MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Křišťan Jan Matyáš** Personal ID number: **457795**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Specialisation: **Bioinformatics**

II. Master’s thesis details

Master’s thesis title in English:

Multiplayer games in the context of computational biology

Master’s thesis title in Czech:

Hry více hráču v kontextu výpočetní biologie

Guidelines:

Various multiplayer games find their application in computational biology.
Among possible applications is the research of evolutionary strategies or gene expressions in cells. Gene expression can be studied using the microarray technology. The analysis of the resulting data is a complex problem and there are many different ways to process it.
We focus on the class of microarray games described by Moretti, Patrone and Bonassi in their paper "The class of microarray games and the relevance index for genes (2007) [1]".
The authors employ theory of coalitional games to analyze data of gene expressions, showing practical results of such analysis.
The definition of the microarray games allows us to compute for each gene its Shapley value and Banzhaf value. These values are used in the subsequent analysis.
Namely, we will use models based on the theory of coalitional games to analyze gene expressions data in cases, where we differentiate between two classes (for example healthy and diseased cells).
Those models can be used for prediction of the class of a new sample and also for finding highly relevant genes.
Effectiveness of these models can be compared with previously known methods. Also the results of our techniques can be compared with relevant biological research.
The thesis will include proofs of correctness and analysis of asymptotic complexity of all algorithms.
Moreover, we will provide experimental evaluations on the following datasets:
Microarray analysis of blood samples from breast cancer patients:
https://discovery.lifemapsc.com/gene-expression-signals/high-throughput/blood-microarray-microarray-analysis-of-blood-samples-from-breast-cancer-patients
Microarray analysis of lung tissues from non-small cell lung cancer patients:
https://discovery.lifemapsc.com/gene-expression-signals/high-throughput/lung-microarray-microarray-analysis-of-lung-tissues-from-non-small-cell-lung-cancer-patients
Microarray analysis of brain tissues of schizophrenia patients:
https://discovery.lifemapsc.com/gene-expression-signals/high-throughput/brain-microarray-microarray-analysis-of-brain-tissues-of-schizophrenia-patients
We will compare the results of our methods with methods described in the included literature.
Also we will compare with the results in [2] using the same dataset as in [2].
The goals of this thesis are:
1. To survey multiplayer games and their possible relevance to computational biology and general.
2. To study the microarray games, Banzhaf value and Shapley value.

3. An effective implementation of a package for the programming language R. The package will contain methods for computing Banzhaf value and analyzing gene expression data.
4. Explore several other multiplayer games and their characteristics relevant to computational biology. Namely, explore predictive models based on coalitional games for gene expressions data.
5. Evalute the implementation of the package on the previously mentioned datasets. Compare the packages ability to predict class with the state of the art methods and the coalitional game methods described in the included literature.

Bibliography / sources:

[1] S. Moretti, F. Patrone, S. Bonassi: The class of microarray games and the relevance index for genes.
in: An Official Journal of the Spanish Society of Statistics and Operations Research 15 (2007), 256-280.
[2] S. Moretti, D. van Leeuwen, H. Gmuender, S. Bonassi, J. van Delft, J. Kleinjans, F. Patrone, D.F. Merlo, Combining Shapley value and statistics to the analysis of gene expression data in children exposed to air pollution, BMC Bioinformatics 9 (2008) 361.
[3] R. Lucchetti, S. Moretti, F. Patrone, P. Radrizzani, The Shapley and Banzhaf indices in microarray games, Computers & Operations Research 37 (2010) 1406–1412.
[4] S. Moretti, Statistical analysis of the Shapley value for microarray games, Computers & Operations Research 37 (2010) 1413–1418.

Name and workplace of master's thesis supervisor:

RNDr. Tomáš Valla, Ph.D., Department of Theoretical Computer Science, FIT

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **19.02.2020** Deadline for master's thesis submission: **22.05.2020**

Assignment valid until: **19.02.2022**

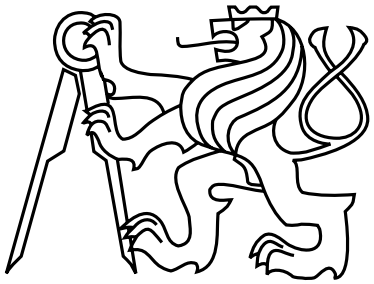
RNDr. Tomáš Valla, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science



Master's Thesis

Multiplayer games in the context of computational biology

Bc. Jan Matyáš Křišťan

Supervisor: RNDr. Tomáš Valla, Ph.D.

Study Programme: Open Informatics

Field of Study: Bioinformatics

May, 2020

Acknowledgements

I would like to thank my thesis advisor Tomáš Valla for his long term cooperation since my bachelor studies, including being the advisor for this and my bachelor thesis, in teaching BI-AG1, and many others endeavours. I would also like to thank my students in the BI-AG1 course for enduring my teaching of them.

I want to thank Kristýna Glasová for her support and encouragement. I also want to thank my friends for their helpful discussions, namely Samuel Křišťan and Josef Erik Sedláček. I would also like to thank Václav Blažej for discussions and letting me use his office as a study room.

Declaration

I declare that I elaborated this thesis on my own and that I mentioned all the information sources and literature that have been used in accordance with the Guideline for adhering to ethical principles in the course of elaborating an academic final thesis.

In Prague on May 22, 2020 .....

Abstract

In this thesis, we study application of game theory to microarray analysis. We focus on models consisting of weighted voting games, which are defined by an assignment of a weight to each player and a quota, which is the minimal total weight required for a vote to pass.

A power index estimates the influence of a player in the game. We present new algorithms computing two of the most popular power indices in weighted voting games: the Shapley-Shubik power index and the Banzhaf index. We show that under assumption of logarithmic cost per arithmetic operation, our algorithm for computing the Shapley-Shubik power index is asymptotically faster than previously published methods. We also argue that this model of computation is necessary to accurately measure complexity of our algorithms, as well as those previously published.

In the second part of the thesis, we construct weighted voting games based on results of microarray experiments and use power indices to identify differentially expressed genes. Based on this identification, we construct predictive models and show their success on real world data.

Abstrakt

V této práce studujeme aplikaci herní teorie pro analýzu experimentů s DNA čipy. Zaměřujeme se na modely založené na hlasovacích hrách, které jsou definované přiřazením váhy každému hráči a kvótou, jež představuje minimální váhu, pro níž je výsledek hlasování pozitivní.

Takzvaný power index odhaduje vliv hráče v dané hře. Představujeme nové algorithmy pro výpočet dvou z nejpopulárnějších power indexů: Shapley-Shubik power index a Banzhaf index. Ukazujeme, že při předpokladu logaritmické ceny pro aritmetickou operaci, náš algoritmus pro výpočet Shapley-Shubik power indexu je asymptoticky rychlejší než jiné, dřívě publikované metody. Dále předkládame argument pro nutnonost tohoto výpočetního modelu pro přesné určení složitosti jak našich algoritmů, tak již dříve představených algoritmů.

V druhé části práce konstruujeme hlasovací hry na základě výsledů experimentů s DNA čipy a používáme power indexy pro nalezení genů s rozdílnou expresí. Na základě těchto nálezů sestavíme prediktivní modely a ukážeme jejich úspěšnost na reálných datech.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we contribute to the two following fields: game theory and analysis of gene expression data. In the first part of the thesis, we limit our focus to game theory. In the second part, we use our theoretical results and apply them to derive useful information out of gene expression data on specific samples.

1.1 Game theory

First, we wish to introduce the reader to the basics of game theory. Game theory is in the most general terms a mathematical study of interaction between multiple selfish agents. It finds its application, among others, in economy, social sciences, biology and computer science. Some situations modeled by game theory may be characterized by conflict, for example agents must compete for a limited shared resource. Such a situation may be for example a political campaign (the resource being the votes of the citizens) or a game of chess (the resource being the victory of the game).

Other situations may drive the agents to cooperate, for example pooling their resources to achieve a greater goal. Such situations arise naturally, for example investors are often incentivized to pool their money instead of each investing individually.

There are two significant classes of games which capture these situations, *non-cooperative* games and *cooperative* (also called *coalitional*) games. In both cases, we assume that the agents act rationally and in their best self-interest.

Under these assumptions and given a precise definition of the game that the agents are playing, we may derive interesting information related to the given game. For non-cooperative games, we are often interested in finding equilibria, which may be intuitively understood as a collective behaviour of all agents, which all individually consider in some sense optimal. One may therefore assume that given enough time, agents will naturally converge their behaviour to such an equilibrium.

In cooperative games, we are also interested in finding coalitions of agents such that no agent would want to leave the coalition. Another topic of interest specific to cooperative games is finding a way to divide the shared profit, such that every player agrees.

Game theory is a vast field of mathematics and surveying all of it is a daunting task. To keep the scope of the thesis reasonable, we focus more deeply on a specific class of coalitional

games. Namely, we will focus on voting games which provide a model with many practical applications.

1.2 Microarray analysis

The other topic to which we wish to introduce the reader is the analysis of gene expression data. Techniques of measuring gene expression are a powerful tool used by various life sciences. Such techniques allow us to gain an insight into the inner working of cells, which may be selected based on their species, behaviour or they may be artificially subjected to certain conditions. The following overview is based on Debnath, Prasad and Bisen [1].

There are various techniques which seek to estimate a degree to which a given gene is expressed in a cell. Among such techniques is microarray analysis, also called gene chip analysis. The idea is that a degree to which a gene is expressed can be estimated by the amount of RNA transcripts of the gene. A big advantage of this approach is that we may perform the analysis for a huge number of genes simultaneously. While RNA transcription is often only an intermediate step in synthesising a protein (or inducing some other reaction), measuring its frequency nevertheless provides a strong insight into the underlying biological processes.

We will briefly and informally describe the process of measuring gene expression using microarrays. A DNA microarray consists of a flat surface to which is attached a collection of short DNA strands. The precise position of each DNA strand is known in advance. One microarray can contain millions of these strands.

Usually, a control sample and a different (for example diseased) sample is measured at the same time. First we collect mRNA molecules from both samples. After this is done, the mRNA molecules are converted into their complementary DNA (cDNA). This is done because only the DNA molecules can bind to their complementary strands on the microarray. Every DNA strand is marked with a fluorescent probe, whose color depends on the original sample. After that, the sample strands are mixed and allowed to bind on the microarray.

After the mixing, each sample strand will with high probability attach to the complementary strand on the microarray. We can then look at a given spot on the microarray with a known DNA strand and see (approximately) how many control and diseased strands have binded to that spot. This estimates the number of mRNA transcripts with the given sequence. Furthermore, we find out the number of transcripts for both the control and diseased sample at the same time. The resulting data usually contains noise and requires further computer processing.

While microarrays are still widely used [2, 3], other techniques for measuring the levels of RNA transcriptions were recently developed, such as RNA-Seq [4]. As for our results, the development of new techniques presents no significant obstacles. We are interested only in the estimate of the level of expression of each gene and use this estimate to construct further models.

We base our models on those described by Moretti et al. [5]. In their work, they construct a class of coalitional games called microarray games. Various authors [6, 7, 8] have since extensively studied statistical and game theoretical properties of microarray games and their relevance to gene expression data.

The idea of microarray games is to consider every gene as a player and a specific condition (such as for example cancer) as a result of a coalition of players. The requirements on such a coalition are derived from results of a gene expression analysis. The next step is to use established techniques of game theory to estimate the importance of each player. This would reflect the importance of each gene in inducing the given condition.

1.3 Original results of the thesis

In Section 3.7 we present a new algorithm for computing the Shapley value of all players. We show that under a more exact model of computation, this algorithm achieves better asymptotic time complexity than previously described methods.

The previously described methods had their time complexity proven under the assumption that any arithmetic operation takes constant time. In Section 3.5 we present new lower bounds on time complexity of those algorithms when assuming logarithmic cost of arithmetic operations.

In Section 4.1.3, we show that finding strong coalitions of bounded size is NP-Hard and even W[1]-hard.

We also describe a new class of coalitional games in Section 4.2 and empirically test their viability as a tool for analyzing results of microarray experiments in Sections 5.5 and 5.6.

Furthermore, we provide an R package for efficient computation of power indices of weighted voting games. The package is described and its performance is measured in Section 5.1. The measurements show that our algorithm for computing the Shapely value in integer weighted voting games has better performance in practice then the previously fastest existing implementation.

1.4 Outline of the thesis

In Chapter 2 we provide introduction into coalitional games and established measures of importance of players. We focus on Shapley value and Banzhaf index, which are among the most popular measures of player influence.

In Chapter 3, we focus solely on weighted voting games, which are an extensively studied subclass of coalitional games. We describe a new technique of computing Banzhaf index and Shapley value in integer weighted voting games. This results in an algorithm for computing the Shapley value of all players in a given integer weighted voting game, which is asymptotically faster than other previously described methods.

Subsequently, in Chapter 4, we describe a class of coalitional games which we call *additive voting games* and show that it is a possible generalization of microarray games described by Moretti et al [5]. We also show how those games can model the situations in which gene expression can indicate a certain condition.

In Chapter 5, we show the performance of our models on real world data and compare them with other methods. We also provide notes on the implementation of the included R package and measure its performance.

Chapter 2

Introduction to coalitional games

In this chapter, we will provide necessary definitions, show practical examples and also show some of the important results regarding coalitional games. A coalitional game is defined by its set of players (agents) and a function, which assigns to every coalition its collective profit.

Definition 1. A *coalitional game in characteristic function form* is a pair (N, v) where N is the finite set of players and $v\colon 2^N \to \mathbb{R}_0^+$ such that $v(\emptyset) = 0$ is the *characteristic function* of the game.

The value given by $v(S)$ is understood as the maximum possible profit that the players in S can gain, no matter what the rest of the players in $N \setminus S$ do. We also assume that the profit is freely transferable among the players.

We will only consider coalitional games in characteristic function form and will simply call them coalitional games from now on. A subset of players is in this context called a *coalition*. Furthermore, the coalition consisting of all players is called *the grand coalition*. The following example demonstrates a simple coalitional game.

Example 2. Consider the following situation. Person A, B and C have independently come up with the idea to start a taxi service. Person A is a fast driver and can be expected to earn \$50 a day in his job as a taxi driver. Person B is an average driver and is expected to earn \$30 a day by themselves. While person C has no practical skills, they hold a great deal of investment capital. The economy is unfortunately tough and his current investment plan is expected to earn a profit of \$10 a day.

One possible alternative is to join the efforts of the individual people. For example, person A and C can start together a luxury taxi service and with the greater charge earn \$60 a day. Doing the same with person B would earn both B and C a total of \$40 a day. On the other hand, person A and B together might at best spread a good word about each other and earn a total of \$85. All three people together might be able to start an international bus service and earn a staggering amount of \$200 a day.

A coalitional game describing this situation is given as $(N = \{A, B, C\}, v)$ with the values of v being $v(\emptyset) = 0, v(A) = 50, v(B) = 30, v(C) = 10, v(\{A, B\}) = 85, v(\{A, C\}) = 60, v(\{B, C\}) = 40, v(\{A, B, C\}) = 200$.

Naturally, one might wonder how each person would behave and what fraction of the profit they would demand for themselves. For example, it is clear that no rational, selfish person would agree to join a coalition, unless they were to gain at least as much as they would earn by themselves.

It is a fundamental assumption that every player is *rational* in the sense that they seek to maximize their own profit. Under these assumptions, we can now try to predict the behaviour of the players. Namely, we can study which division of the profit between individual players they would find acceptable. Of course, the suitability of various ways to divide the profit depends on properties of the game. A common assumption is that it is always beneficial (or not harmful) for players to join their coalitions.

Definition 3. We say that a coalitional game (N, v) is *superadditive* if for every two $S, T \subseteq N$ such that $S \cap T = \emptyset$ holds

$$v(S \cup T) \geq v(S) + v(T)$$

This property is often assumed because it allows us to assume that the grand coalition will form. The game described in Example 2 is superadditive.

2.1 Solution concepts

First we show a formal definition of the notion of dividing the profit given a specific game.

