However, no good approximation algorithms for this power index are known to us.

weights (adding an extra, false-name player to the game). The power index of the player can than be regarded as the sum of the power indices of the 'splitted' new two players. In in [7], this situation is analyzed for the Shapley-Shubik index. The authors show that by these manipulations, in the worst case it is possible for a player to double his Shapley-Shubik index.

In [5], among other topics the same question is analyzed for the case of the Banzhaf index.

A related topic is discussed in [60]. In that paper, the main question is how the central authority can change the quota in order to manipulate a player's power index.

Finally, a paper that is of importance when it comes to manipulation of weighted voting games is [23], where the computational complexity of power index comparison is studied.

As of writing this, the study on the computational aspects of manipulation of weighted voting games is fairly new. In the papers we just mentioned, many interesting open problems are presented.

The Inverse Problem Lastly, there is a problem that is known as "the inverse problem". For this problem, we are interested in constructing a weighted voting game, given the power indices for the players. Not that much is known about this problem. We know of three notable papers where algorithms are presented for this problem; all of them are very recent:

- A notable result in the field of threshold logic is a PRAS[2] for the problem of computing a boolean threshold function that is close to given target Chow parameters [42]. For this problem, some more heuristics have been proposed in the past. We refer the interested reader to [42] for references and a discussion on this.
- Concerning the inverse problem for the Shapley-Shubik index, Fatima et al. present an anytime approximation algorithm [25]. They show that in each iteration, the approxmation error decreases. Also they prove that the runtime of a single iteration is $O(n^2)$, and they present an experimental analysis of the algorithm.
- In [6], Aziz et al. present a method for the inverse problem for the case of the Banzhaf index.

All are approximation algorithms. Complexity theoretical questions regarding this problem remain completely open, and with exception of the Chow parameters algorithm, there exists no algorithm for the inverse problem for which a worst case analysis or an approximation guarantee is given.

[2]Note that this is not a PRAS in the sense of definition 73, but a PRAS in the 'additive error' sense of definition 81.

Bibliography

[1] Computing power indices for large voting games. *Management Science*, 49(6):831–838, June 2003.

[2] E. Algaba, J. M. Bilbao, J. R. Fernández García, and J. J. López. Computing power indices in weighted multiple majority games. *Mathematical Social Sciences*, 46:63–80, 2003.

[3] Pajala Antti. Voting power and power index website: a voting power WWW-resource including powerslave voting body analyser. WWW, april 2002. University of Turku. Turku. URL: http://powerslave.val.utu.fi/.

[4] Haris Aziz. Complexity of comparison of influence of players in simple games. In *Proceedings of the 2nd International Workshop on Computational Social Choice (COMSOC-2008)*, pages 61–72, 2008.

[5] Haris Aziz and Mike Paterson. Complexity of some aspects of control and manipulation in weighted voting games. *Annales du Lamsade*, 9, 2008.

[6] Haris Aziz, Mike Paterson, and Dennis Leech. Efficient algorithm for designing weighted voting games. In *Proceedings of the IEEE Computer Society, 11th IEEE International Multitopic Conference*, 2007.

[7] Yoram Bachrach and Edith Elkind. Divide and conquer: false-name manipulations in weighted voting games. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 975–982, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[8] Yoram Bachrach, Vangelis Markakis, Ariel D. Procaccia, Jeffrey S. Rosenschein, and Amin Saberi. Approximating power indices. In *Proceedings of the The Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 943–950, Estoril, Portual, May 2008.

[9] Yoram Bachrach and Jeffrey S. Rosenschein. Computing the Banzhaf power index in network flow games. In *AAMAS '07: Proceedings of the 6th international joint*

conference on Autonomous agents and multiagent systems, pages 1–7, New York, NY, USA, 2007. ACM.

[10] Yoram Bachrach and Jeffrey S. Rosenschein. Power and stability in connectivity games. In *The Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Estoril, Portual, May 2008.

[11] J. Bilbao, J. Fernández, A. Losada, and J. López. Generating functions for computing power indices efficiently. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 8(2):191–213, December 2000.

[12] S.F. Brams and P.J. Affuso. Power and size: a new paradox. *Theory and Decision*, 7:29–56, 1976.

[13] C.K. Chow. On the characterization of threshold functions. In *Proceedings of the Symposium on Switching Circuit Theory and Logical Design (FOCS)*, pages 34–38, 1961.

