A QOBDD-based Approach to Simple Games

Dissertation
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
(Dr.-Ing.)
der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel

vorgelegt von
Stefan Bolus

Kiel,
im Mai 2012

1. Gutachter:	Prof. Dr. Rudolf Berghammer
2. Gutachter:	Prof. Dr. Anand Srivastav
Tag der mündlichen Prüfung:	28.06.2012

To my beloved wife and children.

Abstract

Simple games are a commonly used model for the analysis of voting systems in which each participant can vote "yes" or "no" and where the outcome is "yes" or "no" as well. Quasi-reduced and ordered binary decision diagrams, or QOBDDs for short, are well-known as compact representations for subsets of powersets and Boolean functions. In this thesis, we use QOBDDs to represent simple games and to develop practically applicable algorithms to solve problems for simple games. We study properties of QOBDDs when they are used to represent simple games as well as the runtime behavior of our algorithms.

Votes are popular for making decisions as in the Council of the European Union. Unfortunately, voting systems are often complicated and it is a problem to verify that they reflect the original intentions. Because for some frequently used representations of simple games in practice even trivial problems are NP-hard already, it is indispensable to have powerful methods at hand that can deal with real world voting systems.

Different representations for simple games are used in practice like (multiple) weighted representations. Algorithms to solve problems are usually developed separately for each such representation. We abstract from these representations and use QOBDDs as an intermediate representation for simple games. We thereby exploit the fact that simple games are essentially up-sets and monotone Boolean functions, respectively. On the basis of this representation, we develop algorithms to solve some fundamental problems for simple games like the computation of a priori power indices. The step to obtain the QOBDD from a (multiple) weighted representation is answered separately.

If QOBDDs are used to represent simple games of a particular class, like those which possess a weighted representation, then they often exhibit structural features. We study these features and use them to establish upper bounds on the size of the QOBDDs as well as to develop specialized algorithms in some cases.

For our algorithms we introduce two novel and fundamental techniques for QOBDDs. On the one hand we present manipulators that can alter the set that is represented by a QOBDD without modifying the QOBDD itself. On the other hand we show how a counting problem for each player can be solved with only a constant number of traversals of the QOBDD. This includes the computation of the Chow parameters for the players which we use for some power indices.

We also study the problem whether a simple game, represented by a QOBDD, has a weighted representation. To this end we present a heuristic which can be used to decide if the simple game does not have such a weighted representation. Furthermore, based on the structure of the QOBDD we develop a linear program that can be used to decide if the simple game has a weighted representation. The resulting linear program has often significantly less non-zero coefficients than linear programs in the literature that are based on models of subsets of the winning and the losing coalitions, respectively.

Contents

Notation and Symbols

Let N be a finite set. For an element $x \in N$ and a subset S of N we usually write $S+x$ instead of $S \cup \{x\}$ and we write $S-x$ instead of $S \setminus \{x\}$. We declare that $+$ and $-$ have precedence over $\cup$, $\cap$ and $\setminus$.

A set $\mathcal{A} \subseteq 2^N$ is called an *up-set* (resp. *down-set*), if for each two coalitions $S, T \subseteq N$ with $S \subseteq T$ (resp. $T \subseteq S$) and $S \in \mathcal{A}$ it also holds $T \in \mathcal{A}$.

In some situations we assign a weight to the elements in N by a *weight function* $w : N \to \mathbb{R}$ and at the same time we define the weight of a subset of the elements $S \subseteq N$ by $\sum_{i \in S} w(i)$. By convention, the value of $w(S)$ is usually a set of weights. Deviating from that, as a convenience, we define $w(S)$ by $\sum_{i \in S} w(i)$ for a weight function w.

A *partition of* N is a list of disjoint sets whose union is N. A partition $P_1, \dots, P_n$ of N is a refinement of another partition $Q_1, \dots, Q_m$ of N, if $n \geq m$ and for each $i \in \{1, \dots, n\}$, there is a $j \in \{1, \dots, m\}$ with $P_i \subseteq Q_j$.

In the following list, the leftmost column refers to the first occurrence in the text.

Page	Notation	Description
	$\mathbb{B}$	Boolean truth values 0 (false) and 1 (true)
	$\mathbb{N}_0$, $\mathbb{N}$	natural number with and without 0, resp.
	$\mathcal{A}$, $\mathcal{B}$	sets of subsets
	$\chi_{\mathcal{A}}$	characteristic function of $\mathcal{A}$
	P, NP	complexity classes for polynomial and non-deterministic polynomial time
12	$(N, \mathcal{W})$	simple game with players N and winning coalitions $\mathcal{W}$
12	$\mathcal{L}$	set of losing coalitions (down-set)
13	$\min \mathcal{A}$, $\max \mathcal{A}$	minimal and, resp., maximal subsets in $\mathcal{A}$
13	$\mathcal{W}_{\min}$, $\mathcal{L}_{\max}$	minimal winning and maximal losing coalitions
94	$\mathcal{W}^d$	blocking coalitions in $\mathcal{W}$ (up-set)
13	$\succeq_I$	desirability relation on individuals (preorder on N)
13	$\approx_I$	relation of equally desirable players (equivalence relation on N)
14	$N_1, \dots, N_t$	types of players w.r.t. $\succeq_I$ and players N (partition of N)
14	$\mathcal{W}_{\text{shift}}$, $\mathcal{L}_{\text{shift}}$	shift-minimal winning and shift-maximal losing coalitions
15	$[Q; w]$	weighted representation with quota $Q \in \mathbb{N}_0$ and weight function $w : N \to \mathbb{N}_0$

15	$[Q; w_1, \ldots, w_n]$	weighted representation with quota $Q \in \mathbb{N}_0$ and weights $w_1, \ldots, w_n \in \mathbb{N}_0$
18	$\{g_1, \ldots, g_m\}$	m-vector-weighted representation with rules $g_1, \ldots, g_m$
19	$(\varphi, \{g_1, \ldots, g_m\})$	multiple weighted representation with formula φ and rules $g_1, \ldots, g_m$
21	$\mathbb{O}, \mathbb{I}$	BDD 0-sink and 1-sink, respectively
25	$\mathbb{O}_i, \mathbb{I}_i$	representatives of $\mathbb{O}$ and $\mathbb{I}$, resp., on level i
21	$\mathsf{V}(r), \mathsf{V}_i(r)$	inner nodes of the BDD with root r (and label i)
21	$\mathsf{then}(v), \mathsf{else}(v)$	1-successor and 0-successor of the BDD node v
21	$\mathsf{var}(v)$	label of the BDD node v
22	π-OBDD	OBDD with ordering π
23	$\mathsf{size}(r)$	number of inner nodes in the BDD with root r
23	$\mathsf{width}_i(r), \mathsf{width}(r)$	$\lvert\mathsf{V}_i(r)\rvert$ and $\max_i \lvert\mathsf{V}_i(r)\rvert$, resp.
24	$\mathsf{ite}(i, t, e)$	operation to create an inner node with label i and successors t, e
24	$\mathsf{set}(v)$	set represented by the QOBDD node v
25	$\overline{v}$	same as QOBDD with root v, but the sinks are interchanged
25	$\mathcal{A}_i, \mathcal{A}_{\neg i}$	the sets $\{S - i \mid i \in S,\ S \in \mathcal{A}\}$ and $\{S \in \mathcal{A} \mid i \notin S\}$, resp.
25	$\mathsf{node}(v, i, S)$	node that is reached on level i starting at node v by the path S
26	$u \xrightarrow{S} v$	node v is reached from node u by the path S
26	$\mathsf{paths}(u, v)$	set of all paths from u to v
30	$\mathsf{apply}(i, u, v, \otimes)$	binary synthesis for QOBDD nodes u, v with label i and binary function $\otimes$
30	$u \otimes v$	application of apply with $\otimes$ to QOBDD nodes u, v
38	$u_w(\mathcal{A})$	infimum of the weights of the sets in $\mathcal{A}$ w.r.t. weight function w
38	$l_w(\mathcal{A})$	supremum of the weights of the sets *not* in $\mathcal{A}$ w.r.t. w
38	$\Delta_w(\mathcal{A})$	difference between $u_w(\mathcal{A})$ and $l_w(\mathcal{A})$
38	$l_w(v), u_w(v), \Delta_w(v)$	defined by $l_w(\mathsf{set}(v))$, $u_w(\mathsf{set}(v))$ and $\Delta_w(\mathsf{set}(v))$, resp.
55	$u \subset v, u \subseteq v$	defined by $\mathsf{set}(u) \subset \mathsf{set}(v)$ and $\mathsf{set}(u) \subseteq \mathsf{set}(v)$, resp.
57	$\preceq_L$	Lapidot's desirability relation on coalitions
70	$\Phi = (\Phi_T, \Phi_E)$	manipulator Φ with successor functions Φ_T and Φ_E
70	id	identity manipulator defined by $\mathsf{id} := (\mathsf{then}, \mathsf{else})$
70	$\Phi \circ \Psi$	composition of manipulators
70	$\Phi(S)$	successors of the QOBDD nodes in S w.r.t. manipulator Φ
71	$\mathsf{apply}_2(u, \Phi, v, \Psi, \otimes)$	binary synthesis for QOBDD nodes u, v and binary function $\otimes$ w.r.t. manipulators Φ, Ψ
70	$\mathsf{set}_\Phi(v)$	set represented by the QOBDD node v w.r.t. manipulator Φ
72	$\mathsf{V}(r, \Phi), \mathsf{V}_i(r, \Phi)$	inner nodes of the QOBDD with root r (and label i) w.r.t. manipulator Φ