Definition 4. A *payoff vector* for a game (N, v) is a vector $x \in \mathbb{R}^N$ such that x_i is the share of the collective profit that is given to the player $i \in N$.

We assume that the given games are superadditive and therefore the payoff vector considers all the players. Usually we want the payoff vector to fulfill some set of properties which we find desirable. Which properties to choose is not obvious and many different ways to divide the profit have been proposed.

A *solution concept* is a certain set of rules for dividing the profit. Therefore, a solution concept defines for each game a set of payoff vectors. Many different solution concepts have been proposed based on different assumptions about the behaviour of the players or fairness of the rules. We present some of the most commonly required properties for solution concepts.

Property 1 (Efficiency)**.** Every payoff vector $x \in \mathbb{R}^N$ given by the solution concept splits all of the profit among the players. That is for every coalition $S \subseteq N$

$$\sum_{i \in S} x_i = v(S)$$

Property 2 (Individual rationality)**.** Necessary given the assumptions that each player is selfish and rational. The property is satisfied if the profit is distributed in such a way that each player does not earn less by joining the grand coalition than by being on their own. That is

$$x_i \geq v(\{i\})$$

for each player $i \in N$.

Property 3 (Coalitional rationality)**.** No coalition will earn less by joining the grand coalition than by being on their own. Therefore it holds

$$\sum_{i \in S} x_i \geq v(S)$$

for every coalition $S \subseteq N$. Note that this implies Property 2.

2.1.1 The core

Using these properties, we can define the solution concept known as the *core*. It has been introduced by Gillies [9] and has been extensively studied since.

Definition 5. The core of a coalitional game (N, v) is the set of all payoff vectors $x \in \mathbb{R}^N$ which satisfy Properties 1 and 3. It holds that no coalition is given less profit in any x than it would earn by itself.

It can also be said that the core contains all payoff vectors that selfish and rational players will find acceptable. Note that the core may be empty in some cases. Consider the following situation.

Example 6. Three robbers are about to steal a golden bar. No single robber can carry the bar by himself but any pair is able to carry it. The profit of the coalition is therefore given by

$$v(S) = \begin{cases} 1 \text{ if } |S| \geq 2 \\ 0 \text{ otherwise} \end{cases}$$

Each payoff vector $x = (x_1, x_2, x_3)$ falls into one of two cases: either every player is given more than 0, in which case two of them can form a coalition without the third. In the other case, one of the players is given 0, in which case he may approach the other player with the lesser allocated profit and offer him a coalition, which is advantageous to both of them. Therefore, the core contains no payoff vector and is empty.

The fact that the core may be empty leads us to consider a different choice of properties.

2.1.2 Shapley value

Shapley value was introduced by Lloyd Shapley [10]. It is one of the most significant solution concepts and has been awarded by a Nobel Prize in Economics in 2012 [11]. It is given by the following definition.

Definition 7. The Shapley value of player i in a game (N, v) is given by

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

Note that $\phi(v)$ is also a payoff vector. The Shapley value can be understood as the expected profit that the player i adds at the moment of their entering the coalition, given that the players enter the coalition one by one and every ordering of the players has the same probability.

For example the players in Example 2 have Shapley values $\phi_A(v) = 87.5$, $\phi_B(v) = 67.5$ and $\phi_C(v) = 45$.

One of the most remarkable properties of the Shapley value is that it is the *only* solution concept which satisfies Property 1 (Efficiency) along with the three properties given below [12].

Let $x(v) \in \mathbb{R}^N$ be the assignment of values to each player according to a given solution concept and for a given game (N, v).

Property 4 (Null player)**.** We say that a player is *null* if they have no effect in any coalition. Formally, a player i is null if for every $S \subseteq N \setminus \{i\}$ it holds $v(S) = v(S \cup \{i\})$. The property is satisfied if every null player i has always $x_i(v) = 0$.

Property 5 (Symmetry)**.** The property is satisfied if for every two players $i, j \in N$ with $v(S \cup \{i\}) = v(S \cup \{j\})$ for every $S \subseteq N \setminus \{i, j\}$ it holds $x_i(v) = x_j(v)$. This means that if any two players are equivalent for every coalition, then they get assigned the same value.

Property 6 (Linearity)**.** For a sum of games, the value assigned to each player is the sum of values in the individual games. That is, for every $i \in N$

$$x_i(v + w) = x_i(v) + x_i(w)$$

Moreover for any $\alpha \in \mathbb{R}$ it holds

$$x_i(\alpha v) = \alpha x_i(v)$$

Theorem 8 (Shapley [12])**.** *The Shapley value is the only solution concept which satisfies Properties 1 (Efficiency), 4 (Null player), 5 (Symmetry) and 6 (Linearity).*

We will show only the easier part of a proof of this theorem. We only show that the properties hold for the Shapley value. We refer the interested reader to the cited literature for the proof that this set of properties defines the Shapley value uniquely.

To make the notation shorter, we set

$$p(S) = \frac{|S|!\ (n - |S| - 1)!}{n!}$$

for $S \subseteq N$.

Lemma 9. *Shapley value satisfies Property 4 (Null player)*

Proof. Suppose that i is a null player and thus for every $S \subseteq N \setminus \{i\}$ it holds $v(S) = v(S \cup \{i\})$. Then by Definition 7 it holds

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} p(S)(v(S \cup \{i\}) - v(S)) = \sum_{S \subseteq N \setminus \{i\}} p(S) \cdot 0 = 0$$

□

Lemma 10. *Shapley value satisfies Property 5 (Symmetry)*

Proof. Suppose that for some $i, j \in N$ and every $S \subseteq N \backslash \{i, j\}$ it holds $v(S \cup \{i\}) = v(S \cup \{j\})$. Consider the definition of the Shapley value.

$$\phi_i(v) = \sum_{S \subseteq N \backslash \{i\}} p(S)(v(S \cup \{i\}) - v(S)) =$$

$$= \sum_{S \subseteq N \backslash \{i,j\}} p(S)(v(S \cup \{i\}) - v(S)) +$$

$$+ \sum_{S \subseteq N \backslash \{i\}, j \in S} p(S)(v((S \setminus \{j\}) \cup \{i, j\}) - v((S \setminus \{j\}) \cup \{j\})) =$$

$$= \sum_{S \subseteq N \backslash \{i,j\}} p(S)(v(S \cup \{j\}) - v(S)) +$$

$$+ \sum_{S \subseteq N \backslash \{j\}, i \in S} p(S)(v((S \setminus \{i\}) \cup \{i, j\}) - v((S \setminus \{i\}) \cup \{i\})) =$$

$$= \sum_{S \subseteq N \backslash \{j\}} p(S)(v(S \cup \{j\}) - v(S)) = \phi_j(v)$$

□

Lemma 11. *Shapley value satisfies Property 6 (Linearity)*

Proof. Let (N, v) and (N, v') be coalitional games. Let $u(S) = v(S) + v'(S)$ for every $S \subseteq N$. We will show that $\phi_i(u) = \phi_i(v) + \phi_i(v')$ for every $S \subseteq N$.

$$\phi_i(v) + \phi_i(v') = \sum_{S \subseteq N \backslash \{i\}} p(S)(v(S \cup \{i\}) - v(S)) + \sum_{S \subseteq N \backslash \{i\}} p(S)(v'(S \cup \{i\}) - v'(S)) =$$

$$= \sum_{S \subseteq N \backslash \{i\}} p(S)(v(S \cup \{i\}) + v'(S \cup \{i\}) - v(S) - v'(S)) + \sum_{S \subseteq N \backslash \{i\}} p(S)(u(S \cup \{i\}) - u(S)) = \phi_i(u)$$

Similarly for multiplication by a constant $\alpha \in \mathbb{R}$. Let $u(S) = \alpha v(S)$ for every $S \subseteq N$.

$$\phi_i(u) = \sum_{S \subseteq N \backslash \{i\}} p(S)(\alpha v(S \cup \{i\}) - \alpha v(S)) = \alpha \sum_{S \subseteq N \backslash \{i\}} p(S)(v(S \cup \{i\}) - v(S)) = \alpha \phi_i(v)$$

□

Lemma 12. *Shapley value satisfies Property 1 (Efficiency)*

Proof. We will show that

$$\sum_{i \in N} \phi_i = v(N)$$

By definition of the Shapley value we get

$$\sum_{i \in N} \phi_i = \sum_{i \in N} \sum_{S \subseteq N \backslash \{i\}} p(S)(v(S \cup \{i\}) - v(S))$$

We will count how many times the terms $v(S \cup \{i\})$ and $-v(S)$ appear in the sum. Every subset of N except the empty set is added as the $v(S \cup \{i\})$ term for each of the players in $S \cup \{i\}$. Similarly the term $v(S)$ is subtracted for every proper subset of N and every player not in S. From this observation, we get

$$\sum_{i \in N} \phi_i = \sum_{S \subseteq N, S \neq \emptyset} |S| \frac{(|S|-1)!\ (n-|S|)}{n!} v(S) - \sum_{S \subsetneq N} (n-|S|) \frac{|S|!\ (n-|S|-1)}{n!} v(S) =$$

$$= \sum_{S \subsetneq N, S \neq \emptyset} \frac{(|S|)!\ (n-|S|)}{n!} v(S) - \frac{|S|!\ (n-|S|)}{n!} v(S) + \frac{n!\ 0!}{n!} v(N) - \frac{0!\ n!}{n!} v(\emptyset) =$$

$$= v(N) - v(\emptyset) = v(N)$$

Recall that by the Definition 1 of a coalitional game, it holds $v(\emptyset) = 0$.

□

The difficulty of computing the Shapley value strongly depends on the structure of the cooperative game. Computing it from definition requires the summation of an exponential number of addends, therefore in practice, the Shapley value is often approximated or the game is restricted in such a way, that the computation is more tractable.

2.2 Voting games

Weighted voting games are an intensively studied class of coalitional games. A given game is defined by a quota and by an assignment of a weight to each player. The coalition has value 1 if the total weight of the coalition is at least the quota. Otherwise it is zero. Formal definition is as follows.

Definition 13. A coalitional game $(N = \{1, 2, \ldots, n\}, v)$ is a *weighted voting game* if there exists $(q; w_1, w_2, \ldots, w_n)$ such that for the characteristic function v holds

$$v(S) = \begin{cases} 1 & \text{if } \sum_{i \in S} w_i \geq q \\ 0 & \text{otherwise} \end{cases}$$

where $q \in \mathbb{R}^+$ the *quota* of the game $w_i \in \mathbb{R}^+$ is the *weight* of the player i.

If $q, w_1, w_2, \ldots, w_n \in \mathbb{N}$ we say that the game is an *integer weighted voting game.*

The quota and assignment of weights to players uniquely defines the weighted voting game, therefore we will use the notation $(q; w_1, w_2, \ldots, w_n)$ to denote a given weighted voting game.

We are often interested in finding the power that a given player has over the outcome of the game. For example in a parliament, it is interesting to see how much a given political party can influence the outcome of a vote. As shown in the following section, the fraction of seats in the parliament that the party holds is often not equal to this influence. The following section also presents formal tools to study the influence of a player in a given game.

2.3 Power indices

The influence that a player has over the game is often measured as a real value called the player's *power index*. Formally, the definition is the same as that of the payoff vector, that is the power indices of all players are a real valued vector with exactly one value assigned to each player.

The main difference is in the context in which it is used and its desired properties. For example the property of Efficiency (Property 1) is no longer a requirement. We present some of the most well known power indices in Sections 2.3.1 and 2.3.2 and show their applications on real world examples in Examples 14 and 15.

2.3.1 Shapley-Shubik power index

The Shapley-Shubik was introduced by Shapley and Shubik [13] as a measure of power in weighted voting games. It is simply the application of the Shapley value to voting games.

It considers the following interpretation of the Shapley value. For a player $i \in N$ and some fixed ordering of players, we say that i is *pivotal* if a coalition induced by all players before i has value 0 but the coalition consisting of players before i and including i has value 1.

Imagine a situation in which the players come into the coalition one by one. Then the player is pivotal if by his entering, he will change the value of the coalition. Then by considering the definition of the Shapley value, we can see that it is the probability that a player is pivotal, given that all orderings have the same probability.

2.3.2 Banzhaf index

The Banzhaf index was originally introduced by Lionel Penrose [14] and later reinvented by John F. Banzhaf III [15] as a measure of power in weighted voting games. The index introduced by Banzhaf is now called the *normalized Banzhaf index*. The power of each player is considered proportional to the total number of different *swings* that each player can do. We say that a player i swings the outcome of a coalition if they change its value from 0 to 1 by entering it. The idea is similar to the one of a pivotal player in the Shapley-Shubik power index but here, we imagine that all the players enter the coalition at once and we do not consider their order of entering.

The number of swings of player i in a given game can be seen to be equal to

$$\theta_i = \sum_{S \subseteq N \setminus \{i\}} v(S \cup \{i\}) - v(S)$$

Then the normalized Banzhaf index of player i is given as

$$\bar{\beta}_i(v) = \frac{\theta_i}{\sum_{j \in N} \theta_j}$$

Alternatively the (non-normalized) *Banzhaf index* is given as

$$\beta_i(v) = \frac{\theta_i}{2^{n-1}}$$

The main difference in interpretation is that the normalized Banzhaf index considers the fraction of all swings, while the Banzhaf index considers the fraction out of all coalitions without i.

The Banzhaf index can be derived axiomatically much like the Shapley value. Grabisch and Roubens [16] show that Banzhaf index is the only power index which has Properties 6 (Linearity), 4 (Null player), 5 (Symmetry) along with Property 7 (2-Efficiency) which is defined as follows.

Property 7 (2-Efficiency)**.** Let $x(v) \in \mathbb{R}^N$ be a payoff vector assigned to the game (N, v) by the given solution concept. For any pair of $i, j \in N$, we first construct the game in which we "merge" the players i and j. Therefore we get a game (N', v') where

$$N' = (N \setminus \{i, j\}) \cup \{ij\}$$

and for every $S \subseteq N \setminus \{i, j\}$ we define v' so that

$$v'(S) = v(S)$$

and

$$v'(S \cup \{ij\}) = v(S \cup \{i, j\})$$

.

The property is satisfied if $x_i(v) + x_j(v) = x_{ij}(v')$ for every pair $i, j \in N$.

2.4 Examples

We use the European parliament and the Chamber of Deputies of the Czech Republic as examples of real life voting situations. To compute the power indices, we use our implementation. The algorithms for computing the power indices are described in Chapter 3 while the implementation of those algorithms is described in Chapter 5.

Example 14. We will use the European parliament as an example. We will consider each political party as a player. Then the weight of the player is set as the number of seats that the party holds in the parliament. Members of parliament not affiliated with any party will be assigned their own player. For important decisions to pass, the absolute majority of votes is required. The parliament consists of 705 seats, therefore the quota is set to 353.

Now we can get an idea of the strength of a party given by the Shapley-Shubik power index (Shapley value) or Banzhaf index, which are shown in Table 2.1.

Example 15. An interesting example is the Chamber of Deputies of the Czech Republic. Note the difference between the power indices and the fraction of the seats of each party, which can be seen in Table 2.2.

Table 2.1: Power indices in European Parliament

Party	**Number of seats**	**Fraction of seats**	**Shapley value**	**Banzhaf index**
EEP	187	0.265625000	0.293111071	0.28877129
S&D	147	0.208806818	0.195987149	0.19299683
RE	98	0.139204545	0.154777820	0.15787109
ID	76	0.107954545	0.094577142	0.09416792
Greens-EFA	67	0.095170455	0.094498797	0.09416791
ECR	61	0.086647727	0.088162083	0.09406177
GUE/NGL	39	0.055397727	0.059476269	0.06217994
29× NI	1	0.001420455	0.000669299	0.00054425

Table 2.2: Power indices in Chamber of Deputies of the Czech Republic

Party	**Number of seats**	**Fraction of seats**	**Shapley value**	**Banzhaf index**
ANO	78	0.390	0.57457820	0.674491253
ODS	23	0.115	0.08985459	0.056676187
Pirates	22	0.110	0.08344433	0.056497679
SPD	20	0.100	0.06188256	0.052749018
KSČM	15	0.075	0.04889555	0.043823634
ČSSD	14	0.070	0.04267954	0.040074973
KDU-ČSL	10	0.050	0.03042791	0.022581221
TOP 09	7	0.035	0.02518315	0.021153160
STAN	6	0.030	0.01887002	0.017225991
5× Independent	1	0.005	0.00483683	0.002945377

Chapter 3

Weighted voting games

In this chapter, we provide a survey of algorithms for computing power indices of weighted voting games and show our own methods of computing the power indices. We examine computational complexity of previously described methods and show that our method for computing Shapley value, given a computational model more resembling real world computation, achieves a better complexity.