[14] James S. Coleman. Control of collectives and the power of a collectivity to act. *Lieberman, Bernhardt, Social Choice*, pages 192–225, 1971.

[15] James S. Coleman. Control of collectivities and the power of a collectivity to act. *Social Choice*, pages 113–123, 1971.

[16] Vincent Conitzer and Tuomas Sandholm. Computing Shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In *Proceedings of the National Conference on Artificial Intelligence*, pages 219–225, San Jose, California, 2004. American Association for Artificial Intelligence.

[17] M. Davis and M. Maschler. The kernel of a cooperative game. *Naval Research Logistics Quarterly*, 12:223–259, 1965.

[18] Xiaotie Deng and Christos H. Papadimitriou. On the complexity of cooperative solution concepts. *Math. Oper. Res.*, 19(2):257–266, 1994.

[19] Pradeep Dubey. On the uniqueness of the Shapley value. *International Journal of Game Theory*, 4(3):131–139, September 1975.

[20] Pradeep Dubey and Lloyd S. Shapley. Mathematical properties of the Banzhaf power index. *Mathematics of Operations Research*, 4(2):99–131, 1979.

[21] Arnaud Durand, Miki Hermann, and Phokion G. Kolaitis. Subtractive reductions and complete problems for counting complexity classes. *Theoretical Computer Science*, 340(3):496–513, 2005.

[22] E. Einy. The desirability relation of simple games. *Mathematical Social Sciences*.

[23] P. Faliszewski and L. Hemaspaandra. The complexity of power-index comparison. In *In Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management*, pages 177–187. Springer-Verlag Lecture Notes in Computer Science, june 2008.

[24] Shaheen S. Fatima, Michael Wooldridge, and Nicholas R. Jennings. A randomized method for the Shapley value for the voting game. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007)*, pages 955–962, Honolulu, Hawaii, May 2007.

[25] Shaheen S. Fatima, Michael Wooldridge, and Nicholas R. Jennings. 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 2008)*, pages 935–942, Estoril, Portugal, May 2008.

[26] Shaheen S. Fatima, Michael Wooldridge, and Nicholas R. Jennings. A linear approximation method for the Shapley value. *Artificial Intelligence*, 172(14):1673–1699, 2008.

[27] A. Francis. *Advanced Level Statistics*, page 425. 1979, Stanley Thornes Publishers.

[28] D.B. Gillies. Solutions to general non-zero-sum games. In A. W. Tucker and R. D. Luce, editors, *Contributions to the Theory of Games IV*, pages 47–85. Princeton University Press, 1959. Annals of Mathematics Studies 40.

[29] M.J. Holler. Forming coalitions and measuring voting power. *Political studies*, 30:262–271, 1982.

[30] Samuel Ieong and Yoav Shoham. Marginal contribution nets: a compact representation scheme for coalitional games. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, pages 193–202, New York, NY, USA, 2005. ACM.

[31] John F. Banzhaf III. Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review*, 19(2):317–343, winter 1965.

[32] R. J. Johnston. On the measurement of power: Some reactions to laver. *Environment and Planning*, 10:907–914, 1978.

[33] John Deegan Jr. and Edward W. Packel. A new index of power for simple n-person games. *International Journal of Game Theory*.

[34] Bettina Klinz and Gerhard J. Woeginger. Faster algorithms for computing power indices in weighted voting games. *Mathematical Social Sciences*, 49:111–116, 2005.

[35] M. W. Krentel. The complexity of optimization problems. In *STOC '86: Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 69–76, New York, NY, USA, 1986. ACM.

[36] Dennis Leech. Computation of power indices. Technical Report 664, Warwick Economic Research Papers, July 2002.

[37] W. F. Lucas. *Measuring Power in Weighted Voting Systems*, pages 183–238. Springer-Verlag, New York, 1975.

[38] I. Mann and Lloyd S. Shapley. Values of large games, IV: Evaluating the electoral college by monte carlo techniques. Technical Report RM-2651, The RAND Corporation, Santa Monica, CA, 1960.

[39] I. Mann and Lloyd S. Shapley. Values of large games, VI: Evaluating the electoral college exactly. Technical Report RM-3158-PR, The RAND Corporation, 1962.

[40] Yasuko Matsui and Tomomi Matsui. A survey of algorithms for calculating power indices of weighted majority games. *J. Oper. Res. Soc. Japan*, 43:71–86, 2000.

[41] Yasuko Matsui and Tomomi Matsui. NP-completeness for calculating power indices of weighted majority games. *Theoretical Computer Science*, 263(1–2):305–310, 2001.