72	$\mathsf{size}(r, \Phi)$	number of inner nodes in the QOBDD with root r w.r.t. manipulator Φ, that is, $\lvert\mathsf{V}(r, \Phi)\rvert$
72	$\mathsf{width}_i(r, \Phi)$, $\mathsf{width}(r, \Phi)$	$\lvert\mathsf{V}_i(r, \Phi)\rvert$ and $\max_i \lvert\mathsf{V}_i(r, \Phi)\rvert$, resp.
72	$\mathsf{and}(u, \Phi, v, \Psi)$	short for $\mathsf{apply}_2(u, \Phi, v, \Psi, \wedge)$
72	$\mathsf{or}(u, \Phi, v, \Psi)$	short for $\mathsf{apply}_2(u, \Phi, v, \Psi, \vee)$
72	$\mathsf{minus}(u, \Phi, v, \Psi)$	short for $\mathsf{apply}_2(u, \Phi, v, \Psi, \not\Rightarrow)$
73	$\mathsf{forall}(u, \Phi, v, \Psi, \otimes)$	variant of the binary synthesis for QOBDD nodes u, v and binary function $\otimes$ w.r.t. manipulators Φ, Ψ that returns either true or false
73	$\mathsf{subseteq}(u, \Phi, v, \Psi)$	short for $\mathsf{forall}(u, \Phi, v, \Psi, \Rightarrow)$
73	$\mathsf{equal}(u, \Phi, v, \Psi)$	short for $\mathsf{forall}(u, \Phi, v, \Psi, \Leftrightarrow)$
74	compls	manipulator that interchanges the 1- and 0-successor of a node
74	$\mathsf{without}_i$	manipulator to select all sets that do not contain element i
76	remove_i, add_i	manipulator to filter and remove (resp. add) element i from (resp. to) all sets
82	$\mathsf{chow}_{\mathcal{A}}(i)$	Chow parameter for player i w.r.t. $\mathcal{A}$
83	$\mathcal{A} \times \mathcal{B}$	Cartesian product for sets $\mathcal{A}$ and $\mathcal{B}$
84	$\text{maxsize}(\mathcal{A})$	maximum cardinality of a set in $\mathcal{A}$
84	$\lVert\mathcal{A}\rVert$	vector for the number of sets in $\mathcal{A}$ by size
85	$\vec{u} \odot \vec{v}$	vector product for counting purposes
87	$\vec{u} \gg k$	right-shift operation for a vector and a scalar
88	$\mathsf{swings}_{\mathcal{A}}(i)$	swings for player i w.r.t. $\mathcal{A}$
106	$\mathsf{models}_{P_1,\ldots,P_d}(\mathcal{A})$	models of elements in $\mathcal{A}$ w.r.t. partition $P_1, \ldots, P_d$
107	$\mathsf{succ}(v, j)$	jth successor of node v

1. Introduction

Voting systems in which a decision is made by a set of voters with respect to a collection of predefined decision rules are ubiquitous today. They directly or indirectly influence important areas of our life, e.g., when a political decision is made on the European level. Even though, we can easily find other examples for those voting systems in our personal or professional environment, especially international organizations like the International Monetary Fund and political unions like the European Union (briefly EU) use complex and elusive voting systems with various decision rules. The aim of this thesis is to support the understanding and the analysis of such voting systems.

By a voting system (resp. voting game) we mean a system in which a proposal, e.g., a bill, is pitted against the status quo. Every voter (resp. player) can affirm the proposal by saying "yes" or it can reject the proposal by saying "no". Whether a proposal is accepted depends on the assembly of the accepting players. Because there are only two possible outcomes, namely, accepting or rejecting a proposal, we talk about binary decision rules.

A voting game with $n \geq 1$ players corresponds to the mathematical concept of a Boolean function $f : \mathbb{B}^n \to \mathbb{B}$ with n arguments. This model is equivalent to the more common model with a set of so-called winning coalitions. If N is a (finite) set of players, then a *coalition* is exactly the subset of players that accept a proposal by saying "yes". If a coalition can accept the proposal and therefore, can enforce a positive outcome, then it is called *winning.* The pair $(N, \mathcal{W})$ consisting of the players N and the set of winning coalitions $\mathcal{W} \subseteq 2^N$ is called a *simple game.* We thereby always assume that supersets of winning coalitions are winning as well, that is, if additional players accept a proposal then this will not render the outcome negative. Formally, the set of winning coalitions is an up-set.

An important special case are *weighted voting games* (WVGs). Here, every player has a non-negative integer *voting weight* and there is a non-negative integer *quota.* A coalition is winning in such a game exactly if the sum of the voting weights of its players meets or exceeds the quota. Examples include the German Bundesrat and the Electoral College to elect the President of the United States.

The expressive power of weighted voting games is limited. Therefore, it is common in practice to use multiple WVGs to create more complex voting games. A coalition then has to win in every WVG or in a combination of the WVGs to win in the composite voting game. Voting games with multiple decision rules are especially interesting, because on the one hand they are elusive and on the other hand we have to suspect that this problem is potentially exploited by some players during bargaining for the voting weights to gain an advantage. It is therefore crucial to have powerful methods at hand to analyze and assess voting games that are relevant in practice.

Country	Weight	Maj.	Popl.
Germany	29	1	170
United Kingdom	29	1	123
France	29	1	122
Italy	29	1	120
Spain	27	1	82
Poland	27	1	80
Romania	14	1	47
Netherlands	13	1	33
Greece	12	1	22
Czech Republic	12	1	21
Belgium	12	1	21
Hungary	12	1	21
Portugal	12	1	21
Sweden	10	1	18
Bulgaria	10	1	17
Austria	10	1	17
Slovak Republic	7	1	11
Denmark	7	1	11
Finland	7	1	11
Ireland	7	1	8
Lithuania	7	1	8
Latvia	4	1	5
Slovenia	4	1	4
Estonia	4	1	3
Cyprus	4	1	2
Luxembourg	4	1	1
Malta	3	1	1
Quota	255	14	620

Table 1.1.: Voting weights and population in the Council of the EU as defined in the Treaty of Nice. Every column corresponds to a WVG.

A good example on this is the Treaty of Nice, which currently governs the decision making process in the *Council of the European Union*. During bargaining for the voting weights there have been controversial opinions, which have been reinforced by numerous analysis in scientific publications. See, for instance, Leech (2002), Heinemann (2003) and Freixas (2004).

The council is part of the legislative branch of the European Union together with the parliament of the EU. Its members (resp. players) are representatives of the 27 member countries of the EU. Depending upon the issue the council votes by simple majority, qualified majority or unanimity. When qualified majority voting is used, a bill (proposed by the commission of the EU) passes, if it is supported by 50% of the member countries (14 out of 27) and by 74% of the voting weights (255 out of 345). Additionally, every member can demand that the bill has to be supported by 62% of the population of the EU (620 out of 1000). This voting game can be represented by three weighted voting games (Freixas 2004) as listed in Table 1.1.

The fact that voting systems can be complex and elusive becomes apparent, if we consider the alternate representation of the council in Table 1.2. The winning coalitions are the same. By this representation we can see that the outcome of the vote mainly depends on the negotiated voting weights. Furthermore, to additionally consider the population is beneficial for only four players, namely Germany, the United Kingdom, France and Italy. To obtain the representation in Table 1.2 we have used the methods developed in this thesis and integer linear programming.

The mathematical structure of a simple game and a weighted voting game is so general, that it naturally appears in various fields of research. In the preface of their monograph on simple games, Taylor and Zwicker (1999) state in this respect:

Country	Weight	Majority	Popl.
Germany	29	0	2
United Kingdom, France, Italy	29	0	1
Spain, Poland	27	0	0
Romania	14	0	0
Netherlands	13	0	0
Greece, Czech Republic, Belgium, Hungary, Portugal	12	1	0
Sweden, Bulgaria, Austria	10	1	0
Slovak Republic, Denmark, Finland, Ireland, Lithuania	7	1	0
Latvia, Slovenia, Estonia, Cyprus, Luxembourg	4	1	0
Malta	3	1	0
Quota	255	6	2

Table 1.2.: Alternate representation of the Council of the European Union as defined in the Treaty of Nice. Every column corresponds to a WVG.

> Few structures in mathematics arise in more contexts and lend themselves to more diverse interpretations than do hypergraphs or simple games.

Weighted voting games have been studied extensively in electrical engineering as linear separable Boolean functions and threshold functions, respectively.

The analysis of a voting game is motivated by several questions. The central question is mostly that of the "power" of a player. Here, we consider only that kind of power that allows a player to influence the outcome of a voting, which is called I-power. Furthermore, we will only consider the structure of the voting game and ignore any assumptions about the players and their interests. Hence, whether a coalition seems unlikely or not does not matter in our considerations. In the literature this approach is known as *a priori* voting power. See Felsenthal and Machover (2004) for an introduction and critics. A priori voting power is very often studied in the context of power indices. On the basis of the structure of the voting game, a power index assigns a numerical value to each player, which indicates the player's power with respect to a certain understanding of what influence means. Two of the most prominent representatives are the power indices by Banzhaf (1965) and by Shapley and Shubik (1954). The list of power indices proposed in the literature is much longer though. Some of them are presented in Section 6.7.