Recall the definition of integer weighted voting game as given in Definition 13. We wish to remind the reader of the notation describing an integer weighted voting game, given as

$$(q; w_1, w_2, \ldots, w_n)$$

where $q \in \mathbb{N}$ is the quota of the game and w_i is the weight of player $i \in N$.

3.1 Survey of existing algorithms for computing power indices

It is NP-hard to decide whether a player $i \in N$ can swing any coalitions in a given weighted voting game [17]. A player has Banzhaf index and Shapley value zero if and only if they can not swing any coalition. Therefore we can not hope for a polynomial algorithm computing either of the power indices exactly.

Klinz and Woeginger [18] describe more efficient exponential algorithms for computing the power indices. Aside from computing the power indices exactly, much research has focused on their approximation. Among approximation methods are those based on multilinear extensions by Owen [19] and methods based on randomization by Castro, Gómez and Tejada [20] or by Fatima, Wooldridge and Jennings [21, 22].

Also Aziz and Paterson [23] describe an algorithm which is polynomial in case the number of weight values is bounded.

If we restrict ourselves to the case where all weights are integer, it is possible to design a pseudo-polynomial algorithm in which the complexity is a polynomial of the value of q. Such an algorithm is not polynomial in the strict sense, as the size of the representation of q is logarithmic in the size of the value of q. This implies that the algorithm is still exponential in the size of the input, which consists among others of the representation of q. The following results have been made in this area.

Tomomi Matsui and Yasuko Matsui [17] present a pseudo-polynomial algorithms for computing Banzhaf and Shapley value of a single player. Takeaki Uno [24] presents pseudo-polynomial algorithms for computing Banzhaf and Shapley values of all players. Both of those studies present algorithms based on dynamic programming. Also, in both cases the authors assume that every arithmetic operation takes constant time, which we argue is not a reasonable assumption in Section 3.5.

An alternative approach using generating functions is proposed by Bilbao, J.M., Fernández, J.R., Losada, A.J. et al. [25]. The complexity depends on the degree of the polynomial of the generating function, with the degree bounded by the sum of all weights.

Bolus [26] describes a method based on binary decision diagrams that achieves better complexity in cases where the quasi-reduced ordered binary decision diagram of the minimal winning coalitions is small and can be built quickly.

Chakravarty et al. [27] also consider various subclasses of integer weighted voting games for which the power indices can be computed more efficiently.

We focus on the case when the weights of players are integer. An overview of the complexities of the presented algorithms, including our own, can be seen below in Table 3.1.

In Section 3.5, we make the argument that assuming a constant time per arithmetic operation is not a reasonable assumption, as this significantly affects the resulting complexity of the algorithm. Instead, when assuming logarithmic cost per arithmetic operation, we re-examine previously described algorithms with the best known running time and show new lower bounds on their running time complexity. Exact definition of the computational models can be found in Section 3.1.1

Subsequently, we present new algorithms and provide an asymptotically faster method to compute the Shapley value of all players in Section 3.7. In addition to proving the improvement of time complexity, we show that our implementation of those algorithms has a better running time on test inputs in Section 5.1.2.

3.1.1 Models of computation

We present two computational models used to analyse complexity of algorithms and compare them.

3.1.1.1 RAM

The description is based on the one presented by Mareš and Valla [28, page 56]. Here we assume that every arithmetic operation takes constant time, no matter the size of operands. Furthermore, any memory cell can store a number of arbitrary size. This model is frequently used for analysis of algorithms. The model reflects real world computation if the numbers saved in the memory cells are reasonably small.

3.1.1.2 LogRAM

We will denote the RAM with logarithmic cost of arithmetic operations as *LogRAM*. The model is described also by Mareš and Valla [28, page 56]. The motivation behind this model

is to accurately measure the complexity of algorithms which use very large numbers. The model is the same as RAM except for the following changes.

- The complexity of addition and subtraction on two integers with at most n bits is $\mathcal{O}(n)$.
- The complexity of multiplication of two integers with at most n bits is denoted by $M(n)$. We assume that $M(n)$ is polynomial in n and $M(n) = \Omega(n)$.
- The complexity of multiplication of two polynomials with degree at most d and with each coefficient being an integer with at most n bits is denoted by $M_p(d, n)$. We assume that $M_p(d, n)$ is polynomial in d and n and that $M_p(d, n) = \Omega(dn)$.

We do not assume explicit time complexities for multiplication, as it is still a researched topic with possible future improvements and is a complicated topic by itself. But we allow the assumption that integer multiplication has linear space complexity. This holds for example for the commonly used Schöhagen-Strassen multiplication algorithm [29] which has time complexity $\mathcal{O}(n \log(n) \log\log(n))$ and space complexity $\mathcal{O}(n)$ [30].

3.1.2 Complexities of existing algorithms

The following table shows time complexities of the known algorithms which can compute power indices in integer weighted voting games. All of the listed algorithms have had their complexities proven under the assumption of constant time per arithmetic operations regardless of the size of the operands. For our algorithms we include the complexity under the assumption of logarithmic cost per bit operation.

We choose to show lower bounds on Uno's algorithms because their upper bounds are in general case the best and we wish to show an improvement on those algorithms.

Table 3.1: Asymptotic complexities of exact algorithms for computing power indices of all players in integer weighted voting games

Power index	**RAM**	**LogRAM**	**Reference**
Banzhaf index	$\mathcal{O}(nq)$	$\mathcal{O}(n^2q)$ by Lemma 39	This thesis
Banzhaf index	$\mathcal{O}(nq)$	$\Omega(n^2q)$ by Proposition 31	[24]
Banzhaf index	$\mathcal{O}(nq)$ (see [2])		[26]
Banzhaf index	$\mathcal{O}(n^2q)$		[31]
Banzhaf index	$\mathcal{O}(n^2C)$ (see [1])		[25]
Banzhaf index	$\mathcal{O}(k(\frac{n}{k})^k)$ (see [3])		[23]
Banzhaf index	$\mathcal{O}(n^2\ 1.415^n)$		[18]
Shapley value	$\mathcal{O}(n^2q)$	$\mathcal{O}(n^3q + n^2M(n\log(n)))$ by Lemma 40	This thesis
Shapley value	$\mathcal{O}(n^2q)$	$\Omega(n^2qM(n\log(n)))$ by Proposition 32	[24]
Shapley value	$\mathcal{O}(n^2C)$ (see [1])		[25]
Shapley value	$\mathcal{O}(n^3q)$		[31]
Shapley value	$\mathcal{O}(n^3q)$ (see [2])		[26]
Shapley value	$\mathcal{O}(n\ 1.415^n)$		[18]

[1]The number of nonzero coefficients of the generating function is bounded by C. For the Banzhaf index, this number is equal to the number of different values that the total weight of any coalition can take. For the Shapley value, this is equal to the number of unique pairs of total weight and number of players that can be found in any coalition.

[2]These bounds are the expected value. The author also shows bounds as a function of the size of quasi-reduced ordered binary decision diagrams of the minimal winning coalitions. Those are not clearly comparable with the bounds of the other algorithms.

[3]The number of unique weight values is bounded by k.

3.2 Additional notation

The following notation will be used in the subsequent text.

Let $S \subseteq N$ for some integer weighted voting game (N, v) with $(q; w_1, \ldots, w_n)$. We say that the *total weight* of S the sum of weights over all players, that is

$$w(S) = \sum_{i \in S} w_i$$

We define $f(S, i, j)$ to be equal to the number of coalitions on j players from S, such that the total weight is i. For example, it holds $f(\emptyset, 0, 0) = f(\{i\}, w_i, 1) = 1$.

Similarly, we define $f(S, i)$ to be the number of coalitions of players from S with total weight i. More formally, for any $S \subseteq N$ and $i, j \in \mathbb{N}$ it holds

$$f(S, i, j) = |\{T \mid w(T) = i \wedge |T| = j \wedge T \subseteq S\}|$$
$$f(S, i) = |\{T \mid w(T) = i \wedge T \subseteq S\}|$$

By $v * u$, we denote the convolution of vectors v and u. By $v *_q u$ we denote the first q elements of $v * u$. By $v' *_{q,n} u'$ we denote the matrix consisting of the first q rows and n columns of $v' * u'$ where v and u are matrices. We assume that log denotes the logarithm of base 2.

3.3 Preliminaries

In this section, we present the observations used in our algorithms. The approach of computing the power indices using generating functions is described by Bilbao et al. [25]. The observation is that the power indices can be calculated by taking a product of certain polynomials and taking coefficients of the resulting polynomial. We summarize those observations in the following Lemmas. For the sake of brevity, we provide proofs without using generating functions.

Lemma 16 (Bilbao et al. [25])**.** *Let $S, S' \subseteq N$ for some weighted voting game, such that $S \cap S' = \emptyset$. Then it holds $(f(S) * f(S'))(i) = f(S \cup S', i)$.*

Proof. It holds

$$f(S \cup S', i) = \sum_{j=0}^{i} f(S, i-j) f(S', j)$$

To create a coalition of total weight i, we must choose some (possibly none) players from S and some from S', such that the total sum of weights is i. This can be seen to be equivalent to the definition of discrete convolution. □

This immediately implies the following observation.

Observation 17. *For every $S \subseteq N$ and $p \in N \setminus S$ it holds*

$$f(S \cup \{p\}, i) = \begin{cases} f(S, i) & \text{if } i < w_p \\ f(S, i) + f(S, i - w_p) & \text{otherwise} \end{cases}$$

Proof. Note that $f(\{p\}, i)$ has non-zero values for at most two different values of i, which are $f(\{p\}, 0) = f(\{p\}, w_p) = 1$. The given relation then follows from Lemma 16.

□

A similar observation holds for $f(S, i, j)$.

Lemma 18 (Bilbao et al. [25])**.** *Let $S, S' \subseteq N$ for some weighted voting game, such that $S \cap S' = \emptyset$. Then it holds $(f(S) * f(S'))(i, j) = f(S \cup S', i, j)$.*

Proof. It holds

$$f(S \cup S', i, j) = \sum_{k=0}^{i} \sum_{\ell=0}^{j} f(S, i-k, j-\ell) f(S', k, \ell)$$

To create a coalition of total weight i, we must choose some (possibly none) players from S and some from S', such that the total sum of weights is i and the total number of players is j. This can be seen to be equivalent to the definition of two dimensional discrete convolution. □

This also immediately implies the following observation.

Observation 19. *For every $S \subseteq N$ and $p \in N \setminus S$ it holds*

$$f(S \cup \{p\}, i) = \begin{cases} f(S, i, j) & \text{if } i < w_p \text{ or } j = 0 \\ f(S, i, j) + f(S, i - w_p, j - 1) & \text{otherwise} \end{cases}$$

Proof. Note that $f(\{p\}, i, j)$ has non-zero values exactly two different values of (i, j), which are $f(\{p\}, 0, 0) = f(\{p\}, w_p, 1) = 1$. The given relation then follows from Lemma 18. □

It is known that computing multiplication of polynomials is equivalent to computing convolution of two vectors. Various algorithms have been devised for fast polynomial multiplication, which are therefore applicable for practical computation of the relation given in Lemma 16. The following Lemma shows that also the relation given in Lemma 18 can be computed using algorithms for fast polynomial multiplication.

Lemma 20. *Two dimensional convolution of two integer matrices with at most n rows and at most m columns can be done in time $\mathcal{O}(M_p(nm, k))$ where k is the maximum number of bits of any coefficient.*

Proof. Naghizadeh and Mauricio [32] describe a straightforward reduction of two demensional discrete convolution to one dimensional discrete convolution with size of the input unchanged. □

The following Lemmas show a simple upper bound on the values of $f(S, i)$ and $f(S, i, j)$.

Lemma 21. *Let $S \subseteq N$ such that $|S| = k$. Then $f(S, i) \leq 2^k$ for every $i \in \mathbb{N}$.*

Proof. We will show a proof by induction over the size of S denoted by k. For $k = 0$ this holds, as every value of $f(\emptyset, i)$ is 0 except for $f(\emptyset, 0) = 1$ which is in this case exactly 2^k.

Now we show that if $\max_{i \in \{0,\dots,q-1\}} f(S, i) \leq 2^k$ with $k = |S|$, then

$$\max_{i \in \{0,\dots,q-1\}} f(S \cup \{p\}, i) \leq 2^{k+1}$$

for any player $p \in N \setminus S$. Note that we can obtain $f(S \cup \{p\}, i)$ from $f(S, i)$ by the relation given by Observation 17.

By the induction hypothesis, it holds that $f(S, i) \leq 2^k \leq 2^{k+1}$ and also

$$f(S, i) + f(S \cup \{p\}, i - w_p) \leq 2^k + 2^k = 2^{k+1}$$

□

Lemma 22. *Let $S \subseteq N$ such that $|S| = k$. Then $f(S, i, j) \leq 2^k$ for every $i, j \in \mathbb{N}$.*

Proof. Note that for any $S \subseteq N$ it holds

$$f(S, i) = \sum_{j=0}^{n} f(S, i, j)$$

This can be seen from the definition of f as $f(S, i, j)$ is the number of coalitions of weight i on j players. Summing over all the possible values of j must give us the number of all coalitions of weight i. Therefore by Lemma 21 it holds $f(S, i, j) \leq f(S, i) \leq 2^k$.

□

In the following Lemma, we show that all weights greater than q can be set to q without affecting the profit of any coalition.

Lemma 23. *Let $v(S)$ be the characteristic function of a weighted voting game with $g = (q; w_1, w_2, \dots, w_n)$ and let $v'(S)$ be the characteristic function of a weighted voting game with*

$$g' = (q; \min(w_1, q), \min(w_2, q), \dots, \min(w_n, q))$$

Then for every $S \subseteq N$ it holds $v(S) = v'(S)$.

Proof. First we show that if $v'(S) = 1$ then $v(S) = 1$. It holds

$$\sum_{i \in S} \min(q, w_i) \leq \sum_{i \in S} w_i$$

Note that $v'(S) = 1$ implies $\sum_{i \in S} \min(q, w_i) \geq q$ which in turn implies $\sum_{i \in S} w_i \geq q$ and therefore $v(S) = 1$.

Now we show that if $v(S) = 1$ then $v'(S) = 1$. Consider two cases. In the first case the weight of every player in S is at most q and therefore total weight of S is the same in both games. In the other case it must hold that there is a player $i \in S$ such that $w_i > q$. This implies that $w_i = q$ in the game g' and therefore $v'(S)$ must be 1. □

3.4 Bounds on complexity of dynamic programming methods

In this section, we construct tools further used in analysis in the following sections. We show bounds on the required number of bits needed to keep the explicit values of $f(S, i)$ and $f(S, i, j)$ as binary numbers. Those bounds are applicable to any methods that compute $f(S, i)$ or $f(S, i, j)$ explicitly for a sufficient number of parameters.

The following definition introduces notation that helps with computing the number of bits when representing sequences as a sequence of binary numbers.

Definition 24. Let $n \in \mathbb{N}$. Then by $bits(n)$ we denote the number of bits in binary representation of the integer n. It holds

$$bits(n) = \lceil \log(n) \rceil + 1$$

Let $a = (a_1, a_2, \ldots, a_k)$ be a sequence of k elements such that every $a_i \in \mathbb{N}$. Then we set

$$bits(a) \coloneqq \sum_{i=1}^{k} bits(a_i)$$

To represent a sequence a of integers in a computer in the most naive way (simply keeping the binary representation of each number independently) we need at least $bits(a)$ bits of memory.