[42] Ryan O'Donnell and Rocco A. Servedio. The chow parameters problem. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 517–526, New York, NY, USA, 2008. ACM.

[43] Guillermo Owen. Multilinear extensions of games. *Management Science*, 18:64–79, 1972.

[44] Guillermo Owen. Multilinear extensions and the Banzhaf value. *Naval Research Logistic Quarterly*, 22:741–750, 1975.

[45] Guillermo Owen. *Game Theory*. Academic Press, 1995.

[46] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley Longman, 1994.

[47] Bezalel Peleg and Peter Sudhlter. *Introduction to the Theory of Cooperative Games*. Springer, 2003.

[48] L. S. Penrose. The elementary statistics of majority voting. *Journal of the Royal Statistical Society*, 109:53–57, 1946.

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

[50] William H. Riker. *The Theory of Political Coalitions*. Greenwood Press, 1962.

[51] D. Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal of Applied Mathematics*.

[52] Lloyd S. Shapley. A value for n-person games. *Annals of Mathematics Study*, 28:307–317, 1953.

[53] Lloyd S. Shapley and Martin Shubik. A method of evaluating the the distribution of power in a committee system. *American Political Science Review*, 48(3):787–792, 1954.

[54] Janos Simon. On some central problems in computational complexity. Technical report, Ithaca, NY, USA, 1975.

[55] Alan D. Taylor and William S. Zwicker. *Simple Games: Desirability Relations, Trading, Pseudoweightings*. Princeton University Press, 1999.

[56] Takeaki Uno. Efficient computation of power indices for weighted majority games. Technical Report NII-2003-006E, National Institute of Informatics, 2003.

[57] Leslie Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[58] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2003.

[59] V. Zankó. #P-completeness via many-one reductions. *International Journal of Foundations of Computer Science*, 2(1):76–82, 1991.

[60] Michael Zuckerman, Piotr Faliszewski, Yoram Bachrach, and Edith Elkind. Manipulating the quota in weighted voting games. In *The Twenty-Third National Conference on Artificial Intelligence*, Chicago, Illinois, July 2008.

Appendix A

New Results

In this appendix, I give some theorems that I didn't find in the literature. For now, consider this as nothing more than a scrapbook. In the final version, this appendix will be removed. The first two theorems of this appendix have been included in chapter 4.

Theorem 87. *The languages PIVOT-iTH-HEAVIEST are in* P *for all* $i \in \mathbb{N}$. *They are defined as:*

$$\{\langle W = (w_1, \ldots, w_n), q\rangle \mid \beta'_i > 0 \text{ in the weighted voting game } (W,q) \wedge \forall i, j : i < j \rightarrow w_i \geq w_j\}.$$

Here, β'_i *is the number of coalitions that* i *is critical in a.k.a. the raw Banzhaf index for player* i.

Proof. Let's first consider the language PIVOT-1TH-HEAVIEST. Clearly, in a weighted voting game $(W = (w_1, \ldots, w_n), q)$ where $w_1 \geq w_2 \geq \cdots \geq w_n$, clearly w_1 there is always a coalition in which w_1 is critical as long as $0 < q \leq \sum_{w \in W} w$. So there is an $O(n)$-time algorithm for PIVOT-1TH-HEAVIEST. Actually, it is even possible to find a coalition for which player i is critical: consider the grand coalition and keep removing the lowest-weighted players until the sum of the weights of the players in the coalition drops below q. If that happens, add the last player w_j that was removed. The resulting coalition is a swing coalition for player 1, because $w_1 \geq w_j$.

For the problems PIVOT-iTH-HEAVIEST, with $i > 1$ we can generalize the idea: In a weighted voting game $(W = (w_1, \ldots, w_n), q)$ where $w_1 \geq w_2 \geq \cdots \geq w_n$, player i has a pivot coalition if and only if at least one of the following coalitions is a swing coalition for i :

$$C = \{s_{<i} \cup s_{\geq i} \mid s_{<i} \in S_{<i} \wedge s_{\geq i} \in S_{\geq i}\},$$

where $S_{<i}$ is the collection of subsets $s \subseteq W$ that contain only weights heavier than w_i, and $S_{\geq i}$ consists of the subsets s of W that contain w_i, and if $w_j \in s$ for any $j > i$, then $w_{j-1} \in S_{\leq i}$.