A weighted voting game instance consists of one voting weight for each player and the quota. From an algorithmic perspective even very fundamental problems on WVGs are NP-complete already. For instance, by a polynomial time reduction from the NP-complete partition problem one can show that the problem to decide if the player with minimum weight can influence any decision is NP-complete. The same holds for the computation of some power indices (Matsui and Matsui 2001) even though there are pseudo-polynomial algorithms (Matsui and Matsui 2000). Therefore, the analysis of voting games relies on methods, that are capable to handle real world voting games.

Current algorithmic approaches often rely on the representation of a voting game as a single weighted voting game and disregard other representations, for instance, voting games with multiple decision rules. In practice, however, these representations are

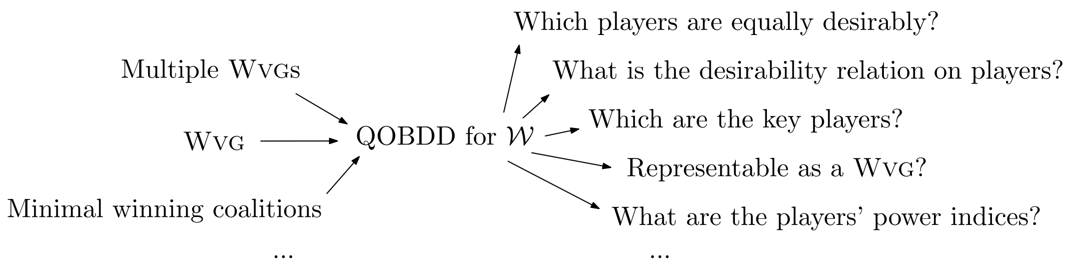

Multiple WvGs
Wvg
Minimal winning coalitions
...
QOBDD for $\mathcal{W}$
Which players are equally desirably?
What is the desirability relation on players?
Which are the key players?
Representable as a Wvg?
What are the players' power indices?
...

Figure 1.1.: Separation between the representation of voting games in practice (left) and algorithms to solve problems on simple games (right).

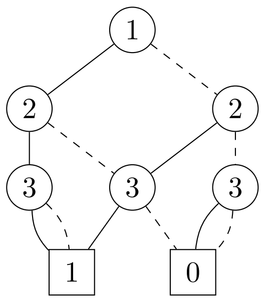
frequently used as in the Council of the EU. Those approaches that do consider other representations of voting games are often rigid and do only solve a single problem for a specific representation. As an example, Algaba, Bilbao, Fernández García, and López (2003) present algorithms to compute some power indices for voting games with multiple decision rules, but they do not consider other problems like the computation of the desirability relation on individuals.

In this thesis we will take a different approach. We will use an intermediate representation of the set of winning coalitions $\mathcal{W}$ of a simple game, that we can easily obtain from other representations of voting games in practice and that is often compact in size. Problems on simple games are then solved by algorithms that use the intermediate representation, so that there is a strict separation between the representation of voting games in practice on the one hand and the algorithms which solve specific problems on the other hand. Figure 1.1 illustrates the separation.

Compact representations of Boolean function and subsets of 2^N by their characteristic function, respectively, are essential in many areas like electrical engineering. Ordered binary decision diagrams and its variants are well-known for this purpose and they have been successfully applied to numerous problems. For instance, ordered binary decision diagrams have been used in the synthesis of Boolean circuits (Möller, Mohnke, and Weber 1993), for the representation of binary relations (Berghammer, Leoniuk, and Milanese 2002) and for multiple 0-1 knapsack problems (Behle 2008).

To be more precise, we will employ so-called quasi-reduced and ordered binary decision diagrams (QOBDDs). Figure 1.2 shows a QOBDD and the represented Boolean function with three variables in form of a truth table. QOBDDs are well-studied in theory (Wegener 2000) and with respect to implementations (Brace, Rudell, and Bryant 1990; Minato, Ishiura, and Yajima 1990). They are known to be able to represent many important Boolean functions and subsets of 2^N compactly. Our approach thereby borrows notions and methods from other research fields like Boolean function logic and threshold logic but also presents new results and new ideas for these fields. For instance, it is still a very important problem in threshold logic to identify weighted voting games and threshold functions, respectively (Smaus 2007; Palaniswamy, Goparaju, and Tragoudas 2010).

We will see that the use of QOBDDs as an intermediate representation has only little



variable	1	2	3	f-value
	0	0	0	0
	0	0	1	0
	0	1	0	0
	0	1	1	1
	1	0	0	0
	1	0	1	1
	1	1	0	1
	1	1	1	1

Figure 1.2.: A QOBDD for the Boolean function $f : \mathbb{B}^3 \to \mathbb{B}$.

negative effect on the time complexity of our algorithms in comparison to algorithms that use a specific representation like WVGs. To the contrary, the use of QOBDDs sometimes allows to obtain better upper bounds for the running time. For instance, this will be the case for computing some power indices when the QOBDD has been obtained from a WVG, because in this case the QOBDD has a particular structure. However, this is at the cost of a worse space complexity of our algorithms.

In this thesis we will solve various problems on simple games. As it will turn out, by the use of QOBDDs some algorithms can easily be derived from the logical formulations of the problems on simple games once a certain set of fundamental operations has been acquired. Hence, most algorithms in this paper can easily be implemented and this approach might therefore be interesting for other research fields as well. Exceptions are those few operations that operate directly on the structure of a QOBDD. To address this issue, in Section 6.1 we will present the idea of so-called manipulators that will reduce the number of such algorithms.

Our main motivation for the use of QOBDDs has been the use of reduced OBDDs (ROBDDs) to represent binary relations. By using relation algebraic methods for the solution of some problems in the context of simple games (Berghammer, Bolus, Rusinowska, and de Swart 2011) and the ROBDD-based tool RELVIEW (Behnke, Berghammer, Meyer, and Schneider 1998; Berghammer, Leoniuk, and Milanese 2002), we encountered that some problems on simple games can be solved more efficiently by using ROBDDs and, respectively, QOBDDs directly. RELVIEW offers the ability to compute results by relation algebraic specifications and programs. Therefore, the use of QOBDDs directly can be considered to be more complicated in general, because they operate on a lower level of abstraction in contrast to the mathematical model of binary relations. Nevertheless, in our case the benefits of using QOBDDs outweigh the drawbacks.

The research in this thesis has been conducted in the context of the European program LogICCC[1] and the project *SOCIAL SOFTWARE for elections, the allocation of tenders and coalition/alliance formation* (SSEAC) under the DFG grant BE 4206/1-1. As part of this project we have developed a software to analyze simple games, called the *Simple Game Laboratory*. The software is available as a user-friendly web application that only

[1]The program title is *Modelling intelligent interaction - Logic in the Humanities, Social and Computational sciences (LogICCC).*

requires a recent web browser to run. It is available online at:

http://sourceforge.net/projects/simple-game-lab/

The laboratory implements nearly all algorithms and ideas that are presented in this thesis. Moreover, it offers various examples and it is capable to visualize quasi-reduced and ordered binary decision diagrams. The real world examples that we use in this thesis refer to those in the laboratory. Even though most results in this thesis are theoretical in nature, the reader may benefit from using the laboratory from time to time. The author kindly remarks, however, that the implementation is not the most efficient one possible due to the commitment to a web application. The software has been written in ECMAScript which is better known as JavaScript.

Parts of this thesis are essentially based on already published articles by the author. A list of the exact references can be found in the introductory texts of the chapters.

The next section presents a short outline of the thesis and its results. The reader is also encouraged to use the index at the end of the thesis for reference.

Outline

This section presents an outline of the thesis and its results. Chapter 2 provides the necessary foundations in the field of cooperative game theory, voting theory and simple games. The main notions of a coalitional game, a simple game, a weighted voting game and a multiple weighted voting game are introduced and exemplified. Concepts such as the desirability relation on individuals, types and shift-minimal winning coalitions are introduced. Further notions are introduced throughout the thesis when they are first used. The index at the end of the thesis can be used for reference.

Chapter 3 complements Chapter 2 and presents the foundations of binary decision diagrams (BDDs). The connection between simple games and BDDs is established in Section 3.2. Section 3.3 discusses implementation issues that become relevant, when we do use BDDs differently than one would usually expect. For instance, the costs of having shared BDD nodes are discussed. The binary synthesis of QOBDDs is discussed in Section 3.4. Here, the focus is on the complexity of the synthesis for a given expression tree. This is necessary, because for multiple weighted and vector-weighted voting games the expression tree is part of the input.

The subject of Chapter 4 is the process of building the QOBDD for a simple game from a weighted representation and a multiple weighted representation with a formula, respectively. Section 4.1 presents the framework in which we study QOBDDs representing weighted voting games. This section also provides elementary results, that are used in Section 4.2 to motivate and to analyze the output sensitive algorithm by Behle (2008) to build the QOBDD representation of (the set of winning coalitions of) a weighted voting game. Because WVGs are the building blocks for more complex voting games in practice and even all simple games in theory, in Section 4.3 we will present the idea, how to use the binary synthesis of QOBDDs to build the QOBDD for an arbitrary simple game.

Hosaka, Takenaga, and Yajima (1994) have shown, that QOBDDs representing threshold functions with n variables (and therefore WVGs) have size at most $\mathcal{O}(2^{n/2})$. In Section 5.3 we will define a new class of so-called *flat* QOBDDs. For each label i, all the nodes of a flat QOBDD with label i constitute an order-theoretic chain with respect to the strict ordering $\supset$ on their represented sets. Based on that we will prove that the upper bound $\mathcal{O}(2^{n/2})$ does already hold for flat QOBDDs. We will see that being flat depends on the ordering of the players and that being flat is a much weaker condition than representing a WVG.