The following definition and the subsequent technical Lemma will be used in the analysis in the later part.

Definition 25. Let $B(n)$ be a sequence of integers such that

$$B(n) \coloneqq (\binom{n}{0}, \binom{n}{1}, \binom{n}{2}, \ldots, \binom{n}{n-1}, \binom{n}{n})$$

Lemma 26. *It holds that* $bits(B(n)) = \Omega(n^2)$

Proof. The lower bound is shown as follows. We expand by the Definitions 25 and 24.

$$bits(B(n)) = \sum_{i=0}^{n} \lceil \log(\binom{n}{i}) \rceil + 1 \geq \sum_{i=1}^{n} \log(\binom{n}{i}) = \sum_{i=1}^{n} \log(\frac{n!}{(n-i)!\ i!}) =$$

$$= \sum_{i=1}^{n} \log(n!) - 2\sum_{i=1}^{n} \log(i!) = n\log(n!) - 2\sum_{i=1}^{n}\sum_{j=1}^{i} \log(j) = n\sum_{i=1}^{n} \log(i) - 2\sum_{i=1}^{n} (n-i+1)\log(i) =$$

$$= \sum_{i=1}^{n} n\log(i) - 2n\log(i) + 2i\log(i) - 2\log(i) = \sum_{i=1}^{n} (2i - n - 2)\log(i) =$$

$$= 2\sum_{i=1}^{n} i\log(i) - (n+2)\sum_{i=1}^{n} \log(i) \overset{1}{\geq} 2\int_{0}^{n} x\log(x)\ \mathrm{d}x - (n+2)\int_{1}^{n+1} \log(x)\ \mathrm{d}x =$$

$$= -n^2\log(n+1) + n^2\log(n) + \frac{1}{2}n^2 - 3n\log(n+1) + 2n - 2\log(n+1)$$

Note that in inequality 1, we use the integral lower bound for the added sum and use the integral upper bound for the subtracted sum. Therefore the whole expression is a valid lower bound.

Now we show that the asymptotic bounds of the expression are $\Theta(n^2)$.

$$\lim_{n\to\infty} \frac{-n^2\log(n+1) + n^2\log(n) + \frac{1}{2}n^2 - 3n\log(n+1) + 2n - 2\log(n+1)}{n^2} =$$

$$= \lim_{n\to\infty} \frac{-n^2\log(n+1) + n^2\log(n) + \frac{1}{2}n^2}{n^2} = \lim_{n\to\infty} \frac{n^2(\log(n) - \log(n+1)) + \frac{1}{2}n^2}{n^2} =$$

$$= \lim_{n\to\infty} \frac{n^2\log(\frac{n}{n+1}) + \frac{1}{2}n^2}{n^2} = \lim_{n\to\infty} \log(\frac{n}{n+1}) + \frac{1}{2} = \frac{1}{2}$$

We derived the asymptotic bound of $\Theta(n^2)$ on the lower bound of the number of bits required to represent $B(n)$. Therefore to represent $B(n)$, we need $\Omega(n^2)$ bits.

□

Lemma 27. *Let $S \subseteq N$ for some integer weighted voting game. It holds that*

$$bits(f(S,0), f(S,1), f(S,2), \ldots, f(S,q-2), f(S,q-1)) = \mathcal{O}(nq)$$

Proof. By Lemma 21 every element in $f(S,i)$ has value at most $2^{|S|}$, therefore $bits(f(S,i))$ for every i is at most $n+1$. Thus the sequence a created by values of $f(S,i)$ for every $0 \leq i < q$ has $bits(a) \leq (k+1)q = \mathcal{O}(nq)$. □

Lemma 28. *For every $n_0 \in \mathbb{N}$ there exists an integer weighted voting game on $n \geq n_0$ players such that for every $S \subseteq N$ with $|S| \geq n/2$, it holds*

$$bits(f(S,0), f(S,1), f(S,2), \ldots, f(S,q-2), f(S,q-1)) = \Omega(nq)$$

Proof. Consider the infinite class of integer weighted voting games $\mathcal{G}$, where every player has weight 1 and the quota is equal to the number of players. That is

$$\mathcal{G} = \{(N = \{1, \ldots, n\}, g = (n; w_1 = 1, w_2 = 1, \ldots, w_n = 1)) \mid n \in \mathbb{N}\}$$

We will show that for this class of inputs, the required number of bits to represent $f(S,i)$ with $|S| \geq n/2$ is at least $\Omega(nq)$.

Let $k = |S|$. Note that $f(S, i) = \binom{k}{i}$. This holds because a coalition of weight i is created by choosing exactly i players out of k, therefore there are $\binom{k}{i}$ possible coalitions of weight i on players in S. Therefore by Lemma 26, representing the sequence created by values of $f(S, i)$ for $0 \leq i \leq k$ takes $\Omega(k^2)$ bits. For any $k \geq n/2$ this results in $\Omega(n^2) = \Omega(nq)$ □

Lemma 29. *Let $S \subseteq N$ for some integer weighted voting game. It holds that*

$$\sum_{i=0}^{n} \sum_{j=0}^{q-1} bits(f(S, i, j)) = \mathcal{O}(n^2 q)$$

Proof. This follows from Lemma 22. For every $i, j \in \mathbb{N}$ it holds $f(S, i, j) \leq 2^{|S|} \leq 2^n$, therefore $bits(f(S, i, j)) \leq n + 1$. Because we are summing over nq values, we get the resulting bound of $\mathcal{O}(n^2 q)$. □

Lemma 30. *For every $n_0 \in \mathbb{N}$ there exists an integer weighted voting game on $n \geq n_0$ players such that for every $S \subseteq N$ with $|S| \geq 3n/4$, it holds*

$$\sum_{i=0}^{n} \sum_{j=0}^{q-1} bits(f(S, i, j)) = \Omega(n^2 q)$$

Proof. Consider the infinite class of integer weighted voting games

$$\mathcal{G}' = \{(N = \{1, \ldots, n\}, g = (q = \sum_i w_i; 1, 2, 3, \ldots, n/2, 1, 1, 1, \ldots, 1)) \mid n \in 4\mathbb{N}\}$$

By $4\mathbb{N}$ we denote the set of all natural numbers divisible by 4.

Note that a game $g \in \mathcal{G}'$ of n players has its quota equal to the sum of all weights. Additionaly, it has exactly $n/2 + 1$ players of weights 1 and the rest of players has weights $2, 3, 4, \ldots, n/2$. Let $S \subseteq N$ such that $|S| \geq 3n/4$. We show that to represent values of $f(S, i, j)$, we need at least $\Omega(n^2 q)$ bits.

Note that because of the size of S it has to contain at least $n/4$ players of weight 1. Let P be all players in S with weight greater than 1. It also holds that P must contain at least $n/4$ players.

Note that for every $j \in P$ and $k \in \mathbb{N}$ it holds $f(S, k + 1, j + k) \geq \binom{n/4}{k}$ because to create a coalition of weight $j + k$ we can create a coalition containing player j of weight j and k other players of weight 1. There are at least $\binom{n/4}{k}$ different ways to choose those k players.

For every $p \in P$, we get a sequence of coordinates

$$s_p = ((1, p), (2, p + 1), (3, p + 2), \ldots, (n/4 + 1, p + n/4))$$

Note that for every two different $p, p' \in P$, the sets induced by their respective s_p and $s_{p'}$ are disjoint. Therefore for every $p \in P$ we get a sequence of $n/4$ values in $f(S, i, j)$ for which it holds

$$f(S, s_{p,i}) \geq \binom{n/4}{i}$$

for every $i \in \{1, \ldots, n/4\}$. Representing all those $f(S, s_{p,i})$ for a fixed p takes $\Omega((n/4)^2)$ by Lemma 26. Representing $f(S, s_{p,i})$ for all $p \in P$ therefore requires the total of at least

$$\Omega(n/4 \cdot (n/4)^2) = \Omega(n^3) = \Omega(n^2 q)$$

bits.

□

3.5 Lower bounds on complexity of previously described methods

The main observation is that the number of coalitions of a given weight can be very high. Consider the simplest case where we have n players of weight 0. Then we have exactly 2^n coalitions of total weight 0.

This is clearly a degenerate case, as we can safely ignore all players of weight 0 when computing the power indices. It follows from Lemmas 28 and 30 that the numbers of coalitions are sufficiently high as to affect the complexity of resulting algorithms by a factor of n, even when no players of weight 0 are present.

We therefore argue that the model in which an arithmetic operation takes constant time is not sufficient for many algorithms that compute Shapley value or Banzhaf index. While the tools presented in Section 3.4 may be applicable to several published algorithms, we choose to show lower bounds on the algorithms presented by Uno [24]. The reason is that those are to our knowledge asymptotically fastest algorithms for computing power indices of integer weighted voting games.

Proposition 31. *Algorithm by Uno [24] for computing Banzhaf indices of all players has running time at least* $\Omega(n^2 q)$.

Proof. The following pseudocode describes the algorithm for computing Banzhaf indices of all players by Uno [24]. In the pseudocode, we refer to content of and notation used in the paper by Uno [24], namely in Chapter 2.

Algorithm 1 Banzhaf index of all players by Uno [24]

procedure UnoBanzhaf($g = (q; w_1, w_2, \ldots, w_n)$)
  Compute $V(f(p_{n-1}))$ by Property 1 [24, Chapter 2]
  for $i \in \{n, n-1, n-2, \ldots, 1\}$ **do**
    Compute $V(h(i))$ from $V(b(i))$ [24, Chapter 2]
    ▷ Given in [24, Chapter 2]
    $\phi_i = \sum_{z=0}^{q-1} f(i,z) \times (h(i, q-1-z) - h(i, \max(q - w_i - z, 0) - 1))$
    if $i < n$ **then**
      Compute $V(b(i))$ from $V(b(i+1))$ by the relation in [24, Chapter 2]
    end if
  end for
  return ϕ
end procedure

Note that the definition of $f(i, y)$ by Uno [24, Chapter 2] is equal to our definition of $f(\{1, 2, \ldots, i\}, y)$ given in Section 3.2. Also note that on line 2 we compute values of $f(\{1, 2, \ldots, i\}, y)$ for every $1 \leq i \leq n$ and $0 \leq y < q$. Therefore Lemma 28 applies and it follows that we have to perform at least $n/2$ additions on a total of $\Omega(nq)$ bits. Therefore the resulting complexity is at least $\Omega(n^2 q)$.

□

Proposition 32. *Algorithm by Uno [24] for computing Shapley values of all players has running time in the worst case at least* $\Omega(n^2 q M(n \log(n)))$.

Proof. The following pseudocode describes the algorithm for computing Shapley value of all players by Uno [24]. In the pseudocode, we refer to content of and notation used in the paper by Uno [24], namely in Chapter 4.

Algorithm 2 Shapley value of all players by Uno [24]

```
1:  procedure UnoShapley(g = (q; w_1, w_2, ..., w_n))
2:      Compute V(f(p_{n-1})) by Property 2 [24, Chapter 4]
3:      for i ∈ {n, n-1, n-2, ..., 1} do
4:          Compute V(h(i)) from V(b(i)) [24, Chapter 4]
5:          φ_i = Σ_{z=0}^{q-1} Σ_{k=0}^{i-1} (f(i-1, k, z) × (h(i, k, q-1-z) - h(i, k, max(q-w_i-z, 0)-1)))
6:          ▷ Given in Lemma 1 [24, Chapter 4]
7:          if i < n then
8:              Compute V(b(i)) from V(b(i+1)) by Property 3 [24, Chapter 4]
9:          end if
10:     end for
11:     return φ
12: end procedure
```

The high complexity can be found on line 5. Consider for example the class of weighted voting games $\mathcal{G}$ given in proof of Lemma 28. For such a game, it holds $b(i, k, y) \geq \lfloor (n/2) \rfloor !$ for every $i \leq n/2$, $k \leq n$ and $y \geq n/2$ by the definition of b [24, Chapter 4]. This holds, because in this case $b(i, k, y)$ is at least the number of coalitions on players from $\{i, i+1, i+2, \ldots, n\}$ and of weight y, multiplied by $(y+k)!(n-y-k-1)!$, therefore $bits(b(i, k, y)) = \Omega(n \log(n))$. It also holds $h(i, k, y) \geq b(i, k, y)$ which can be seen from definition of h [24, Chapter 4].

Now note that the multiplication on line 5 takes at least $\mathcal{O}(M(n \log(n)))$ time for every value of z and k. Therefore the resulting complexity is at least $\Omega(n^2 q M(n \log(n)))$.

□

3.6 Computing the power index of a single player

3.6.1 Banzhaf index

We use a divide and conquer approach to compute the Banzhaf index of a single player. The idea is to recursively compute $f(S, i)$ by splitting S into two disjoint subsets of the same size until we reach an instance containing only one (or zero) player.

Algorithm 3 Computes $f(S, i)$ for $S = (\{first, first + 1, \ldots, last - 1, last\})$ and $0 \leq i < q$

```
1: procedure BanzhafRec(first, last, g = (q; w_1, w_2, ..., w_n))
2:     if first < 0 or last ≥ n then
3:         return (1)                                   ▷ Empty range - f(∅, i)
4:     else if first = last then
5:         return (1, 0, ..., 0, 1) ∈ N^{w_first}       ▷ Return f({first}, i)
6:     else
7:         mid ← ⌊(first + last)/2⌋
8:         ▷ Return f({first, ..., mid} ∪ {mid + 1, ..., last}, i)
9:         return BanzhafRec(first, mid, g) *_q BanzhafRec(mid + 1, last, g)
10:    end if
11: end procedure
```

Lemma 33. *Let $a, b \in \{1, 2, \ldots, n\}, a \leq b$.*

For a given integer weighted voting game with known $g = (q; w_1, w_2, \ldots, w_n)$ Algorithm 3 correctly computes $f(\{a, a + 1, \ldots, b\}, i)$ for every $0 \leq i < q$. Furthermore, it runs in time $\mathcal{O}(M_p(q, n) \log(n))$.

Proof. Lines 3 and 5 return correct results by definition of $f(S, i)$. On line 9, we get the first q elements of $f(S, \{first, \ldots, last\})$ by Lemma 16. Note that to compute $(x * y)_i$, we need only the first i elements of x and y. Therefore we only require at most q first elements of $f(S, i)$ for any S throughout the algorithm.

Now we show the time complexity. We can assume that the number of players is $n = 2^k$, if it is less, players of weight 0 can be added without affecting the outcome. This also does not change the asymptotic complexity of $M_p(q, n)$ because we assume that $M_p(q, n)$ is a polynomial of q and n.

In every nontrivial case of the algorithm, we perform two recursive calls. Consider an instance on the i-th level of recursion which returns $f(S, i)$ for some $S \subseteq N$ with $|S| = n/2^i$. It holds by Lemma 21 that $f(S, i) \leq 2^{n/2^i}$ and the time to merge the two instances on line 9 is at most $\mathcal{O}(M_p(q, n/2^i))$.

On the i-th level of recursion we gave 2^i. It holds $M_p(q, n/2^i) = \Omega(qn/2^i)$ therefore the i-th level of recursion takes at most $M_p(q, n)$ time. We have at most $\log(n)$ levels of recursion, therefore the final complexity is at most $\mathcal{O}(M_p(q, n) \log(n))$.

□

Algorithm 4 Banzhaf index of player p

```
1: procedure BanzhafSingle(p, g = (q; w_1, w_2, ..., w_n))
2:     r = BanzhafRec(1, i − 1, g) *_q BanzhafRec(i + 1, n, g) ▷ Computes f(N \ {p}, i)
3:     return 2^{−n+1} Σ_{i=q−w_i}^{q−1} r_i
4: end procedure
```

Lemma 34. *For a given integer weighted voting game with known $g = (q; w_1, w_2, \ldots, w_n)$, Algorithm 4 correctly computes the Banzhaf index of player p. Furthermore, it runs in time $\mathcal{O}(M_p(q, n) \log(n))$.*

Proof. On line 3, we compute β_p from $f(N \setminus \{p\}, i)$ by the following relation.

$$2^{n-1}\beta_i = \sum_{S \subseteq N \setminus \{p\}} v(S \cup \{i\}) - v(S) = \sum_{S \subseteq N, q - w_p \leq w(S) < q} 1 = \sum_{i=q-w_p}^{q-1} f(S \cup \{p\}, i)$$

By Lemma 33, computing r on line 2 takes $\mathcal{O}(M_p(q,n)\log(n))$ time and $\mathcal{O}(nq)$ space. The sum over elements of r can be done in time $\mathcal{O}(nq)$. This follows from Lemma 27. Division by a power of two can be done quickly. Therefore the resulting time and space complexity are the same as that of Algorithm 3.