This is true because we can transform any arbitrary swing coalition for i into a swing coalition for i in C: Consider a swing coalition D for i that is not in C. Remove all the weights lower than w_i from D to obtain D'. If the resulting coalition is a swing coalition, then we're done. Otherwise, we have $\sum_{d \in D'} d < q - w_i$, so we iteratively add the highest weight

lower than w_i that's not in D'. Clearly at some iteration we will have $q - w_i \leq \sum_{d \in D'} < q$. At that point, D' is in C.

We can enumerate all of the coalitions in C, and check whether each coalition is a swing coalition for i. The cardinality of C is $2^{i-1}(n-i)$, so we can decide PIVOT-iTH-HEAVIEST in linear time. □

Theorem 88. *The languages PIVOT-iTH-LIGHTEST are* NP*-complete for all $i \in \mathbb{N}$. They are defined as:*

$$\{\langle W = (w_1, \ldots, w_n), q\rangle \mid \beta'_i > 0 \text{ in the weighted voting game } (W,q) \wedge \forall i, j : i < j \rightarrow w_i \leq w_j\}.$$

Here, β'_i is the number of swings a.k.a. the raw Banzhaf index for player i.

Proof. For $i = 1$, it has already been proved in [41], by a polynomial-time reduction (a.k.a. Karp reduction) from PARTITION. It is very easy to extend it to reductions for the cases that $i > 1$. We give a family of Karp reductions from PIVOT-1TH-LIGHTEST to PIVOT-iTH-LIGHTEST for any i. Given an instance $\langle W, q\rangle$ of PIVOT-1TH-LIGHTEST, append $i-1$ weights with value $\frac{1}{i-2}$ to the beginning of W to get W'. Because (W,q) is a weighted voting game, q and all weights in W are integer. The newly added weights are not integer, but if we would allow the weights in weighted voting games to be rational numbers, then obviously the new players in (W',q) would all be dummy players. We eliminate the rationals from (W',q) by multiplying all weights by $i-2$ and multiplying q by $i-2$. In the resulting game we still have that the first $i-1$ players are dummy players. So if $\beta'_1 > 0$ in the PIVOT-1TH-LIGHTEST-instance, then $\beta'_i > 0$ in the PIVOT-iTH-LIGHTEST-instance, and vice versa. Therefore, all of the problems PIVOT-iTH-LIGHTEST are NP-complete. □

Theorem 89. *Two players in a weighted majority game are symmetric iff their (raw) Banzhaf indices are equal and their (raw) Shapley-Shubik indices are equal. Recall: two players i and j are considered symmetric iff*

$$\begin{aligned} \forall S \subseteq W \backslash \{w_i, w_j\} : \qquad & q - w_i \leq \left(\sum_{w \in S} w\right) < q \\ \leftrightarrow \quad & q - w_j \leq \left(\sum_{w \in S} w\right) < q, \end{aligned}$$

Proof. First of all, it's obvious that equalness of the Banzhaf indices is equivalent to equalness of the Shapley-Shubik indices.

The forward proof for this theorem is trivial.

As for the backward case, let $(W = (w_1, \ldots, w_n), q)$ be an arbitrary weighted voting game. Assume without loss of generality that $\forall i, j : i < j \rightarrow w_i \geq w_j$. Suppose that for two different players i and j, $i < j$, the Banzhaf indices are equal. Firstly, there cannot exist a coalition $S \subseteq W \backslash \{w_i, w_j\}$ such that $w_j \cup S$ is a pivot coalition for j, and $w_i \cup S$ is not a pivot coalition for i. This is because $w_i \geq w_j$. Secondly, suppose there is a coalition $S \subseteq W \backslash \{w_i, w_j\}$ such that $w_i \cup S$ is a pivot coalition for i, and $w_j \cup S$ is not a pivot coalition for j. If this is so, then by equalness of the Banzhaf indices of both players, there must also

be a coalition $S' \subseteq W \backslash \{w_i, w_j\}$ such that $w_j \cup S'$ is a pivot coalition for j, and $w_i \cup S'$ is not a pivot coalition for i. So we fall back to the first case, which is impossible as we showed.

Actually, by now I have read a bit more literature and I realize that there are much easier ways to prove this. □

Theorem 90. *Let SWINGTOTAL-WVG-EXISTENCE be the following language:*

$$\{\langle \beta, n\rangle \quad | \quad \exists G = (W = (w_1, \ldots, w_n), q) : \\ \beta \textit{ is the total number of swings in weighted voting game } G\}.$$

SWINGTOTAL-WVG-EXISTENCE is in EXPSPACE.