Beside the trivial upper bound of $\mathcal{O}(nQ)$ for the size of a QOBDD representing a weighted voting game with quota Q and n players, the just mentioned bound of $\mathcal{O}(2^{n/2})$ has been the lowest upper bound so far. In Section 5.3 we will establish an improved bound of $\mathcal{O}(\max\{n - \log Q, 1\}Q)$ for QOBDDs representing WVGs.

Homogeneous simple games have been studied early in the area of cooperative game theory (Morgenstern and von Neumann 1944). In Section 5.4 we will show that QOBDDs representing homogeneous simple games with n players have a size polynomial in n for at least two orderings of the players. In contrast, we will also present an example and an ordering of the players such that the corresponding QOBDD has size $\Omega(2^{n/2})$.

Chakravarty, Goel, and Sastry (2000) and Aziz and Paterson (2008) have presented some classes of weighted voting games, for which some problems on simple games can be solved in polynomial time in the number of players n. One such problem is the computation of the Banzhaf power index. In both papers the class of WVGs with sequential weights has been considered but neither of them has been able to find a polynomial time algorithm for the computation of the Banzhaf power index for these games. In Section 5.5 we will show that QOBDDs representing WVGs with sequential weights have polynomial size and, therefore, such an algorithm exists. Similar to homogeneous simple games we will also show that there is a WVG with sequential weights, whose QOBDD has size $\Omega(2^{n/2})$ for at least one ordering of the players.

Chapter 6 starts with an introduction of so-called manipulators in Section 6.1. Manipulators can be used to manipulate a traversal of a QOBDD without changing the QOBDD itself. While without manipulators even trivial changes in a set represented by a QOBDD can require a completely new QOBDD that has no inner node in common with the previous one, manipulators can solve this problem gracefully in many situations without creating additional nodes. The concept of manipulators is used throughout the chapter.

Section 6.2 presents elementary counting algorithms for QOBDDs. Given $n \in \mathbb{N}$ and a set $\mathcal{A} \subseteq 2^{\{1,\dots,n\}}$ we will show that if $\mathcal{A}$ is represented by a QOBDD, then, for instance, the values $|\{S \in \mathcal{A} \mid i \in S\}|$ for *all* $i \in \{1,\dots,n\}$ can be obtained by just two traversals of the QOBDD for $\mathcal{A}$, what intuitively requires n traversals. In the context of simple games we will use these values (and others) to compute power indices in Section 6.7. This result, however, does also have potential to be useful in the identification of symmetric Boolean variables in electrical engineering, even though the algorithms are less trivial in the context of reduced OBDDs. Section 6.2 also presents a solution to obtain the values $|\{S \in \mathcal{A} \mid i \in S, |S| = k\}|$ for all $k \in \{0,\dots,n\}$ and $i \in \{1,\dots,n\}$. Here, an additional factor of n in the running time is necessary.

The remaining sections in Chapter 6 present solutions to various problems on simple games. The problems include the computation of the desirability relation on individuals $\preceq_I$ in Section 6.3 and the test, if a simple game is complete, the computation of the dual of a simple game and the test, if a simple game is proper and strong, respectively, in Section 6.4, the computation of the QOBDD for the set of minimal winning and the maximal losing coalitions in Section 6.5, and based on that, the computation of the shift-minimal winning and shift-maximal losing coalitions in Section 6.6. Based on the counting algorithms in Section 6.2, Section 6.7 presents formulas to compute the Banzhaf, the Shapley-Shubik, the Holler-Packel, the Deegan-Packel and the Shift power indices. As one of the main results we will see, that the Banzhaf power indices for *all* players can be computed in time, linear in the size of the QOBDD that represents the set of winning coalitions. Because of the results in Sections 4.2 and 5.3 we obtain an algorithm with (deterministic) running time $\mathcal{O}(\max\{n - \log Q, 1\} Q \log Q)$ for a weighted voting game with n players and quota Q. This enhances the best known upper bound of $\mathcal{O}(nQ)$ (Uno 2003) for this problem. Section 6.8 presents an algorithm to obtain the so-called models for a QOBDD representing a set of subsets. An application of models is sketched in Chapter 7. A general advantage of our approach is, that if the QOBDD for a simple game is small, as it is the case for homogeneous simple games and WVGs with sequential weights, then the problems can be solved quickly.

The problem to decide, whether a simple game can (not) be represented by a weighted voting game, and if so, to find such a representation, is the subject of Chapter 7. The problem does also naturally arise in the context of Boolean functions and has been studied extensively in electrical engineering. Usually linear programming is used for this purpose and the linear program is built from a subset of the winning and losing coalitions or its models. In Section 7.1, however, we will see that we can build a linear program for the same purpose by using the structure of the QOBDD, that represents the winning coalitions of a simple game. In many relevant cases the resulting linear program has less non-zero coefficients and therefore can be solved faster. This statement is justified by experiments in Section 7.2. Section 7.3 presents a heuristic, based on flat QOBDDs to identify QOBDDs, that do not present weighted voting games. Random experiments are used to evaluate the heuristic. Section 7.4 presents a method to obtain a witness of not representing a WVG for a QOBDD, if the QOBDD is not flat. Chapter 7 closes with some concluding remarks in Section 7.5.

Finally, Chapter 8 presents concluding remarks, comments and ideas for future work.

Part I.

Basics

2. Simple Games

In this chapter we lay the foundations for most parts of this thesis. We introduce the notion of a simple game and many related concepts from the areas of cooperative game theory and voting systems. We start with simple coalitional games, better known as simple games, in Section 2.1 as a class of coalitional games. In Section 2.2 we introduce the important class of weighted voting games. This class is often used to model real world voting systems but it is not expressive enough in some cases. In these situations vector-weighted representations and multiple weighted representations with a formula are used which are introduced in Section 2.3. This brings us back to the class of simple games again, because each simple game has such a representation.

The introduction of some notions is postponed until they are needed in later chapters. For instance, homogeneous simple games are introduced in Section 5.4, power indices are introduced in Section 6.7 and models of coalitions are introduced in Section 6.8.

2.1. Simple Coalitional Games

Cooperative game theory is about the idea that individuals (e.g., players, companies, countries) can gain more when they work together instead of acting alone. Therefore, concepts such as solutions and payoffs are central in this discipline. This thesis has little to do with cooperative game theory in this respect. However, it is similar in others like the Shapley value as a solution concept and the Shapley-Shubik index (see Section 6.7) as a measure of "power". In this thesis we are interested in the analysis and the design of certain kinds of voting systems. From our perspective, a voting system is a mathematical function for which the decisions of the individuals (resp. players) are the function's inputs and the output is the decision. In our considerations a decision is either "yes" or "no". The ability to influence the outcome is known as the "power" of a player. Different definitions of this term exist as we will see in Section 6.7. Because we do only consider the mathematical function and ignore any interests or preferences of the players, we are concerned with *a priori voting power*; see Felsenthal and Machover (2004) for an introduction.

We start by introducing the notation that we borrow from the classical cooperative game theory.

Definition 2.1. Let $N \subseteq \mathbb{N}$ be a set of *players* and let $v : 2^N \to \mathbb{R}$ be a function. Then (N, v) is called a *coalitional game (with transferable utility).* $\square$ (N, v)

Many authors use the additional restriction that the empty coalition fulfills $v(\emptyset) = 0$. For $n \geq 1$ we will usually assume $N = \{1, \ldots, n\}$ as the set of players. The subsets of

2^N are called *coalitions*. For a coalition $S \subseteq N$ the value $v(S)$ is called the *worth of* S, for instance, $v(S)$ could be an amount of money. Usually, it is assumed that disjoint coalitions $S, T \subseteq N$ can earn more by working together, that is, $v(S \cup T) \geq v(S) + v(T)$ what is called *superadditivity.* Cooperative game theory is mainly about studying the distribution of worth of a coalition among its players. Numerous solution concepts like the core, the kernel and the nucleolus have been developed to this end. See Peleg and Sudhölter (2007) for an introduction and an overview.

Let (N, v) be a coalitional game. By restricting the range of v from $\mathbb{R}$ to $\mathbb{B}$ and additionally imposing $v(S) \leq v(T)$ for $S \subseteq T \subseteq N$ we obtain the class of simple games. For a thorough introduction to the theory of simple games see the textbook by Taylor and Zwicker (1999).

$(N, \mathcal{W})$

Definition 2.2. For players $N \subseteq \mathbb{N}$ such that $n := |N| \geq 1$ and $\mathcal{W} \subseteq 2^N$ the pair $(N, \mathcal{W})$ is called a *(n-person) simple game* if $\mathcal{W}$ is an up-set. □

Most authors exclude the cases $\mathcal{W} = \emptyset$ and $\mathcal{W} = 2^N$, because they do not have meaningful counterparts in the real world. We include these cases for technical reasons and highlight possible conflicts with results in the literature on simple games.

$\mathcal{L}$

A coalition in $\mathcal{W}$ is called *winning* while a coalition in the complement $2^N \setminus \mathcal{W}$ is called *losing*. The losing coalitions are denoted by $\mathcal{L} := 2^N \setminus \mathcal{W}$. For a simple game $(N, \mathcal{W})$ the associated cooperative game is $(N, \chi_{\mathcal{W}})$ where $\chi_{\mathcal{W}} : 2^N \to \{0, 1\}$ is the characteristic function of $\mathcal{W}$.