□

3.6.2 Shapley index

We apply the same divide and conquer approach to computing the Shapley value of a single player.

Algorithm 5 Computes $f(\{first, first+1, \ldots, last-1, last\}, i, j)$ for $0 \leq i < q$ and $0 \leq j \leq n$

```
1:  procedure ShapleyRec(first, last, g = (q; w_1, w_2, ..., w_n))
2:      if first < 0 or last ≥ n then
3:          return (1)                                  ▷ Empty range - f(∅, i, j)
4:      else if first = last then
5:          r ∈ ℕ^{(w_first+1)×2}
6:          fill r with zeros
7:          r_{0,0} ← 1
8:          r_{w_first,1} ← 1
9:          return r                                    ▷ Return f({first}, i, j)
10:     else
11:         mid ← (first + last)/2
12:         ▷ Return f({first, ..., mid} ∪ {mid + 1, ..., last}, i, j)
13:         return ShapleyRec(first, mid, g) *_{q,n} ShapleyRec(mid + 1, last, g)
14:     end if
15: end procedure
```

Lemma 35. *Let $a, b \in \mathbb{N}, 1 \leq a \leq b \leq n$. For a given integer weighted voting game with known $g = (q; w_1, w_2, \ldots, w_n)$, Algorithm 5 correctly computes $f(\{a, a+1, \ldots, b\}, i, j)$. Furthermore, the algorithm runs in time $\mathcal{O}(M_p(nq, n))$.*

Proof. Lines 3 and 9 are correct by definition of $f(S, i, j)$. On line 13, we correctly merge two instances by Lemma 18.

Now we show the time complexity. We can assume that $n = 2^k$. If it is less, players of weight 0 can be added without affecting the outcome. This also does not change the asymptotic complexity of $M_p(q,n)$ because we assume that $M_p(q,n)$ is a polynomial of q and n.

In every nontrivial case of the algorithm, we perform two recursive calls. Consider an instance on the i-th level of recursion which returns $f(S, i, j)$ for some $S \subseteq N$ with $|S| = n/2^i$ and for all $0 \leq i < q$ and $0 \leq j \leq |S|$. It holds by Lemma 22 that $f(S, i, j) \leq 2^{n/2^i}$ and the merging of the two instances is done by convolution of two matrices of size $q \times (|S| + 1)$. Therefore the time to merge the two instances on line 9 is at most $\mathcal{O}(M_p(nq/2^i, n/2^i))$ by Lemma 20.

Therefore the running time is given by

$$c \sum_{i=1}^{n} 2^i M_p(nq/2^i, n/2^i) \leq c \sum_{i=1}^{n} 2^{-i} M_p(nq, n) = \mathcal{O}(M_p(nq, n))$$

for some fixed constant $c \in \mathbb{R}$.

□

Algorithm 6 Shapley index of player p

1: **procedure** ShapleySingle($p, g = (q; w_1, w_2, \ldots, w_n)$)
2: $r \leftarrow$ ShapleyRec$(1, i-1, g) *_q$ ShapleyRec$(i+1, n, g)$ ▷ Computes $f(N \setminus \{p\}, i, j)$
3: **return** $(n!)^{-1} \sum_{k=0}^{n-1} k!\ (n-k-1)! \sum_{j=q-w_i}^{q-1} r_{j,k}$
4: **end procedure**

Lemma 36. *For a given game $g = (q; w_1, w_2, \ldots, w_n)$ and a player p, Algorithm 6 correctly computes Shapley value of player p. Furthermore, the algorithm runs in time*

$$\mathcal{O}(M_p(nq, n) + nM(n \log(n)))$$

.

Proof. On line 3, we compute ϕ_p from $f(N \setminus \{p\})$ by the following relation.

$$\phi_p = \sum_{T \subseteq N \setminus \{i\}} \frac{|T|!\ (n - |T| - 1)!}{n!} (v(T \cup \{i\}) - v(T)) = \sum_{T \subseteq N, q - w_p \leq w(T) < q} \frac{|T|!\ (n - |T| - 1)!}{n!} =$$

$$= \frac{1}{n!} \sum_{i=q-w_p}^{q-1} \sum_{j=0}^{n-1} k!\ (n - j - 1)!\ f(N \setminus \{p\}, i, j)$$

By Lemma 35, the computation on line 2 is done in time $\mathcal{O}(M_p(nq, n))$ and using $\mathcal{O}(n^2 q)$ space. In addition to that, we have to compute $1!, 2!, 3!, \ldots, n!$. This can be done in time $\mathcal{O}(nM(n \log(n)))$. □

3.7 Computing power indices of all players

The following key observation is made by Uno [24].

Lemma 37 (Uno [24]). *For any $p \in N$, let $S \subseteq N$ such that $p \in S$. Then*

$$f(S \setminus \{p\}, i) = \begin{cases} f(S, i) & \text{if } i < w_p \\ f(S, i) - f(S \setminus \{p\}, i - w_p) & \text{otherwise} \end{cases}$$

Proof. We will show a proof by induction over j. In case $0 \leq j < w_i$ this holds, as the player i can not be in any coalition of weight less than w_i. Otherwise it holds that

$$f(S, i) = f(S \setminus \{p\}, i - w_i) + f(S \setminus \{p\}, i)$$

as this amounts to all coalitions of weight i with and without player i. Expressing $f(S \setminus \{p\}, i)$ concludes the proof. □

A similar observation holds regarding Shapley value.

Lemma 38 (Uno [24]). *For any $p \in N$, let $S \subseteq N$ such that $p \in S$. Then*

$$f(S \setminus \{p\}, i, j) = \begin{cases} f(S, i, j) & \text{if } i < w_i \text{ or } j = 0 \\ f(S, i, j) - f(S \setminus \{p\}, i - w_i, k - 1) & \text{otherwise} \end{cases}$$

Proof. We will show a proof by induction over $i + j$. In case $0 \leq j < w_i$ this holds, as the player p can not be in any coalition of weight less than w_i. Also if $j = 0$ this clearly holds as the number of coalitions is independent on S. Otherwise it holds that

$$f(S, i, j) = f(S \setminus \{p\}, i - w_p, j - 1) + f(S \setminus \{p\}, i, j)$$

as this amounts to all coalitions of weight i with j players with and without player p. Expressing $f(S \setminus \{p\}, i, j)$ concludes the proof. □

3.7.1 Banzhaf index

The main idea is to first compute $f(N, i)$ for all necessary values of i. Using Lemma 37 we can quickly transform $f(N, i)$ to $f(N \setminus \{p\}, i)$ for each player $p \in N$ which then allows us to compute the number of swings for each p.

Algorithm 7 Banzhaf index of all players

1: **procedure** BANZHAFALL($g = (q; w_0, w_1, \ldots, w_{n-1})$)
2: **for** $p \in \{1, \ldots, n\}$ **do**
3: $w_p \leftarrow \min(w_p, q)$
4: **end for**
5: $t \leftarrow (0, \ldots, 0) \in \mathbb{N}^{2q}$ ▷ t contains values of $f(\emptyset, i)$ for $0 \leq i < 2q$
6: **for** $p \in \{1, \ldots, n\}$ **do**
7: Recompute t by Observation 17 so that t contains values of $f(\{1, 2, \ldots, p\}, i)$
8: **end for**
9: $res \leftarrow (0, \ldots, 0) \in \mathbb{N}^n$ ▷ t now contains values of $f(N, i)$ for $0 \leq i < 2q$
10: **for** $p \in \{1, \ldots, n\}$ **do**
11: $c \leftarrow t$
12: Recompute c by Lemma 37 so c contains values of $f(N \setminus \{p\}, i)$
13: $res_p \leftarrow 2^{-n+1} \sum_{i=q-w_p}^{q-1} c_i$
14: **end for**
15: **return** res
16: **end procedure**

Theorem 39. *Algorithm 7 correctly returns a vector of Banzhaf indices of all players and runs in time $\mathcal{O}(n^2 q)$ and uses $\mathcal{O}(nq)$ memory.*

Proof. By Lemma 23, line 3 transforms the game in a way that will not affect the value of any coalition, therefore it can not affect the resulting power indices. Note that t will contain the number of coalitions of total weight up to $2q - 1$ after finishing the for loop on line 6. We show that this suffices to compute the number of swings for each player.

Suppose that a coalition of size greater than $2q - 1$ can be swinged by removing any player and therefore such a coalition would affect the outcome. To swing the coalition, we need to decrease its weight by at least $q + 1$ by removing a single player. But that is not possible, because on line 3, we set the weight of every player to be at most q.

The following will show the time complexity. The loop at line 6 requires at most $2qn$ additions of integers with at most n bits by Lemma 21. The same is true for the loop on line 10 which additionally only copies t and computes the number of swings on line 13. We assume that the division by a power of two can be done quickly. Therefore the resulting time complexity is $\mathcal{O}(n^2 q)$.

As for the space complexity, we only keep values of $f(N, i)$ and $f(N \setminus \{p\}, i)$ for $0 \leq i < 2q$ and for only one value of p. By Lemma 27 this takes $\mathcal{O}(nq)$ space.

□

3.7.2 Shapley value

The idea is similar to the one on which the algorithm for computing Banzhaf indices is based. First we compute $f(N, i, j)$ for all necessary values of i and j. Using Lemma 38 we can quickly transform $f(N, i, j)$ to $f(N \setminus \{p\}, i, j)$ for each player $p \in N$ which then allows us to compute the Shapley value of each p.

Algorithm 8 Shapley value of all players

```
1:  procedure ShapleyAll(g = (q; w_0, w_1, ..., w_{n-1}))
2:      for p ∈ {1, ..., n} do
3:          w_p ← min(w_p, q)
4:      end for
5:      t ← (0, ..., 0) ∈ ℕ^{2q×(n+1)}
6:      ▷ t contains values of f(∅, i, j) for 0 ≤ i < 2q and 0 ≤ j ≤ n
7:      for p ∈ {1, ..., n} do
8:          Recompute t by Observation 19 so that t contains values of f({1, 2, ..., p}, i, j)
9:          with 0 ≤ i < 2q and 0 ≤ j ≤ n
10:     end for
11:     res ← (0, ..., 0) ∈ ℕ^n
12:     for p ∈ {1, ..., n} do
13:         c ← t
14:         Recompute c by Lemma 38 so c contains values of f(N \ {p}, i, j)
15:         for j ∈ {0, 1, ..., n − 1} do
16:             s ← Σ_{i=q−w_p}^{q−1} c_{i,j}
17:             res_p ← res_p + (s · j! · (n − j − 1)!)
18:         end for
19:         res_i ← res_i/n!
20:     end for
21:     return res
22: end procedure
```

Theorem 40. *Algorithm 8 correctly returns a vector of Shapley values of all players and runs in time* $\mathcal{O}(n^3q + n^2M(n\log(n)))$ *and uses* $\mathcal{O}(n^2q + n^2\log(n))$ *memory.*

Proof. By Lemma 23, line 3 transforms the game in a way that will not affect the value of any coalition, therefore it can not affect the resulting power indices.

Note that t will contain the number of coalitions of total weight up to $2q-1$ after finishing the for loop on line 7. We show that this suffices to compute the Shapley value of each player. The argument is the same as the one in Theorem 39.

Suppose that a coalition of size greater than $2q-1$ can have its profit changed by removing any player and therefore such a coalition would affect the outcome. To change the profit of the coalition, we need to decrease its weight by at least $q+1$ by removing a single player. But that is not possible, because on line 3, we set the weight of every player to be at most q.

We compute the numerator of the Shapley value of player p on line 15 and divide by the denominator on line 19.

As for the time complexity, throughout the algorithm we need values of $1!, 2!, 3!, \ldots, n!$. Computing those takes $\mathcal{O}(nM(n\log(n)))$ time. Note that computing both $n!$ from $(n-1)!$ and vice versa can be done in $M(n\log(n))$ using (TODO) good division. Computing the expression on line 17 takes $\mathcal{O}(M(n\log(n)))$ and is done for each player n times, therefore it adds $\mathcal{O}(n^2M(n\log(n)))$ to the total complexity.

Additionally, the loop at line 7 requires at most $2qn^2$ additions of integers with at most n bits by Lemma 22. The same is true for the loop on line 12 which additionally copies t and

computes the numerator of the Shapley value, which has been discussed above. Therefore the resulting time complexity is $\mathcal{O}(n^3q + n^2M(n\log(n)))$.

Now we show the space complexity. At any point in time, we keep the values of $f(N, i, j)$ and $f(N \setminus \{p\}, i, j)$ for $0 \leq i < 2q$, $0 \leq j \leq n$ and for one player p. This by Lemma 29 takes $\mathcal{O}(n^2q)$ space. Additionally, the number of bits of both the numerator and denominator of res_p is $\mathcal{O}(n\log(n))$, therefore the size of the output is $\mathcal{O}(n^2\log(n))$. The total space complexity is therefore $\mathcal{O}(n^2q + n^2\log(n))$.

□

3.7.3 Possible optimizations

We show some possible straightforward optimizations of the algorithms, which may be useful in practice. We employ these optimizations in our implementation. More on the implementation can be found in Section 5.1.

Note that to compute $f(N, i)$ for all i, we can use Algorithm 3 and similarly use Algorithm 5 to compute $f(N, i, j)$ for all necessary i and j. This can be significantly faster than the simple computation given by Observations 17 and 19, provided we can quickly multiply large polynomials. This provides one possible speed-up of Algorithms 7 and 8.

Also note that after computing $f(N, i)$ and $f(N, i, j)$, the computation of the resulting power index of each player is independent. Therefore, we need to compute the power index for every unique weight only one.

Chapter 4

Coalitional games and computational biology

In this chapter we focus on microarray games described by Moretti et al. [5] and present a new class of coalitional games called additive voting games. Microarray games present a framework used to analyse data from microarray experiments. The practical use of microarray games has been shown by Moretti et al. [6].

As for our original results in this chapter, in Section 4.1.3 we show that finding strong coalitions of limited size in microarray games is NP-hard and even W[1]-hard. The notion of additive voting games, described in Section 4.2 is also an original result.

4.1 Microarray games

Morreti et al. [5] describe microarray games as a tool for extracting information about relevance of genes in situations where we want to differentiate between two classes of samples. First, we show a definition of the class of microarray games and subsequently show how they can be used to analyse gene expression data.

We provide a definition equivalent to the one in [5]. Every microarray game can be defined by a sequence of *unanimity games*. First, we define unanimity games and then show how they can be used to construct microarray games.

Definition 41. A coalitional game (N, v) is an *unanimity game* if there exists $T \subseteq N$ such that for every $S \subseteq N$

$$v(S) = \begin{cases} 1 & \text{if } T \subseteq S \\ 0 & \text{otherwise} \end{cases}$$

Every unanimity game can be uniquely defined by its set $T \subseteq N$.

Definition 42. A coalitional game (N, v) is a *microarray game* if there exists a sequence of unanimity games $(g_1, g_2, \ldots, g_m)$ such that for every $S \subseteq N$

$$v(S) = \frac{1}{m} \sum_{i=1}^{m} v_{g_i}(S)$$

where v_{g_i} is the characteristic function of the unanimity game g_i.

A microarray game can be therefore defined by a sequence $(T_1, T_2, \ldots, T_m)$ where each $T_i \subseteq N$. We will call each T_i a *check*. The value of $v(S)$ for a microarray game with $(T_1, T_2, \ldots, T_m)$ can be seen to be equal to the number of checks that are a subset of S, normalized by the number of checks.