Proof. We use a result in [4] that states that it can be decided in polynomial time whether a game in minimal winning coalition form can be represented as a weighted voting game. We can enumerate all games in minimal winning coalition form on n players, compute in exponential time the swing total of this game, and decide in polynomial time whether this game can be represented as a weighted voting game.

The number of games that we have to enumerate is absolutely huge. For a game of n players, this number is equal to the number of antichains on a set that has cardinality n. Equivalently, this number is equal to the number of monotone boolean functions (is obvious). No explicit mathematical expression has been found for these numbers yet; actually this is quite a big open problem in the thoery of set systems and monotone boolean function. The problem is known as Dedekind's problem, and the numbers are known as Dedekind numbers.

However, when we incorporate the following theorem, known as Sperner's theorem, it is easy to determine a crude upper bound on the amount of antichains.

Theorem 91 (Sperner's theorem)**.** *The maximum size of an antichain on a set of cardinality n is $\binom{n}{\lfloor n/2 \rfloor}$.*

Clearly an upperbound on the number of antichains of size k on a set of cardinality n is 2^{n^k} So an upperbound on the total number of antichains is

$$\sum_{k=0}^{\binom{n}{\lfloor n/2 \rfloor}} 2^{n^k} \leq 2^{n^{\binom{n}{\lfloor n/2 \rfloor}}} 2^n.$$

I guess much better upper bounds are known (I didn't study any literature on this) but Dedekind numbers are huge either way.

An algorithm that decides this problem could work as follows: first it enumerates the antichains of size 1, then the antichains of size 2, and so on until we reach size $\binom{n}{\lfloor n/2 \rfloor}$. Each antichain represents a simple monotone game. For this game we count the total number of swings in exponential time, and if it has the correct number of swings, then we decide in polynomial time if it is representable as a weighted voting game.

Of course, this is just a quick result and possibly it can be sharpened. □

Theorem 92. *Let INVERSE-RAWBANZHAF-WVG-EXISTENCE be the following language:*

$$\{\langle \beta_1, \ldots, \beta_n \rangle \quad | \quad \exists G = (W = (w_1, \ldots, w_n), q) : \\ \beta'_i \text{ is the raw Banzhaf index of player } i \text{ in weighted voting game } G\}.$$

INVERSE-RAWBANZHAF-WVG-EXISTENCE is in EXPSPACE.

Proof. Naively we can try to enumerate all vectors of $n+1$ integers, and check if the simple game denoted by that vector has the required power index. The problem with this is that we don't know when to stop. If the game exists however, then eventually we will find one. So from this we conclude that the problem is at least recursively enumberable.

To see that it's decidable we can use the same method as in the previous theorem.

From my inuition, by now it's starting to look like the inverse power index problem really is insanely hard :) □

Theorem 93. *There are minimally $|N|$ swings in a weighted majority game. This bound is tight. (Also, the upper bound is $\lceil \frac{n}{2} \rceil \binom{n}{\lceil \frac{n}{2} \rceil}$. See [20] for that.)*

Proof. Any weighted majority game has a minimal winning coalition. Suppose a coalition $S \subseteq N$ is a minimal winning coalition. For this coalition there are $|S| \geq 1$ swings. Partition the players not in S into L, H, and B. L are the players lighter than the lightest player in S. H are the players heavier than the heaviest player in S, and B are the players in between the lightest and heaviest player in S.

If we add a player in L to S and call the resulting set S', then S' is a winning coalition and not a minimal winning coalition. If we remove the smallest player in S from S' then we either end up with a losing coalition, or another minimal winning coalition. The former case implies that we have identified a new swing, and the latter case implies that we have identified $|S| \geq 1$ new swings. So for each player in L we can identify at least 1 swing.

If we add a player x in H to S and call the resulting set S', then S' is a winning coalition and not a minimal winning coalition. If we remove the heaviest player in S from S' then we end up with a new winning coalition for which it holds that it turns into a losing coalition if we remove x. So for each player in H we can identify at least 1 swing.

If we add a player x in B to S and call the resulting set S', then S' is a winning coalition that is not minimal. If we remove the heaviest player in S from S' that is lower than x, we end up with a winning coalition for which it holds that if we remove x, it turns into a losing coalition. So for each player in H, we have identified a swing.