As has been mentioned by Taylor and Zwicker (1999), simple games cannot be studied in isolation in the field of cooperative game theory or voting theory. The structure of a simple game naturally appears in many other disciplines as monotone hypergraphs or positive Boolean functions and even though each discipline has its own motivation and goals, many results are useful in the others. In this thesis we emphasize the connection to the area of electronic engineering and threshold logic that studies positive Boolean functions and threshold functions. For that reason, we will sometimes refer to literature in these areas even though the reader might not be familiar with either discipline.

In the remainder of this section, $(N, \mathcal{W})$ is a simple game with players N, winning coalitions $\mathcal{W}$ and losing coalitions $\mathcal{L}$. The following notions are important in the analysis of simple games. Algorithms are presented in Chapter 6.

Definition 2.3. A simple game $(N, \mathcal{W})$ is called *proper* if the complement of each winning coalition is losing and it is called *strong*[1] if the complement of each losing coalition is winning. It is called *constant-sum* or *decisive* if it is both proper and strong, and it is called *dual-comparable* if it is either proper or strong. □

We illustrate the notions that we have introduced so far.

Example 2.1. We consider players $N = \{1, 2, 3, 4\}$. As a convenience we will use letters instead of numbers in our examples. Hence, A denotes player 1, B denotes player 2 and

[1]Definitions of strong simple games are inconsistent. For instance, in Peleg and Sudhölter (2007) a strong simple game fulfills $S \in \mathcal{W}$ if and only if $N \setminus S \in \mathcal{W}$ for all $S \in 2^N$.

so on. Furthermore, we shall write, for instance, AB instead of $\{A, B\}$ to promote the presentation. Let

$$\mathcal{W} = \{AB, ABC, ABD, ACD, ABCD\}$$

be the set of the winning coalitions. Then the losing coalitions are:

$$\mathcal{L} = \{\emptyset, A, B, C, D, AC, AD, BC, BD, CD, BCD\}\,.$$

By inspecting the complement of every winning coalition in $\mathcal{W}$, we can see that the simple game $(N, \mathcal{W})$ is proper. It is not strong, though, because both coalitions AC and BD and their complements are losing. Hence, the simple game $(N, \mathcal{W})$ is dual comparable, but it is not decisive. □

Because it is reasonable not to have two winning coalitions at the same time, real world voting systems are at least proper. Many of them, however, fail to be strong and therefore to be decisive. Examples include the US Electoral College (2004-2008, 2012-2020) and the Council of the EU under the Treaty of Nice and the Treaty of Lisbon.

For $\mathcal{A} \subseteq 2^N$ by $\min \mathcal{A}$ (resp. $\max \mathcal{A}$) we denote the subset of minimal (resp. maximal) elements of $\mathcal{A}$ with respect to set inclusion. $\min \mathcal{A}$ $\max \mathcal{A}$

Definition 2.4. A coalition in $\min \mathcal{W}$ is called *minimal winning* while a coalition in $\max \mathcal{L}$ is called *maximal losing*. The sets of the minimal winning and maximal losing coalitions are denoted by $\mathcal{W}_{\min}$ and $\mathcal{L}_{\max}$, respectively. □ $\mathcal{W}_{\min}$ $\mathcal{L}_{\max}$

The idea is illustrated in the following example.

Example 2.2. We consider the simple game $(N, \mathcal{W})$ from Example 2.1 again. Because $ABCD$ is winning, but also AB is winning, $ABCD$ is not minimal winning. The coalition AB is minimal winning, because both A and B are losing. The corresponding sets are $\mathcal{W}_{\min} = \{AB, ACD\}$ and $\mathcal{L}_{\max} = \{AC, AD, BCD\}$. □

In voting systems some players are usually more "powerful" than others. One relation in this respect is the so-called *desirability relation on individuals* (Isbell 1958).

Definition 2.5. Given players $i, j \in N$, player j is called *at least as desirable as* i, denoted by $j \succeq_I i$, if $j \succeq_I i$

$$\forall S \subseteq N \setminus \{i, j\} : (S + i \in \mathcal{W} \implies S + j \in \mathcal{W})\,.$$

If $j \succeq_I i$ then player i is called *at most as desirable as* j. The strict part of the relation is denoted by $\succ_I$. We use $i \preceq_I j$ and $i \prec_I j$ for $j \succeq_I i$ and $j \succ_I i$, respectively. Players i, j are called *equally desirable*, denoted by $i \approx_I j$, if both $j \succeq_I i$ and $i \succeq_I j$. □ $j \succ_I i$ $i \approx_I j$

Intuitively, it holds $j \succeq_I i$ if in any winning coalition containing i but not j, player i can be replaced by j and the resulting coalition is still winning.

The relation $\succeq_I$ is a preorder and therefore reflexive and transitive (Isbell 1958). If $\succeq_I$ is total, i.e., each pair of players is comparable w.r.t. $\succeq_I$, then the simple game is called *complete*. A complete simple game is called *directed* if

$$\forall i, j \in N : (i \succ_I j \implies i < j)\,. \tag{2.1}$$

If $N = \{1, \ldots, n\}$ where $n = |N|$ then (2.1) is equivalent to $1 \succeq_I \cdots \succeq_I n$. Notice that the players in every complete simple game can be renamed, so that the resulting simple game is directed.

Example 2.3. We consider the simple game in Example 2.1 with players A, B, C and D again. This game's desirability relation on individuals $\succeq_I \subseteq N \times N$ corresponds to

$$A \succeq_I B \succeq_I C \succeq_I D \quad \text{and} \quad D \succeq_I C$$

and $i \succeq_I i$ for all players $i \in N$. It also holds $A \succ_I B$, $B \succ_I C$ and $C \approx_I D$. For instance, because for all coalitions $S \in \{\emptyset, C, D, CD\} \subseteq 2^N$ not containing A and B it holds $S \cup \{A\}$ is winning whenever $S \cup \{B\}$ is winning, it holds $A \succeq_I B$. Because for $S = CD$ the coalition $S \cup \{A\}$ is winning but $S \cup \{B\}$ is not winning, it holds $A \succ_I B$. The game is directed and therefore complete. □

$N_1, \ldots, N_t$
t

The relation $\approx_I$ is an equivalence relation. Its equivalence classes are called *types*. The types are denoted by $N_1, \ldots, N_t$ and they are a partition of N where t is the number of types. W.l.o.g. we will assume that the types are ordered such that whenever $i \succ_I j$ and $i \in N_x$ and $j \in N_y$ it holds $x < y$. For instance, N_1 contains the most desirable players for a complete simple game.

A simple game can solely be represented by its minimal winning (resp. maximal losing) coalitions, because every winning (resp. losing) coalition can be derived by the up-set (resp. down-set) property. In other words it holds

$$\mathcal{W} = \{T \in 2^N \mid \exists S \in \mathcal{W}_{\min} : S \subseteq T\}$$

and analogously,

$$\mathcal{L} = \{T \in 2^N \mid \exists S \in \mathcal{L}_{\max} : T \subseteq S\}.$$

Because the set $\mathcal{W}_{\min}$ is usually much smaller than $\mathcal{W}$, the representation of a simple game by $\mathcal{W}_{\min}$ is more compact. Such a representation, however, can let problems on simple games become NP-hard. Freixas, Molinero, Olsen, and Serna (2012) have shown that for the set $\mathcal{W}_{\min}$ as the input, the problem to decide whether the associated simple game is strong, is NP-complete. For $\mathcal{W}$ as the input, however, the problem can be solved in polynomial time in $|N|$ and $|\mathcal{W}|$.

The desirability relation on individuals can be used to obtain a subset of $\mathcal{W}_{\min}$ (resp. $\mathcal{L}_{\max}$) that represents a simple game already. This representation is usually more compact, especially if the simple game is complete.

Let $S, T \in 2^N$ be coalitions and $i, j \in N$ be players such that $i \in S$, $j \notin S$ and $i \notin T$, $j \in T$ and which do not differ in the remaining players, that is, $S \setminus \{i\} = T \setminus \{j\}$. If $i \succ_I j$ then S is winning if T is winning. Thus, the information if S is winning, can be derived from the information whether T is winning and $i \succ_I j$. The replacement of player j by player i is sometimes called a *shift*[2].

[2]The wording becomes clear from the representation of coalitions in directed simple games as Boolean vectors $(x_1, \ldots, x_n) \in \mathbb{B}^n$. Replacing a player means to shift a 1 in the vector to the left or to the right.

Definition 2.6. A minimal winning coalition $S \in \mathcal{W}_{\min}$ is called *shift-minimal winning*, if for players $i, j \in N$ with $i \in S$, $j \notin S$ and $i \succ_I j$ it holds $(S - i) + j \notin \mathcal{W}$. Dually, a coalition $S \in \mathcal{L}_{\max}$ is called *shift-maximal losing* if

$$\forall i, j \in N : (i \in S \land j \notin S \land j \succ_I i \implies (S - i) + j \notin \mathcal{L}) \,.$$

The sets of the shift-minimal winning and shift-maximal losing coalitions for $(N, \mathcal{W})$ are denoted by $\mathcal{W}_{\text{shift}}$ and $\mathcal{L}_{\text{shift}}$, respectively. □ $\mathcal{W}_{\text{shift}}$, $\mathcal{L}_{\text{shift}}$

Intuitively, a coalition is shift-minimal winning if no player inside the coalition can be replaced by a less desirable player outside the coalition such that it remains winning.