4.1.1 Construction based on microarray experiments

Now we show how to model a result of a microarray experiment using microarray games as by Moretti et al. [5]. First, we measure expressions of two different classes of samples, for example healthy tissue and corresponding tumor tissue. The players of the game are the genes, for which we performed the experiment.

Then by some external methods, we identify genes which are differentially expressed in the tumor tissue. Moretti et al. [6] consider genes to be abnormally expressed if their expression value is at least one standard deviation lower or greater than the mean expression of the given gene in the control group.

Now we construct the sequence of checks that will define the microarray game. We create a check for every sample of tumor tissue. Each check will consist of the set of abnormally expressed genes. The coalition is then considered a set of abnormally expressed genes and its profit is proportional to the number tumor samples, which exhibit abnormal expression in only those genes that are a part of the coalition.

The profit of the coalition can be understood as something resembling "likelihood" of the coalition representing a new tumor sample. The term likelihood being used merely as an intuitive notion.

4.1.2 Computing power indices

Computing the Shapley value of a microarray game turns out to be surprisingly easy. Consider a unanimity game defined by $T \subseteq N$. Every player in $N \setminus T$ can have no effect on the outcome of the game and are a null player (as defined in Property 4).

Lemma 43. *The Shapley value of each player in a unanimity game (N, v) defined by $T \subseteq N$ is given as*

$$\phi_i = \begin{cases} 1/|T| & \text{if } i \in T \\ 0 & \text{if } i \notin T \end{cases}$$

Proof. Recall that as shown in Section 2.1.2, Shapley value holds the properties Null player, Symmetry and Efficiency (Properties 4, 5 and 1). Each player $i \in N \setminus T$ is a null player and from Property 4 (Null player) follows that $\phi_i(v) = 0$.

From Property 1 (Efficiency) follows that $v(N) = 1$ and from Property 5 (Symmetry) follows that for every two $j, j' \in T$ it holds $\phi_j(v) = \phi_{j'}(v)$, therefore $\phi_j(v) = \phi_{j'} = 1/|T|$. □

Now we can compute the Shapley values of microarray games as sums of Shapley values of its unanimity games.

Lemma 44. *The Shapley value of each player in a microarray game (N, v) defined by a sequence of checks $(T_1, T_2, \ldots, T_m)$ is given as*

$$\phi_i(v) = \sum_{i=1}^{m} \phi_i(v_i)$$

where v_i is the characteristic function of the unanimity game defined by T_i.

Proof. Follows from Definition 42 and Property 6 (Linearity) of the Shapley value, which holds by Lemma 6. □

Similar computation can be shown for the Banzhaf index.

Lemma 45. *The Banzhaf index of each player in a unanimity game (N, v) defined by $T \subseteq N$ is given as*

$$\beta_i = \begin{cases} 2^{n-|T|} & \text{if } i \in T \\ 0 & \text{if } i \notin T \end{cases}$$

Proof. The Banzhaf index satisfies the Property 4 (Null player) [19], therefore if i is a null player then $\beta_i = 0$. Otherwise, the number of swings can be counted using the following observation. The player i can swing the coalition $S \subseteq N \setminus \{i\}$ if and only all players in T except i are present in S. The presence of players in $N \setminus T$ affects nothing, therefore there are $2^{n-|T|}$ subsets that i can swing. □

Lemma 46. *The Banzhaf index of each player in a microarray game (N, v) defined by a sequence of checks $(T_1, T_2, \ldots, T_m)$ is given as*

$$\beta_i(v) = \sum_{i=1}^{m} \beta_i(v_i)$$

where v_i is the characteristic function of the unanimity game defined by T_i.

Proof. Follows from Definition 42 and Property 6 (Linearity) of the Banzhaf index value [19]. □

4.1.3 Hardness of finding small strong coalitions

This section is motivated by the following question. Given a microarray game, can we efficiently find small coalitions with large influence? In the context of gene expressions, this would mean finding small groups with high relevance to inducing for example tumorous behaviour. For its most natural formulation as a decision problem, we show that the problem is at least as hard as Clique. In the Clique problem, on the input we are given a graph and an integer k and ask, whether the graph contains a set of maximally connected vertices of size at least k.

Theorem 47. *Let $g = (N, v)$ be a microarray game and $k \in \mathbb{N}$. Then deciding if there exists a coalition $S \subseteq N$ with profit at least $c \in \mathbb{Q}$ such that $|S| \leq k$ is at least as hard as* Clique

Proof. We show a reduction from CLIQUE. Let $G = (V, E)$ be a simple graph. We construct a microarray game g, such that G has a clique of size k if and only if the maximum profit of coalition of size k is at least $\binom{k}{n} / |E(G)|$.

Let the set of players $N = V$. Let g be defined by the set of checks $C = E$. Let

$$r = \max_{S \subseteq N, |S| \leq k} v_g(S)$$

That is, r is the maximum profit a coalition of size at most k. Then $r \geq \binom{k}{2} / |C|$ if and only if G has a clique of size k.

First we show that if $r \geq \binom{k}{2} / |C|$, then G has a clique of size k. If $r \geq \binom{k}{2} / |C|$, then there must exist a coalition S^* such that $\binom{k}{2}$ checks are subsets of S and $|S| \geq k$. Therefore, the coalition S^* is a set of vertices in G, such that each pair in S^* is connected by an edge, as there is no other way to obtain $\binom{k}{2}$ edges on a subgraph induced by some k vertices. Therefore S^* corresponds to a clique of size k in G.

Now we show that if G has a clique of size k, then $r \geq \binom{k}{2} / |C|$. If there is a clique S^* in G of size k, then this clique contains exactly $\binom{k}{2}$ edges. Every edge between a pair of vertices in S^* is a subset of S^*, therefore S^* will have profit $r = \binom{k}{2} / |C|$ in g. □

This shows that finding coalitions with bounded size of a given profit is NP-hard. Using results in theory of parameterized algorithms, we can derive even stronger results.

Theorem 48 (Downey, Fellows [33])**.** INDEPENDENT SET *is complete for W[1]*

This immediately implies that CLIQUE is also W[1]-complete and therefore finding coalitions with bounded size and a given profit is also W[1]-hard. This in turn implies that there is no algorithm with running time of the form $\mathcal{O}(f(k)n^{\mathcal{O}(1)})$ [34] solving the given problem.

As shown in Section 4.2.1, every microarray game can be encoded as an additive voting game, therefore these results also extend to additive voting games.

4.2 Additive voting games

We introduce a new class of coalitional games called *additive voting games*. The motivation is to construct coalitional games which can be used as models of different behaviours arising as a result of abnormal gene expressions. In Section 4.2.1 we also show that additive voting games are a possible generalization of the class of microarray games.

Definition 49. A coalitional game (N, v) is an *additive voting game* if there exists a sequence of weighted voting games $(g_1, g_2, \ldots, g_m)$ such that for every $S \subseteq N$

$$v(S) = \sum_{i=1}^{m} v_{g_i}(S)$$

where v_{g_i} is the characteristic function of the weighted voting game g_i.

An additive voting game can be uniquely defined by its sequence of weighted voting games. For an additive voting game $g = (g_1, g_2, \ldots, g_m)$ we say that g_i is a *subgame* of g.

4.2.1 A generalization of microarray games

In this section, we show how to encode any microarray game as an additive voting game. From this follows that microarray games are a subset of additive voting games.

By saying that two games (N, v) and (N, v') are *equivalent*, we mean that for every $S \subseteq N$ it holds $v(S) = v'(S)$.

Proposition 50. *For every microarray game, there is an equivalent additive voting game*

Proof. Note the similarity of definitions of microarray games (Definition 42) and additive voting games (Definition 41). We show how to encode a unanimity game as a weighted voting game. Then we can simply transform a sequence of unanimity games into a sequence of weighted voting games.

Let (N, v) be a microarray game defined by $T \subseteq N$. Then we create the equivalent weighted voting game (N, v') by setting $q = |T|$ and

$$w_i = \begin{cases} 1 & \text{if } i \in T \\ 0 & \text{otherwise} \end{cases}$$

for every $i \in N$. Now it holds that $v'(S) = 1$ if and only if all players with weight 1 are present in S. Therefore, for every $S \subseteq N$ it holds $v(S) = v'(S)$. □

4.2.2 Computing power indices

Computing the Shapley value of an additive voting game can be done simply by summing up the Shapley values of its subgames. This fact is summarized as follows.

Lemma 51. *Let (N, v) be an additive voting game defined by a sequence of weighted voting games $(g_1, g_2, \ldots, g_m)$. Then the Shapley value of player $i \in N$ is given as*

$$\phi_i(v) = \sum_{i=1}^{m} \phi_i(v_{g_i})$$

where v_{g_i} is the characteristic function of the weighted voting game g_i.

Proof. Follows from Definition 41 of the additive voting game and Property 6 (Linearity) of the Shapley value, which is shown in Lemma 11. □

The same argument holds for the Banzhaf index.

Lemma 52. *Let (N, v) be an additive voting game defined by a sequence of weighted voting games $(g_1, g_2, \ldots, g_m)$. Then the Banzhaf index of player $i \in N$ is given as*

$$\beta_i(v) = \sum_{i=1}^{m} \beta_i(v_{g_i})$$

where v_{g_i} is the characteristic function of the weighted voting game g_i.

Proof. Follows from Definition 41 of the additive voting game and Property 6 (Linearity) of the Banzhaf index [19]. □

Efficient methods to compute power indices of the individual weighted voting games that are described in Chapter 3.

Chapter 5

Practical results

5.1 The package implementation

We present a package called `VotingGames` for the programming language R, containing methods for computing Shapley values and Banzhaf indices of integer weighted voting games and additive voting games. The package is publicly accessible [1].

We choose R as the programming language as it is a popular choice for statistical and data analysis. Furthermore, many tools used for reading and processing the results of microarray experiments are written in R, for example the packages by Bioconductor [2].

Our implementation of the algorithms is done in C++ for maximum efficiency. The implementation reflects the descriptions in Chapter 3. The R package itself consists mostly of wrapper methods.

As for dependencies of the package, NTL [3] is required fast multiplication of large polynomials and GMP [4] is required for arithmetic on big integers.

5.1.1 Existing implementations

To our knowledge, the only other implementation focusing on weighted voting games is by Uno [24], it is open source and publicly accessible [5].

Also Alejandro Saavedra-Nieves has published an R package called `GameTheoryAllocation` [6] that among other things can compute the Shapley value of coalitional games. It requires as the input an explicit statement of profit of all possible coalitions, therefore it is not suitable for large games.

[1] <https://github.com/kristja6/VotingGames>
[2] <https://bioconductor.org/packages/release/bioc/>
[3] Can be found on <https://www.shoup.net/ntl/>. Many Linux distributions include NTL in their default repositories as `libntl-dev`.
[4] Can be found on <https://gmplib.org/>. Usually a part of repositories as `libgmp-dev`
[5] <http://research.nii.ac.jp/~uno/code/powic.html>
[6] <https://cran.r-project.org/web/packages/GameTheoryAllocation/index.html>

5.1.2 Evaluation of performance

For comparison, we use Uno's own implementation of the algorithm described in [24]. The implementation is open source and can be found on Uno's website [7]. To measure the performance, we use the real elapsed time and maximum reserved memory as reported by the Unix utility `time`. We allow the maximum running time of ten minutes using another Unix utility `timeout`. The measurements were performed on a computer with AMD Ryzen 5 3600 and Corsair 16GB DDR4 3466MHz CL16. The resulting time and memory use is the average taken from 5 distinct runs for each algorithm. The measurements are presented in Table 5.1.

5.1.2.1 Synthetic data

The measurements are done on three different distributions of weights.

- in `seq`n`.in` the weights are a sequence of numbers 1 to n and the quota is n
- in `norm`n`.in` the weights are normally distributed with mean n and variance $n/2$ and the quota is $20n$.
- in `uni`n`.in` the weights are uniformly distributed in range 1 to n and the quota is n.

Note that for the larger inputs, the running time and memory usage of our algorithm are significantly lower than the Uno's implementation.

5.2 Data sets

All the used data sets are published on the NCBI's Gene Expression Omnibus (GEO) [35]. Included with the raw data sets is also a so-called series matrix. A series matrix is preprocessed raw data taken from the microarray experiment. Therefore, the processing of the data is not performed by us, instead it is done by the publisher of the dataset.

Each series matrix file contains log values of expression of each measured DNA strand. We use the series matrix files as the input of our experiments.

We provide a list of data sets and their GEO Series accession numbers

- Breast cancer [36]: GSE27562
- Lung cancer [37]: GSE18842
- Schizophrenia [38]: GSE53987
- Air pollution [39]: GSE7543. It is also used by Moretti et al. to show the practical applicability of microarray games [6]

[7] <http://research.nii.ac.jp/~uno/code/powic.html>

Table 5.1: Average running times for computing Shapley value of all players over 5 distinct runs of each algorithm

Input file	**Our algorithm**		**Uno's algorithm**	
	time	**memory**	**time**	**memory**
seq100.in	0:00.04s	10.164MB	0:00.01s	3.438MB
seq200.in	0:00.22s	17.95MB	0:00.17s	10.616MB
seq400.in	0:00.97s	33.302MB	0:06.09s	64.056MB
seq800.in	0:05.19s	73.294MB	1:44.52s	506.214MB
seq1600.in	0:28.67s	200.066MB	*>10:00.00s*	*unknown*
seq3200.in	2:56.60s	636.028MB	*>10:00.00s*	*unknown*
seq6400.in	*>10:00.00s*	*unknown*	*>10:00.00s*	*unknown*
uni100.in	0:00.51s	72.18MB	0:00.24s	25.046MB
uni200.in	0:03.24s	268.398MB	0:04.17s	152.74MB
uni400.in	0:30.41s	1062.124MB	1:03.29s	1103.702MB
uni800.in	3:47.18s	3825.712MB	*>10:00.00s*	*unknown*
uni1600.in	*>10:00.00s*	*unknown*	*>10:00.00s*	*unknown*
norm100.in	0:00.28s	58.844MB	0:00.24s	27.118MB
norm200.in	0:01.53s	157.796MB	0:03.60s	164.53MB
norm400.in	0:08.72s	415.412MB	0:59.56s	1200.876MB
norm800.in	1:09.67s	1166.314MB	*>10:00.00s*	*unknown*
norm1600.in	4:54.25s	2885.082MB	*>10:00.00s*	*unknown*
norm3200.in	*>10:00.00s*	*unknown*	*>10:00.00s*	*unknown*
eu_par.in	0:00.02s	11.122MB	0:00.00s	2.918MB
hoc.in	0:00.00s	9.156MB	0:00.00s	2.348MB
npc_congress.in	0:01.22s	130.732MB	0:11.16s	112.246MB

5.3 Definition of the models

On the input, we are given two expression matrices. One for the control group, one for the diseased group. For each patient, we are given log values of expressions for every measured DNA strand. Our model consists of a sum of weighted voting games based on those data. We create a weighted voting game for each diseased sample. Each game contains two players for each gene: one player representing upregulation of the gene, the other player representing downregulation of the gene.

The output of such a model will be a power index assigned to each player. If for example some player has a high power index, this indicates that the respective gene and its upregulation/downregulation has high influence on the condition.

We test the practical results of those models by creating regression models using expressions of genes with high power indices.

5.3.1 Method 1

Let μ_i be the average log expression level of gene i in the control group. Let s_i be the estimated standard deviation of the log expression of gene i in the control group. Then for weighted voting game j, we set the weight of two players i^- and i^+ as

$$w_{j,i^+} = \max(0, (X_{j,i} - \mu_i)/s_i)$$

$$w_{j,i^-} = \min(0, (X_{j,i} - \mu_i)/s_i)$$

where $X_{j,i}$ is the expression level of gene i in the diseased sample j. The quota for game j is set as

$$q = \lceil \frac{1}{2} \sum_{i=1}^{n} w_{j,i^+} + w_{j,i^-} \rceil$$

where n is the number of genes

This model can be seen as a generalization of the binary classification method used by Moretti et al. [6]. In their work, they classify a gene in a given diseased sample as upregulated when its log expression level is greater than $\mu_i + s_i$. Similarly, they classify it as downregulated when its log expression level is less than $\mu_i - s_i$.