So in total we have identified $|L| + |H| + |B| = |N| - |S|$ swings additional to the $|S|$ swings in the minimal winning coalition S. So in a weighted majority game there are at least $|S| + |N| - |S| = |N|$ swings.

To see that this lower bound is tight consider the weighted majority game in which only the grand coalition is winning. □

Theorem 94. *For any $i, ji \neq j, 1 \leq i, j \leq n$: In a weighted voting game, if C_1 is a swing coalition for player i and does not contain player j, and if C_2 is a swing coalition for player j and does not contain player i, then $C_1 \cup C_2$ is not a swing coalition for player i or j.*

Proof. Suppose w.l.o.g. $w_i \geq w_j$. We have $q - w_i \leq w(C_1 \{i\}) < q$ and $q - w_j \leq w(C_2 \{j\}) < q$. Let $C_3 = C_1 \{i\} \cup C_2 \{j\}$. $w(C_3) >= q - w_i$. Now we divide the proof in two cases: if $w(C_1) \leq w(C_2)$ then we have $w(C_1 \{i\}) \leq w(C_2 \{j\})$ and hence $w(C_3) \geq w(C_2 \{j\}) \geq q - w_j$. $C_1 \cup C_2 = C_3 \cup \{i\} \cup \{j\}$ and $w(C_3 \cup \{i\} \cup \{j\}) \geq q - w_j + w_i + w_j = q + w_i$. Because $w_i > w_j$, $C_1 \cup C_2$ is neither a swing coalition for i nor j.

In the case that $w(C_1) > w(C_2)$ it must be that either $w(C_1 \{i\}) = w(C_2 \{j\})$ and $w_j > w_i$ or $w(C_1 \{i\}) > w(C_2 \{j\})$ and $w_j = w_i$. In the first case $C_3 \geq q - w_j$, hence $w(C_3 \cup \{i, j\}) = q + w_i$ and $C_1 \cup C_2$ is not a swing coalition. In the second case, $C_3 \supseteq C_2 \{i\}$, so $w(C_3) > q - w_j$, $w(C_3 \cup \{i, j\}) = q + w_i$ and $C_1 \cup C_2$ is not a swing coalition.

By adding w_i and w_j to C_3 we get $w(C_3 \cup \{i, j\}) \geq q + w_j$. There are now 4 cases: The first three are that C_3 contains i, j or both: Then $C_3 \supseteq C_2 \cup i$. Because $v(C_2) = 1$, $v(C_2 \cup i) = 1$ and it follows that i is not a swing in C_3. Also, because $w_i \geq w_j$, w_j also can not be a swing player in $C_2 \cup \{i\}$ and it follows that i is not a swing in C_3. □

Theorem 95. *In a weighted voting game, for two players i and j, there is no pair of coalitions C_1 and C_2 such that i is a swing coalition for player i and not for player j, and C_2 is a swing coalition for player j and not for player i.*

Proof. Obvious, but anyway: Assume w.l.o.g. that $w_i \geq w_j$. If $w_i = w_j$ then C_1 and C_2 are swing coalitions for both players. If $w_i > w_j$, then C_1 may be a swing coalition for player i and not for player j, but then if C_2 is a swing coalition for player j, it is also for i because $w_i > w_j$. □

Theorem 96. *For any $i, j, i \neq j, 1 \leq i, j \leq n, w_i > w_j$: In a weighted voting game, if C_1 is a swing coalition for player i and not for player j, and if C_2 is a swing coalition for player j, then $C_1 \cup C_2$ is not a swing coalition for player j.*

Proof. Clearly $C_1 \backslash \{j\}$ is a swing coalition for player i. $q - w_i \leq w(C_1 \backslash \{i, j\}) < q$. Now either C_2 contains i or not. If C_2 does not contain i, then by theorem 94, $C_1 \backslash \{j\} \cup C_2$ is not a swing coalition for player i or j, and hence $C_1 \cup C_2$ also isn't.

If C_2 does contain i, C_2 is a swing coalition for i because $w_i \geq w_j$. $w(C_1 \cup C_2) = w(C_1 \backslash \{i, j\} \cup C_2 \backslash \{i, j\}) + w_i + w_j$. Because $w(C_1 \backslash \{i, j\} \cup C_2 \backslash \{i, j\}) \geq q - w_i$, it follows that $w(C_1 \backslash \{i, j\} \cup C_2 \backslash \{i, j\}) \geq q - w_i$, so $w(C_1 \cup C_2) \geq q + w_j$, so the theorem follows. □