	$\lvert\mathcal{W}\rvert$	$\lvert\mathcal{W}_{\min}\rvert$	$\lvert\mathcal{W}_{\text{shift}}\rvert$
UN Security Council	848	210	210
Canadian Constitution (1995)	167	112	56
Treaty of Nice	2 718 752	561 820	117 055
Treaty of Lisbon	17 196 173	4 016 553	53 764
US Electoral College (2004-2008)	$\approx 1.117 \cdot 10^{15}$	$\approx 51.199 \cdot 10^{12}$	$\approx 17.054 \cdot 10^{12}$

Table 2.1.: Comparison of the representation size of some real world simple games.

The number of winning, minimal winning and shift-minimal winning coalitions for some real world simple games is compared in Table 2.1. We have obtained the numbers using the Simple Game Laboratory[3].

Even the set $\mathcal{W}_{\text{shift}}$ can be further reduced by using so-called models of coalitions. They are formally introduced in Section 6.8 due to technical reasons. The idea, however, is very simple for the set $\mathcal{W}$. Consider the types $N_1, \ldots, N_t$ of $(N, \mathcal{W})$. Any winning coalition $S \in \mathcal{W}$ can be represented by a vector $\vec{m} = (m_1, \ldots, m_t) \in \mathbb{N}_0^t$ such that

$$\forall k \in \{1, \ldots, t\} : m_k = |N_k \cap S| \,.$$

The vector $\vec{m}$ is then called a model of S. Obviously, when $|N_k| > 1$ for some k, then this representation is more compact. Carreras and Freixas (1996) use the models of shift-minimal winning coalitions to characterize complete simple games.

2.2. Weighted Voting Games

A very common paradigm in the design of voting systems is to assign weights to the players and to set a quota, which has to be reached by the sum of weights of the players in a winning coalition. This concept is usually known as *weighted voting*. The restriction to integer weights and an integer quota in the introduction has been for illustration purposes. Here, we do allow real weights and a real quota, even though in most parts of the thesis we will use integers exclusively. The only exception is Chapter 7.

[3]See `http://sourceforge.net/projects/simple-game-lab/`.

Definition 2.7. Let N be a set of players, let $Q \in \mathbb{R}_{\geq 0}$ be a *quota* and let $w : N \to \mathbb{R}_{\geq 0}$ be a *weight function*. For $i \in N$ the value $w(i)$ is called the *weight of player i*. We call the pair $[Q; w]$ a *real weighted representation* and for a simple game $(N, \mathcal{W})$ we say $[Q; w]$ *represents* $(N, \mathcal{W})$ if [$[Q; w]$]

$$\forall S \in 2^N : (S \in \mathcal{W} \iff w(S) \geq Q)$$

where $w(S) := \sum_{i \in S} w(i)$ is the *weight of the coalition* $S \in 2^N$. If the quota Q, and the weights $w(i)$, $i \in N$, are elements from $\mathbb{N}_0$, then we call $[Q; w]$ an *(integer) weighted representation*. We call $(N, \mathcal{W})$ *weighted* or a *weighted voting game* (WVG) if it has a real weighted representation. In case $N = \{1, \dots, n\}$ for some $n \geq 1$ we usually use a *vector of weights* $w_1, \dots, w_n$ instead of a weight function and write $[Q; w_1, \dots, w_n]$. The weight function in this case is implicitly defined by $w(i) = w_i$ for player $i \in N$. □ [$w(S)$] [$[Q; w_1, \dots, w_n]$]

One might expect that there is a simple game, that has a real weighted representation but which does not have an integer weighted representation. This is not the case. Because the rational numbers are dense in $\mathbb{R}$, being a weighted voting game and having an integer weighted representation is equivalent (Hu 1965). Another proof can be found in Freixas and Molinero (2008, Proposition 2.2). Bohossian (1998) presents a sufficient condition for the restriction of the sets of weights for a weighted voting game, that also covers the case of integers.

Theorem 2.1 (Theorem 2.2 in Bohossian (1998))**.** *Let $D = \{d_1, d_2, \dots\} \subseteq \mathbb{R}$ be an ordered set with $d_1 < d_2 < \cdots$. If for any given arbitrarily large positive constant $C \in \mathbb{R}$ there exists i_0 such that for any $i \geq i_0$ it holds $(d_{i+1} - d_i)C < d_i$ then being a weighted voting game is equivalent to having a real weighted representation $[Q; w_1, \dots, w_n]$ such that $Q, w_1, \dots, w_n \in D$.* □

If not explicitly stated otherwise, in the following we will always use integer weighted representations. Moreover, we will use (integer) weighted representations and weighted voting games interchangeably. For instance, by $[Q; w]$ we refer to the simple game $(N, \mathcal{W})$ with players N and winning coalitions $\mathcal{W} = \{S \in 2^N \mid w(S) \geq Q\}$. This is valid, because the set $\mathcal{W}$ is an up-set. The simple game $(N, \mathcal{W})$ is called *associated with* $[Q; w]$.

The term weighted voting game is very specific to the areas of cooperative game theory and voting systems. Other areas such as electronic engineering or neural logic prefer the names[4] *threshold function* and *linearly separable function*.

Many aspects of weighted voting games have been studied in the different disciplines. Beside the analysis of real world voting systems, one of the most interesting problems

[4]A nice quota about this fact can be found in Akers and Rutter (1964):

> *An unofficial contest appears to be underway to determine how many new terms may be successfully applied to threshold functions. Current favorites include linearly separable functions, majority decision functions, linear decision functions, linear-input functions, vote-taking functions, and weighted decision functions with their popularity decreasing in about the same order.*

in the design of voting systems is the problem to find a weighted voting game that fits a given distribution of "power", for instance, by a power index. The problem is therefore sometimes called the *inverse power index problem* (Kurz and Napel 2012) or *inverse Banzhaf problem* (Alon and Edelman 2010), which refers to the particular Banzhaf power index, that is introduced in Section 6.7.

One of the most important problems in various areas is the identification of weighted voting games and threshold functions, respectively. Here, a simple game in some representation is given as the input and the question is, if there is a weighted representation for it and, if so, to find one. In the field of electronic engineering, this problem has been studied extensively in the 1960s (Winder 1962; Hu 1965; Lewis and Coates 1967; Sheng 1969). The main driver of this research was, and still is, the idea that we can build more efficient switching circuits by using more complex gates, such as gates that implement a threshold function instead of gates such as AND and NOR. In the area of the design of voting systems, finding weighted representations is interesting for games for which no weighted representation is known already, and for WVGs that might have a weighted representation with, e.g., less quota, or which is minimal in some other respect; see, for instance, Freixas and Molinero (2008) and the following example.

Example 2.4. The German parliament (lower house), called the *Bundestag*, currently has 620 members and five factions, where the number in parenthesis is the current number of seats: CDU/CSU (237), SPD (146), the liberal party (93), the left party (76) and the green party (68). In some situations it requires an absolute majority of 311 votes for a bill to pass in the Bundestag which is called a *Kanzlermehrheit*. This voting game can be represented by the WVG $[311; 237, 146, 93, 76, 68]$ if every faction votes en bloc. However, this game does also have $[3; 2, 1, 1, 1, 0]$ as a weighted representation. From the latter we can immediately see the structure of the game. □

Weighted voting games are much less expressive than general simple games and even complete simple games. There is no closed formula for the number of simple games (resp. monotone Boolean functions) yet, but for $n = 8$ players there are[5] more than $5.61 \cdot 10^{22}$ different simple games already. Even though, the number of complete simple games (resp. regular Boolean functions) is much smaller, for $n = 8$ there are[6] 16,175,188+2 complete simple games after all. The "+2" takes into account the cases $\mathcal{W} = \emptyset$ and $\mathcal{W} = 2^N$. Kurz and Tautenhahn (2012) use cliques to enumerate complete simple games. In comparison, the number of weighted voting games is less than 2^{n^2} and for 8 players there are[7] "just" 2,730,164+2 weighted voting games (Kurz 2011). For a discussion of upper and lower bounds for the number of threshold functions, see Bohossian (1998, Section 2.3.1).

Weighted voting games are sometimes inconvenient and sometimes insufficient in the design of voting systems. In the next section we will use WVGs as the building blocks of so-called vector-weighted representations and multiple weighted representations with

[5]This is sequence A000372 in the *On-Line Encyclopedia of Integer Sequences* (OEIS) (OEIS Foundation Inc. 2012).

[6]This is sequence A132183 in the OEIS.

[7]This is sequence A000617 in the OEIS.

a formula. These will lead back to the class of general simple games again, because each simple game has such a representation.

2.3. Multiple Weighted Voting Games

Weighted voting games are sometimes inconvenient and they are sometimes insufficient to represent a simple game. Because each Wvg is complete, Wvgs are unable to represent simple games which are not complete. But even complete simple games in the real world often do not have a weighted representation. Examples include the Council of the European Union under the Treaty of Nice and the Treaty of Lisbon.

An example, when a Wvg is inconvenient, is the Security Council of the United Nations (UN), which has the weighted representation (Taylor and Zwicker 1999, Section 1.2)

$$[39; \underbrace{7, \dots, 7}_{\text{5-times}}, \underbrace{1, \dots, 1}_{\text{10-times}}] \,. \tag{2.2}$$

The UN Security Council has 15 members. Five of them are permanent members, each of which can veto a decision, and the remaining ten members are elected every two years. A resolution requires nine affirmative votes to pass and no veto member must reject it. The possibility of abstention is ignored here. The game can therefore also be modeled by two *rules*, each of which is a Wvg:

$$[5; \underbrace{1, \dots, 1}_{\text{5-times}}, \underbrace{0, \dots, 0}_{\text{10-times}}] \quad \text{and} \quad [9; \underbrace{1, \dots, 1}_{\text{15-times}}] \,. \tag{2.3}$$

The rule on the left models the veto power of the permanent members while the rule on the right models the necessary nine votes of all members. A coalition has to win in both Wvgs in (2.3). In comparison to (2.2), the representation is self-explaining and hence, preferable.