5.3.2 Method 2

In the second method, we set the weights as

$$w_{j,i^+} = \max(0, \log(X_{j,i} - \mu_i))$$

$$w_{j,i^-} = \max(0, \log(X_{j,i} - \mu_i))$$

where $X_{j,i}$ is the expression level of gene i in the diseased sample j. The quota is set in the same way as in Method 1.

$$q = \lceil \frac{1}{2} \sum_{i=1}^{n} w_{j,i^+} + w_{j,i^-} \rceil$$

where n is the number of genes

5.3.3 Approximation by lowering the quota

Our methods of setting the weights and quota require a very high value of the quota. This significantly raises the computation time of our algorithms. We lower the quota a to smaller fixed value (500), so that the computations are tractable on an average modern computer. Although this significantly speeds up the runtime of the algorithms, we are given no formal guarantees about the closeness of our solution. Because the subsequent results are tested empirically, we do not consider this lack of guarantees a setback.

5.4 Evaluation of the models

We follow the approach used by Moretti et al. [6]. In their study, they compare their methods with the results of t-test. The t-test is performed for each gene separately and the resulting p-value is used as a measure of importance of the gene. In Section 5.5 we compare how similar are the sets of important genes selected by t-test and by our models. Subsequently, in Section 5.6 we test models which predict the class of a sample based on the expression levels of the most important genes. We again compare the models based on expressions of genes selected by t-test and our methods.

5.5 Genes selected as important

We present graphs showing, for a given value x, how many genes are selected among top x by both t-test and some other method based on game theory. This value is normalized by $x/100$, so that we get a percentage.

Figure 5.1: Breast cancer - overlap of genes selected by t-test and other methods

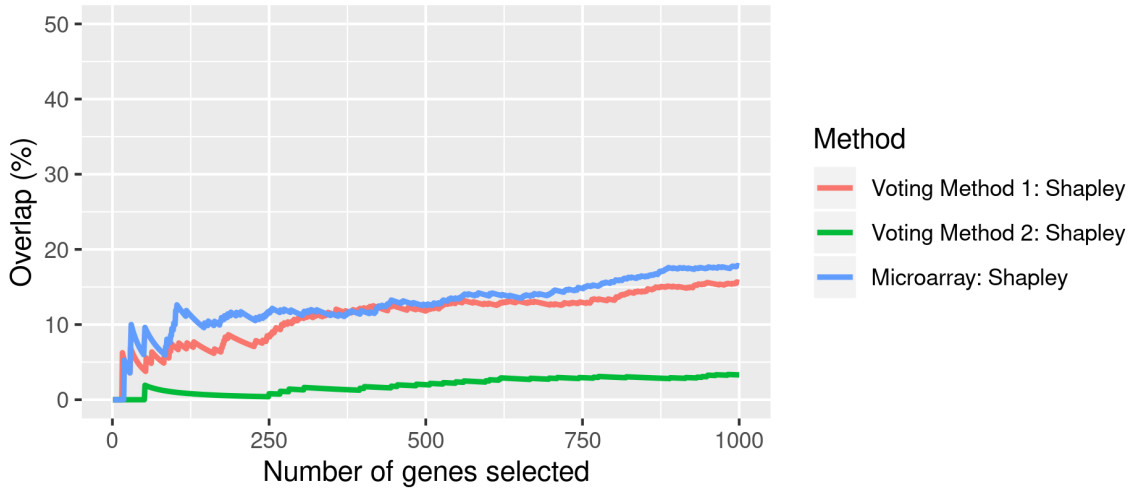


Figure 5.2: Lung cancer - overlap of genes selected by t-test and other methods

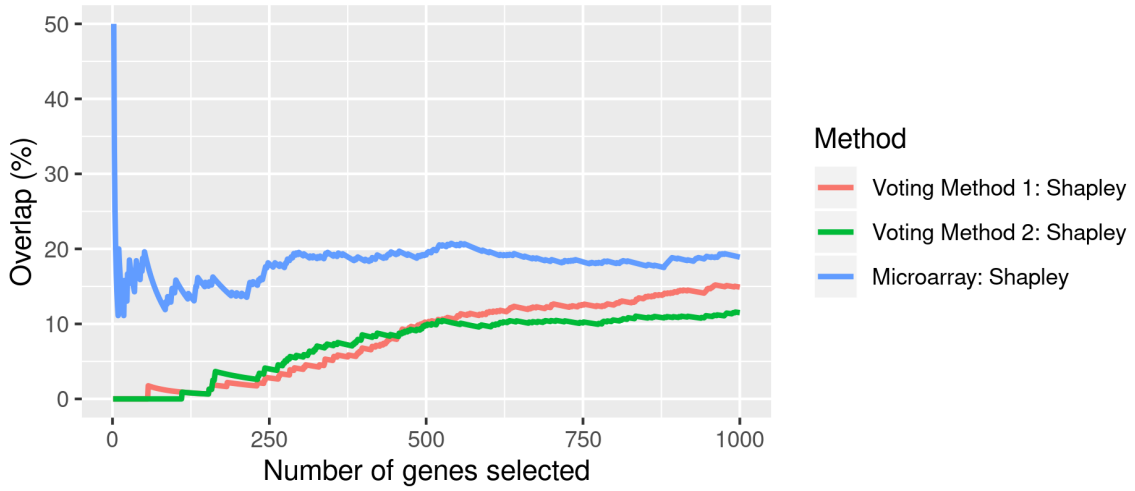


Figure 5.3: Schizophrenia - prefrontal cortex - overlap of genes selected by t-test and other methods

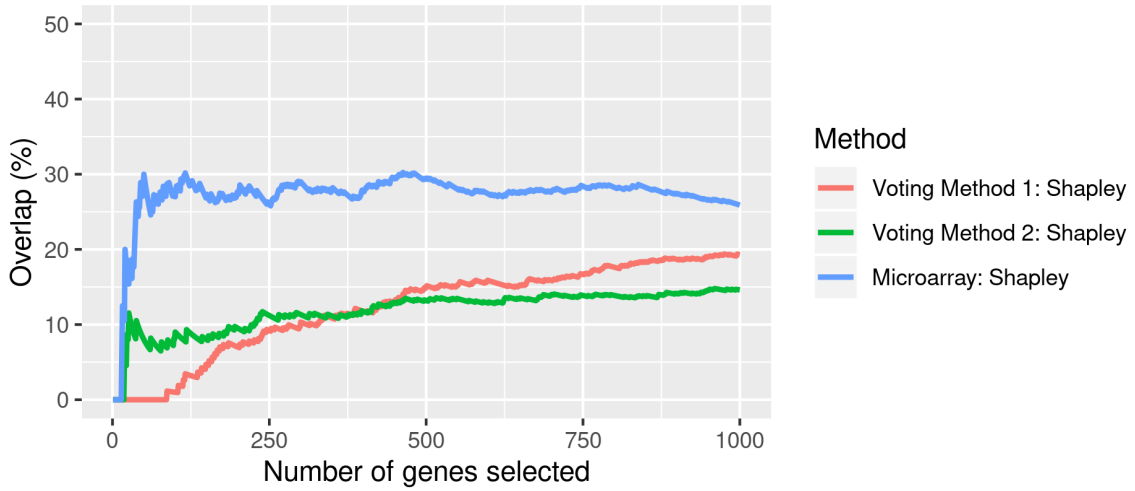


Figure 5.4: Air pollution - overlap of genes selected by t-test and other methods

This shows that Method 2 seems to select significantly different genes than t-test. In the later section, we will see that the predictive model based on that selection has a comparable accuracy.

5.6 Class prediction

For our prediction model, we use the standard logistic regression included in R. The model is created using the expressions of n genes with the highest estimated importance. We use several methods to select important genes and compare the accuracies of their respective models. As a baseline, we order genes by the p-value of t-test, performed on each gene separately. For comparison, we also show efficiency of selecting genes at random.

We test accuracies of logistic regression on log values of gene expressions with the genes selected by following methods.

- Shapley value of microarray games
- Additive voting games constructed using Method 1
- Additive voting games constructed using Method 2
- T-Test
- Randomly

5.6.1 Measuring the model accuracy

To measure the accuracy of a predictive model, we use leave-one-out cross validation. That is, for each sample i, we fit the model to all other samples and then let the resulting model predict the class of i. The resulting accuracy is the fraction of correct predictions out of all predictions.