Representations of simple games in which a coalition has to win in multiple rules simultaneously are well-known (Taylor and Zwicker 1999, Section 1.7).

$\{g_1, \dots, g_m\}$

Definition 2.8. Let N be a set of players and for $m \geq 1$ let $g_1, \dots, g_m$ be Wvgs with $g_k = [Q_k; w_k]$ where $k = 1, \dots, m$. The set $\{g_1, \dots, g_m\}$ is called a *m-vector-weighted representation* or *vector-weighted representation with m rules*. For a simple game $(N, \mathcal{W})$ we say $\{g_1, \dots, g_m\}$ *represents* $(N, \mathcal{W})$ if

$$\forall S \in 2^N : (S \in \mathcal{W} \iff \forall k \in \{1, \dots, m\} : w_k(S) \geq Q_k) \,.$$

The game g_k is called the *kth rule*. □

The name originates from the fact that m-vector-weighted representations can also be defined like weighted representations with vectors from $\mathbb{N}_0^m$ for the weights and the quota instead of scalars. As for Wvgs, we will use m-vector-weighted representations and simple games interchangeably without any risk of confusion. The simple game $(N, \mathcal{W})$ in the following statement is called *associated with* $\{g_1, \dots, g_m\}$.

Proposition 2.2. *For a set of players N and a m-vector-weighted representation with rules $g_1, \dots, g_m$, the pair $(N, \mathcal{W})$ is a simple game where*

$$\mathcal{W} = \{S \in 2^N \mid \forall k \in \{1, \dots, m\} : w_k(S) \geq Q_k\} .$$

Proof. It is rather easy to see that $\mathcal{W}$ is an up-set. $\square$

It is well-known that every simple game $(N, \mathcal{W})$ has a vector-weighted representation with at most $|\mathcal{L}_{\max}|$ rules (Taylor and Zwicker 1993, Section 2). How many rules are necessary for a vector-weighted representation is usually not clear though.

Definition 2.9. The *dimension* $d \in \mathbb{N}$ of a simple game $(N, \mathcal{W})$ is the minimum number such that there is d-vector-weighted representation for $(N, \mathcal{W})$. $\square$

A Wvg has dimension 1. For a m-vector-weighted representation, the problem to decide if there is a vector-weighted representation with less rules, is NP-complete in general (Deineke and Woeginger 2006). A simple game for which there is a m-vector-weighted representation, is sometimes referred to as *m-vector-weighted voting game.*

A m-vector-weighted voting game is the conjunction of m rules. As for Wvgs, this is sometimes inconvenient. For example, there is a situation in the Council of the EU[8] under the Treaty of Lisbon where a coalition has to satisfy the following conditions:

1. It represents 55% of the countries (15 out of 27) and
2. it represents 65% of the population or
3. a blocking minority would otherwise have only three or less members.

Disregarding the exact rules, this game contains a disjunction, because a coalition either has to fulfill the first two rules *or* it has to contain 24 members. The following notion of a multiple weighted voting game with a formula allows to conveniently model this situation. A very similar approach named *Boolean Weighted Voting Games* has been studied quiet recently by Faliszewski, Elkind, and Wooldridge (2009) for modeling decision-making processes in which decisions are delegated to committees.

Definition 2.10. Let N be a set of players and for $m \geq 1$ let $g_1, \dots, g_m$ be Wvgs with $g_k = [Q_k; w_k]$. Additionally, let φ be a propositional formula with variables $1, \dots, m$ and connectives $\wedge$ and $\vee$ where each variable appears exactly once. The pair $(\varphi, \{g_1, \dots, g_m\})$ is called *multiple weighted representation with formula* φ and for a simple game $(N, \mathcal{W})$ we say $(\varphi, \{g_1, \dots, g_m\})$ *represents* $(N, \mathcal{W})$ if

$(\varphi, \{g_1, \dots, g_m\})$

$$\forall S \in 2^N : (S \in \mathcal{W} \iff P(\varphi, S))$$

where for a propositional formula ψ with variables $1, \dots, m$ and connectives $\wedge$ and $\vee$, and a coalition $S \subseteq N$ the predicate P is inductively defined by

$$P(\psi, S) := \begin{cases} w_\psi(S) \geq Q_\psi & \text{if } \psi \in \{1, \dots, m\} \\ P(\alpha, S) \diamond P(\beta, S) & \text{if } \psi = \alpha \diamond \beta \text{ where } \diamond \in \{\wedge, \vee\} \end{cases} . \tag{2.4}$$

The game g_k is called the *kth rule.* $\square$

[8]The members of the council are representatives for the 27 countries in the EU.

An example for φ is $(1 \wedge 2) \vee 3$ but not $1 \wedge 3$ or $(1 \wedge 2) \vee 1$, because 2 is missing in the former and 1 appears twice in the latter formula. The restriction to connectives $\wedge$ and $\vee$ is necessary, because otherwise the set defined by $\{S \in 2^N \mid P(\varphi, S)\}$ is not guaranteed to be an up-set. In the current setting, each multiple weighted representation with a formula represents a simple game. We therefore can safely use a simple game and a multiple weighted representation with a formula of it interchangeably. The simple game $(N, \mathcal{W})$ in the following statement is called *associated with* $(\varphi, \{g_1, \dots, g_m\})$.

Proposition 2.3. *Let* $(\varphi, \{g_1, \dots, g_m\})$ *be a multiple weighted representation with formula* φ *and players* N. *Then* $(N, \mathcal{W})$ *is a simple game where the winning coalitions are* $\mathcal{W} = \{S \in 2^N \mid P(\varphi, S)\}$. □

If for $(\varphi, \{g_1, \dots, g_m\})$ the formula φ does only contain $\wedge$ as connective, then the simple game represented by $(\varphi, \{g_1, \dots, g_m\})$ and by the m-vector weighted voting game $\{g_1, \dots, g_m\}$ are the same.

To support the presentation, if there is no risk of confusion, we will use the rules $g_1, \dots, g_m$ of a multiple weighted representation with formula φ directly inside the formula φ instead of the numbers $1, \dots, m$. This is illustrated in the following example.

Example 2.5. Consider the WVGs $g_1 = [2; 1, 1, 1, 1]$, $g_2 = [66; 30, 30, 15, 15]$ and $g_3 = [3; 1, 1, 1, 1]$. The rule g_1 models a majority of two players, g_2 models a 66% majority and g_3 avoids a blocking majority with only one player. The formula is $\varphi = (1 \wedge 2) \vee 3$. By using the rules in φ instead of $1, 2, 3$, the game can be stated more compact as

$$([2; 1, 1, 1, 1] \wedge [66; 30, 30, 15, 15]) \vee [3; 1, 1, 1, 1]\,.$$

We also denote this formula by φ for convenience. □

A particular class of representations is the following:

Definition 2.11. Let $(N, \mathcal{W})$ be a simple game with types $N_1, \dots, N_t$. A multiple weighted representation with formula φ and rules $[Q_1; w_1], \dots, [Q_m; w_m]$ of $(N, \mathcal{W})$ is called *type preserving*, if all players of the same type have the same weight in each rule:

$$\forall j \in \{1, \dots, t\} : \forall p, q \in N_j : \forall k \in \{1, \dots, m\} : w_k(p) = w_k(q)\,.$$ □

We will come back to type preserving vector-weighted representations in the conclusions in Chapter 8. In Chapter 7 we consider type preserving weighted representation. For this case Def. 2.11 can easily be adopted.

3. Binary Decision Diagrams

In this chapter we lay the foundations of binary decision diagrams (BDD), which are used throughout the thesis. In Section 3.1 we present basic notions and notation of BDDs in general and QOBDDs in particular. After the introduction, Section 3.2 describes the representation of simple games and, more generally, the representation of sets of subsets by QOBDDs. Restrictions and concepts in implementations of BDDs are discussed in Section 3.3. This is necessary, because sometimes we do things that cannot (easily) be done with existing implementations and the results therefore might seem questionable. The synthesis of QOBDDs is an important aspect in the process of building the QOBDD for a simple game in practice. It is therefore discussed separately in Section 3.4.

3.1. Introduction to BDDs

Binary decision diagrams (abbreviated as BDDs) are a data structure and a model of computation, that is widely used in computer science and electrical engineering. Even though BDDs are around for some time (Lee 1959; Akers 1978), the contribution by Bryant (1986) to use a fixed variable ordering is usually considered as one of the main contributions that made BDDs applicable in practice. We will present a rather brief introduction in this section. The interesting reader is referred to Bryant (1992) for an introduction and Wegener (2000) for details.