5.6.2 Results of experiments

Figure 5.5: Breast cancer - testing accuracy of models

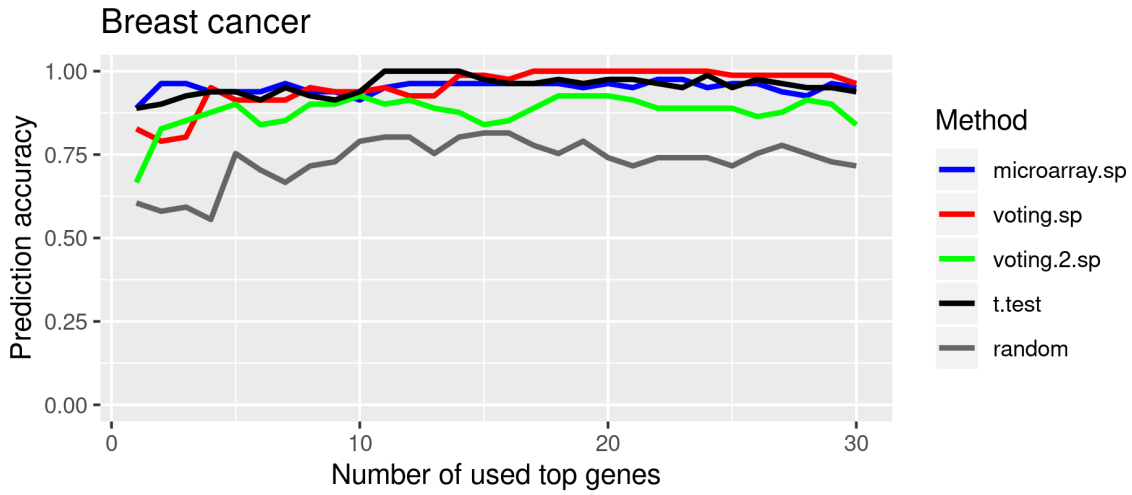


Figure 5.6: Lung cancer - testing accuracy of models

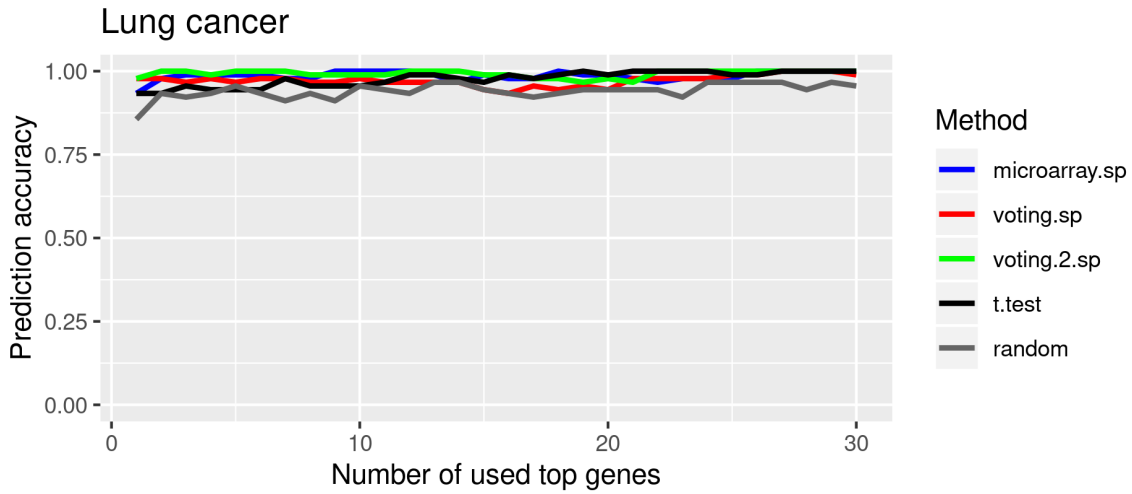


Figure 5.7: Schizophrenia - associative striatum - testing accuracy of models

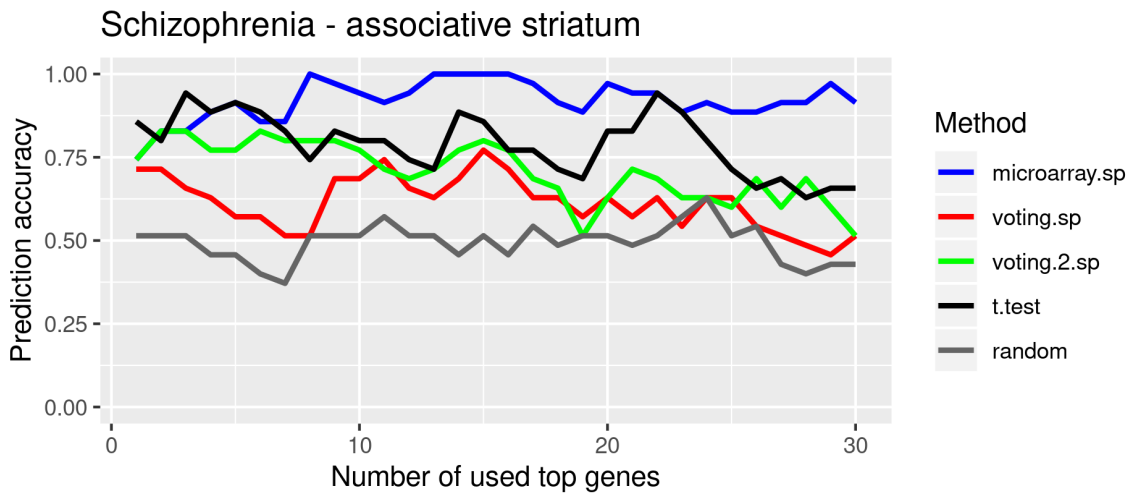


Figure 5.8: Schizophrenia - hippocampus - testing accuracy of models

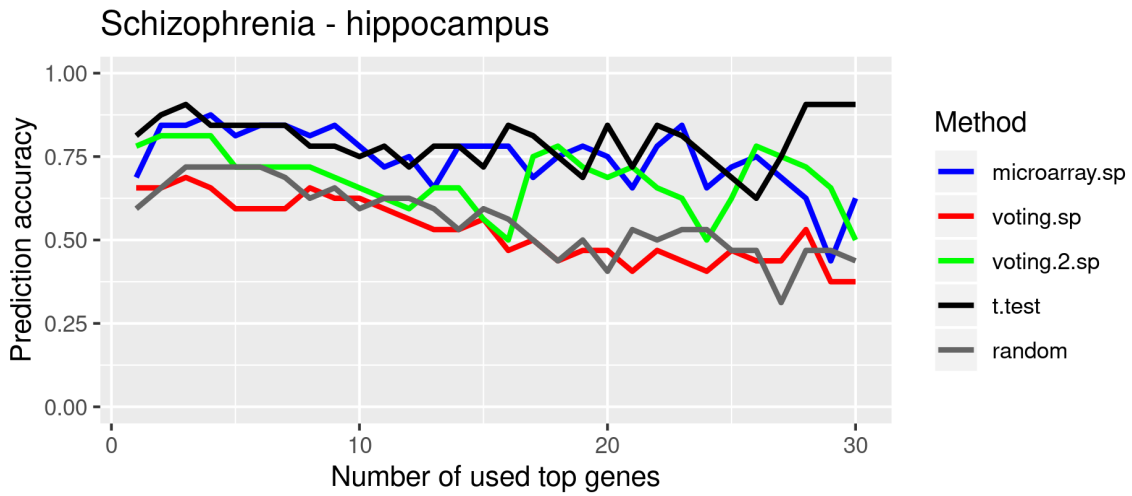


Figure 5.9: Schizophrenia - prefrontal cortex - testing accuracy of models

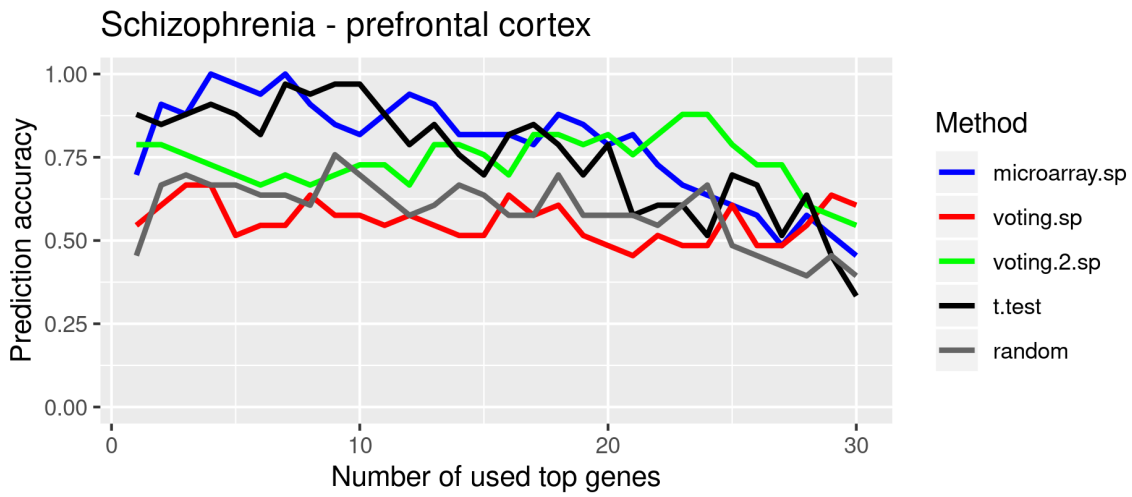
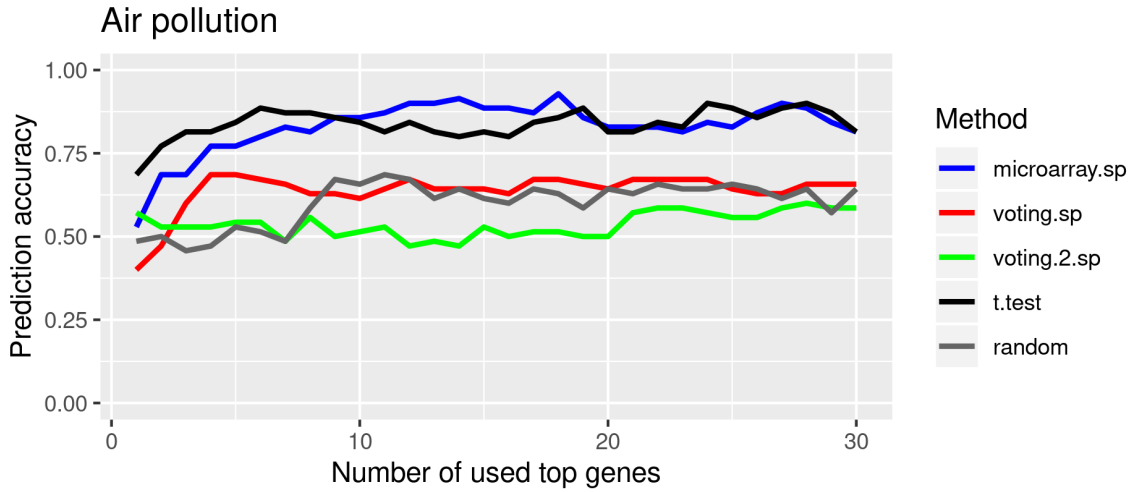


Figure 5.10: Air pollution - testing accuracy of models



5.6.3 Conclusion

First, consider the results in section 5.5. Note that Additive voting games with Method 2 selects a substantially different set of genes than other methods (including t-test) as seen in Figures 5.1, 5.2, 5.3 and 5.4. The effect is most notable in the breast cancer sample, where the overlap with genes selected by t-test does not exceed 5 percent. This shows that using the Shapley value of weighted voting games with the weights selected by the Method 2 provides a novel way of selecting important genes. Of course, this depends on the validity of the selection, which is shown in the next section.

Let us consider the results in Section 5.6.2. In the case of the breast cancer sample, we can see that Additive voting games with Method 1 achieves 95% testing accuracy with a very

low number of samples, as seen in Figure 5.5. In the lung cancer sample, Additive voting games with Method 2 achieves 100% testing accuracy with the lowest number of genes as seen in Figure 5.6. This makes it the most desirable predictive model out of those tested. Models using lower number of predictors are easier to interpret and are usually more robust in practice (less prone to overfitting).

In the case of the sample taken from the prefrontal cortex of schizophrenia patients, we still achieve very interesting results. The Voting 2 model has lower accuracy than other models, but it is still quite high as seen in Figure 5.7. Most importantly, it does so using a very different selection of genes, as seen in Figure 5.3.

Overall, our experiments show that our models provide a very useful as well as novel measure of relevance of genes.

Chapter 6

Conclusion

6.1 Future work

Weighted voting games constructed from gene expression data present a reasonable model of behaviour governed by the expression of genes. In this thesis, we focus on using power indices as indications of important genes. It would be interesting to use other parameters of weighted voting games to derive additional information. For example interaction indices as defined by Grabisch et al. [16] might reveal some information about interactions between genes.

Bibliography

[1] Debnath, M.; Prasad, G.; et al. *Microarray*. 10 2010, ISBN 9789048132604, pp. 193–208, doi:10.1007/978-90-481-3261-4_13.

[2] Behzadi, P.; Ranjbar, R. DNA microarray technology and bioinformatic web services. *Acta Microbiologica et Immunologica Hungarica AMicr*, volume 66, no. 1, 2019: pp. 19 – 30. Available from: <https://akjournals.com/view/journals/030/66/1/article-p19.xml>

[3] Li, S.; Teng, S.; et al. Microarray is an efficient tool for circRNA profiling. *Briefings in Bioinformatics*, volume 20, no. 4, 02 2018: pp. 1420–1433, ISSN 1477-4054, doi:10.1093/bib/bby006, <https://academic.oup.com/bib/article-pdf/20/4/1420/30119674/bby006.pdf>. Available from: <https://doi.org/10.1093/bib/bby006>

[4] Wang, Z.; Gerstein, M.; et al. RNA-Seq: a revolutionary tool for transcriptomics. *Nature reviews. Genetics*, volume 10, no. 1, Jan 2009: pp. 57–63, ISSN 1471-0064, doi:10.1038/nrg2484, 19015660[pmid]. Available from: <https://pubmed.ncbi.nlm.nih.gov/19015660>

[5] Moretti, S.; Patrone, F.; et al. The class of microarray games and the relevance index for genes. *Top*, 12 2007.

[6] Merlo, D.; Gmuender, H.; et al. Combining Shapley value and statistics to the analysis of gene expression data in children exposed to air pollution. *BMC Bioinformatics*, volume 9, 09 2008, doi:10.1186/1471-2105-9-361.

[7] Lucchetti, R.; Moretti, S.; et al. The Shapley and Banzhaf values in microarray games. *Computers & Operations Research*, volume 37, no. 8, 2010: pp. 1406 – 1412, ISSN 0305-0548, doi:https://doi.org/10.1016/j.cor.2009.02.020, operations Research and Data Mining in Biological Systems. Available from: <http://www.sciencedirect.com/science/article/pii/S0305054809000604>

[8] Moretti, S. Statistical analysis of the Shapley value for microarray games. *Computers & Operations Research*, volume 37, no. 8, 2010: pp. 1413 – 1418, ISSN 0305-0548, doi:https://doi.org/10.1016/j.cor.2009.02.016, operations Research and Data Mining in Biological Systems. Available from: <http://www.sciencedirect.com/science/article/pii/S0305054809000598>

[9] Gillies, D. B. Solutions to general non-zero-sum games. *Contributions to the Theory of Games*, volume 4, 1959: pp. 47–85.

[10] Shapley, L. S. Notes on the n-Person Game—II: The Value of an n-Person Game. *Research Memoranda*, 1951.

[11] The Prize in Economic Sciences 2012. 2012. Available from: <https://www.nobelprize.org/prizes/economic-sciences/2012/press-release/>

[12] Shapley, L. S. A value for n-person games. *Contributions to the Theory of Games*, volume 2, no. 28, 1953: pp. 307–317.

[13] Shapley, L. S.; Shubik, M. A Method for Evaluating the Distribution of Power in a Committee System. *The American Political Science Review*, volume 48, no. 3, 1954: pp. 787–792, ISSN 00030554, 15375943. Available from: <http://www.jstor.org/stable/1951053>

[14] Penrose, L. S. The Elementary Statistics of Majority Voting. *Journal of the Royal Statistical Society*, volume 109, no. 1, 1946: pp. 53–57, ISSN 09528385. Available from: <http://www.jstor.org/stable/2981392>

[15] Banzhaf III, J. F. Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev.*, volume 19, 1964: p. 317.

[16] Grabisch, M.; Roubens, M. An Axiomatic Approach to the Concept of Interaction among Players in Cooperative Games. *International Journal of Game Theory*, volume 28, 11 1999: pp. 547–565, doi:10.1007/s001820050125.

[17] Tomomi, M.; Yasuko, M. A Survey of Algorithms for Calculating Power Indices of Weighted Majority Games. *Journal of the Operations Research Society of Japan*, volume 43, 03 2000, doi:10.1016/S0453-4514(00)88752-9.

[18] Klinz, B.; Woeginger, G. J. Faster algorithms for computing power indices in weighted voting games. *Mathematical Social Sciences*, volume 49, no. 1, 2005: pp. 111 – 116, ISSN 0165-4896, doi:https://doi.org/10.1016/j.mathsocsci.2004.06.002. Available from: <http://www.sciencedirect.com/science/article/pii/S016548960400068X>

[19] Owen, G. Game theory: 4th Edition. 2013.

[20] Castro, J.; Gómez, D.; et al. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, volume 36, no. 5, 2009: pp. 1726 – 1730, ISSN 0305-0548, doi:https://doi.org/10.1016/j.cor.2008.04.004, selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X). Available from: <http://www.sciencedirect.com/science/article/pii/S0305054808000804>

[21] Fatima, S. S.; Wooldridge, M.; et al. A Randomized Method for the Shapley Value for the Voting Game. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '07, New York, NY, USA: Association for Computing Machinery, 2007, ISBN 9788190426275, doi:10.1145/1329125.1329316. Available from: <https://doi.org/10.1145/1329125.1329316>

[22] Fatima, S. S.; Wooldridge, M.; et al. A linear approximation method for the Shapley value. *Artificial Intelligence*, volume 172, no. 14, September 2008: pp. 1673–1699. Available from: <https://eprints.soton.ac.uk/265802/>

[23] Aziz, H.; Paterson, M. Computing voting power in easy weighted voting games. *CoRR*, volume abs/0811.2497, 2008, <0811.2497>. Available from: <http://arxiv.org/abs/0811.2497>

[24] Uno, T. Efficient Computation of Power Indices for Weighted Majority Games. In *Algorithms and Computation*, edited by K.-M. Chao; T.-s. Hsu; D.-T. Lee, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ISBN 978-3-642-35261-4, pp. 679–689.

[25] Bilbao, J. M.; Fernández, J. R.; et al. Generating functions for computing power indices efficiently. *Top*, volume 8, no. 2, Dec 2000: pp. 191–213, ISSN 1863-8279, doi:10.1007/BF02628555. Available from: <https://doi.org/10.1007/BF02628555>

[26] Bolus, S. Power indices of simple games and vector-weighted majority games by means of binary decision diagrams. *European Journal of Operational Research*, volume 210, no. 2, 2011: pp. 258 – 272, ISSN 0377-2217, doi:https://doi.org/10.1016/j.ejor.2010.09.020. Available from: <http://www.sciencedirect.com/science/article/pii/S0377221710006181>

[27] Chakravarty, N.; Goel, A. M.; et al. Easy weighted majority games. *Mathematical Social Sciences*, volume 40, no. 2, 2000: pp. 227 – 235, ISSN 0165-4896, doi:https://doi.org/10.1016/S0165-4896(99)00050-5. Available from: <http://www.sciencedirect.com/science/article/pii/S0165489699000505>

[28] Mareš, M.; Valla, T. *Průvodce labyrintem algoritmů*. CZ.NIC, z.s.p.o., 2017, ISBN 9788088168195. Available from: <http://pruvodce.ucw.cz/>

[29] Schönhage, A.; Strassen, V. Schnelle Multiplikation großer Zahlen. *Computing*, volume 7, no. 3, Sep 1971: pp. 281–292, ISSN 1436-5057, doi:10.1007/BF02242355. Available from: <https://doi.org/10.1007/BF02242355>

[30] Roche, D. S. Space-and time-efficient polynomial multiplication. In *Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, 2009, pp. 295–302.

[31] Lucas, W. F. *Measuring Power in Weighted Voting Systems*. New York, NY: Springer New York, 1983, ISBN 978-1-4612-5430-0, pp. 183–238, doi:10.1007/978-1-4612-5430-0_9. Available from: <https://doi.org/10.1007/978-1-4612-5430-0_9>

[32] Naghizadeh, M.; Sacchi, M. Multidimensional convolution via a 1D convolution algorithm. *The Leading Edge*, volume 28, 11 2009: pp. 1336–1337, doi:10.1190/1.3259611.

[33] Downey, R. G.; Fellows, M. R. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, volume 141, no. 1, 1995: pp. 109 – 131, ISSN 0304-3975, doi:https://doi.org/10.1016/0304-3975(94)00097-3. Available from: <http://www.sciencedirect.com/science/article/pii/0304397594000973>

[34] Cygan, M.; Fomin, F. V.; et al. *Parameterized algorithms*, volume 4. Springer, 2015, 421 to 424 pp.

[35] Barrett, T.; Wilhite, S. E.; et al. NCBI GEO: archive for functional genomics data sets–update. *Nucleic acids research*, volume 41, no. Database issue, Jan 2013: pp. D991–D995, ISSN 1362-4962, doi:10.1093/nar/gks1193, 23193258[pmid]. Available from: <https://pubmed.ncbi.nlm.nih.gov/23193258>

[36] LaBreche, H. G.; Nevins, J. R.; et al. Integrating Factor Analysis and a Transgenic Mouse Model to Reveal a Peripheral Blood Predictor of Breast Tumors. *BMC Medical Genomics*, volume 4, no. 1, Jul 2011: p. 61, ISSN 1755-8794, doi:10.1186/1755-8794-4-61. Available from: <https://doi.org/10.1186/1755-8794-4-61>

[37] Sanchez-Palencia, A.; Gomez-Morales, M.; et al. Gene expression profiling reveals novel biomarkers in nonsmall cell lung cancer. *International Journal of Cancer*, volume 129, no. 2, 2011: pp. 355–364, doi:10.1002/ijc.25704, <https://onlinelibrary.wiley.com/doi/pdf/10.1002/ijc.25704>. Available from: <https://onlinelibrary.wiley.com/doi/abs/10.1002/ijc.25704>

[38] Lanz, T. A.; Reinhart, V.; et al. Postmortem transcriptional profiling reveals widespread increase in inflammation in schizophrenia: a comparison of prefrontal cortex, striatum, and hippocampus among matched tetrads of controls with subjects diagnosed with schizophrenia, bipolar or major depressive disorder. *Translational Psychiatry*, volume 9, no. 1, May 2019: p. 151, ISSN 2158-3188, doi:10.1038/s41398-019-0492-8. Available from: <https://doi.org/10.1038/s41398-019-0492-8>

[39] Leeuwen, D.; Herwijnen, M.; et al. Genome-wide differential gene expression in children exposed to air pollution in the Czech Republic. *Mutation research*, volume 600, 09 2006: pp. 12–22, doi:10.1016/j.mrfmmm.2006.05.032.