Let $n \in \mathbb{N}$ be a number and let $N := \{1, \dots, n\}$ be a set of Boolean variables (*labels*). A BDD is a directed, acyclic and labeled multigraph G with node set $\mathsf{V} \cup \{\mathbb{O}, \mathbb{I}\}$ where $\mathbb{O}, \mathbb{I} \notin \mathsf{V}$ and edges E. The node set contains exactly one source, called the *root of* G, and the two sinks $\mathbb{O}$ and $\mathbb{I}$, called the *0-sink* and the *1-sink*, respectively. The sinks are also called *terminal nodes*. The non-terminal nodes in V are called *inner nodes*. If G is a BDD, then we use $\mathsf{V}(G)$ to denote the set of inner nodes of G. Each inner node has two outgoing edges, called the *0-edge* and the *1-edge*, respectively. Given an inner node $v \in \mathsf{V}(G)$ and the 0-edge (v, u) and the 1-edge (v, u') of v, there are two functions then and else, defined by $\mathsf{then}(v) = u$ and $\mathsf{else}(v) = u'$, respectively. The nodes $\mathsf{then}(v)$ and $\mathsf{else}(v)$ are called *successors of* v. Each inner node $v \in \mathsf{V}$ is labeled with an element from N, denoted by $\mathsf{var}(v)$, which is called the *label of node* v. The sinks have the special label $n + 1$, i.e., $\mathsf{var}(\mathbb{O}) = \mathsf{var}(\mathbb{I}) = n + 1$.

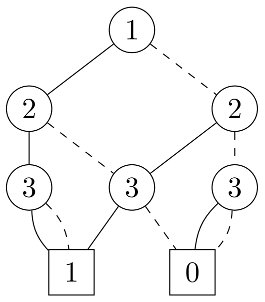
$\mathbb{O}, \mathbb{I}$

$\mathsf{V}(G)$

$\mathsf{then}(v), \mathsf{else}(v)$

$\mathsf{var}(v)$

Example 3.1. BDDs are usually drawn in a layered top-down fashion as depicted in Figure 3.1. The root is placed at the top and the two sinks are placed at the bottom layer. Inner nodes are drawn as circles, while the two sinks are drawn as squares to emphasize their special meaning. Labels of inner nodes are written inside the inner



variable	1	2	3	f-value
	0	0	0	0
	0	0	1	0
	0	1	0	0
	0	1	1	1
	1	0	0	0
	1	0	1	1
	1	1	0	1
	1	1	1	1

Figure 3.1.: A BDD for a Boolean function.

node's circles. For the sinks the label $n+1$ is omitted. Instead, the truth-values identify the sinks, i.e. 1 is written inside the square of the 1-sink and 0 inside the square of the 0-sink. Edges are always directed downwards, so that arrows can be omitted. An 1-edge is drawn solid, while a 0-edge is drawn dashed. As example, for the root node r in Figure 3.1, $\mathsf{then}(r)$ is the left node with label 2 and $\mathsf{else}(r)$ is the right node with label 2. $\square$

$\mathsf{V}(r)$

Each node v of a BDD G can be considered as the root of a BDD. The nodes of the corresponding BDD are those in $\mathsf{V}(G)$, that are reachable from v, plus the sinks $\mathbb{O}, \mathbb{I}$. We denote the subset of the reachable inner nodes by $\mathsf{V}(r)$. The labels and the edges of the BDD with root v can be defined similarly. For this reason, in the remaining text we will not distinguish between a BDD and its root. We usually prefer to use roots as representatives for BDDs.

f_v

Each BDD node v with label i represents a Boolean function $f_v : \mathbb{B}^{n-i+1} \to \mathbb{B}$ as follows. Let $(x_i, \ldots, x_n) \in \mathbb{B}^{n-i+1}$ be an input vector for f_v. We obtain the value for $f_v(x_i, \ldots, x_n)$ by a traversal of the BDD starting at v. At each node u, if u is not a sink and the label of u is k, then take the 1-edge if $x_k = 1$ and take the 0-edge otherwise. If u is a sink, return 0 if $v = \mathbb{O}$ and return 1 otherwise. Figure 3.1 shows the BDD and the truth table of the represented Boolean function with 3 variables.

Let the node v be (the root of) a BDD with label i. By $\overline{v}$ we denote the complemented BDD, that is, the BDD where both terminal nodes are interchanged. It holds $f_v(x_i, \ldots, x_n) = 1$ if and only if $f_{\overline{v}}(x_i, \ldots, x_n) = 0$ for $x_i, \ldots, x_n \in \mathbb{B}$.

π-OBDD

Let $\pi : N \to N$ be a permutation. The position of label $i \in N$ in the ordering π is $\pi^{-1}(i)$. A BDD G is called *π-ordered* (abbreviated as π-OBDD), if for each node $v \in \mathsf{V}(G)$ with $\mathsf{then}(v), \mathsf{else}(v) \notin \{\mathbb{O}, \mathbb{I}\}$ it holds

$$\pi^{-1}(\mathsf{var}(\mathsf{then}(v))) = \pi^{-1}(\mathsf{var}(v)) + 1 \quad \text{and} \quad \pi^{-1}(\mathsf{var}(\mathsf{else}(v))) = \pi^{-1}(\mathsf{var}(v)) + 1\,.$$

OBDD

The permutation π is called *ordering (of the labels N)* or *variable ordering.* In case π is the canonical ordering $\pi(i) = i$, then the explicit reference to π is omitted and we call such a BDD *ordered* (OBDD). In most parts of this thesis we will make use of the canonical ordering, that is, the label at the top is 1 and the label at the bottom is n. The sinks have the special label $n+1$. Statements hold without loss of generality for arbitrary orderings, if not stated otherwise.

An inner node $v \in \mathsf{V}(G)$ is called *redundant*, if $\mathsf{then}(u) = \mathsf{else}(u)$. Two different inner nodes u, v with $\mathsf{var}(u) = \mathsf{var}(v)$ are called *equivalent*, if $\mathsf{then}(u) = \mathsf{then}(v)$ and $\mathsf{else}(u) = \mathsf{else}(v)$. A π-OBDD is called *complete*, if for each inner node v and each non-terminal successor u of v it holds $\pi^{-1}(\mathsf{var}(u)) = \pi^{-1}(\mathsf{var}(v)) + 1$, that is, the label of u is the next label w.r.t. the ordering π. A π-OBDD is called *quasi-reduced* (π-QOBDD), if it is complete and does not have any equivalent inner nodes. The BDD in Figure 3.1 is complete, does not have any equivalent nodes and therefore is quasi-reduced. QOBDDs are canonical, i.e., a π-QOBDD representing a Boolean function is unique up to isomorphism (Bryant 1986). π-QOBDD QOBDD

For an OBDD with root r and label $i \in N$, the nodes in $\mathsf{V}(r)$ with label i are $\{v \in \mathsf{V}(r) \mid \mathsf{var}(v) = i\}$. We denote this set by $\mathsf{V}_i(r)$ and refer to it as *level i of the* OBDD *with root r*. The term "level" has been chosen because of the vertical arrangement of the nodes in illustrations of OBDDs. See Figure 3.1 again. For convenience, we define $\mathsf{V}_{n+1}(r) := \{\mathbb{O}, \mathbb{I}\}$. If the root of the OBDD is clear from the context, the reference to the OBDD is omitted and as for V, we write V_i. $\mathsf{V}_i(r)$ $\mathsf{V}_{n+1}(r)$

The *size of a* BDD *with root r* is defined as the number of its inner nodes, that is, $\mathsf{size}(r) := |\mathsf{V}(r)|$. The *width of the* OBDD *with root r* is $\mathsf{width}(r) := \max_{i \in N} |\mathsf{V}_i(r)|$. For instance, the QOBDD in Figure 3.1 has size 6 and width 3. The *width of level i of* r is defined by $\mathsf{width}_i(r) := |\mathsf{V}_i(r)|$. $\mathsf{size}(r)$ $\mathsf{width}(r)$ $\mathsf{width}_i(r)$

The ordering π of the labels (resp. variables) can have huge impact on the size of an π-OBDD. The problem to find an ordering that minimizes the size of an OBDD for a given Boolean function has been investigated extensively. The problem is NP-complete for OBDDs representing general Boolean functions (Bollig and Wegener 1996) and remains NP-complete for OBDDs representing monotone Boolean functions[1] (Iwama, Nozoe, and Yajima 1998). For QOBDDs representing threshold functions[2] it is unknown if finding an optimal variable ordering is still NP-hard. Behle (2008) has presented an integer linear program to solve this problem. There are various heuristics and also exact methods to find "good" variable orderings in the general case. See Somenzi (1999) for an overview.

Implementations like CUDD[3] (Somenzi 1998) use so-called reduced OBDD (ROBDDs) instead of QOBDDs. ROBDDs neither have redundant nor equivalent nodes and therefore usually require less nodes in comparison to QOBDDs. This reduction, however, comes at the cost of more complex operations on ROBDDs. Liaw and Lin (1992) have evaluated the number of redundant inner nodes experimentally. They have figured out that for random QOBDDs the proportion of redundant nodes decreases if the number of variables increases and at 14 variables the proportion of redundant inner nodes is less than 1%. They also remark that in some situations it therefore could be beneficial to use QOBDDs instead of ROBDDs, because a data structure for QOBDD nodes does not have to store the label with each node. Furthermore, Wegener (1994) has shown that for almost all ROBDD

[1]A Boolean function is called *monotone*, if it can be represented as a formula with only positive literals. The notions of an up-set and a monotone Boolean function coincide.

[2]A Boolean function $f : \mathbb{B}^n \to \mathbb{B}$ is called a *threshold function* if there are (maybe negative) integers $w_1, \ldots, w_n$ and T such that $f(x_1, \ldots, x_n) = 1$ if and only if $\sum_{i=1}^{n} x_i \cdot w_i \geq T$, where $x_1, \ldots, x_n \in \mathbb{B}$. The notions of a weighted voting game and a threshold function nearly coincide up to negative weights.

[3]Short for *CU Decision Diagram.*

Boolean functions the number of redundant nodes is exponentially small.

The use of QOBDDs in this thesis is essential for our approach. Some of the ideas, especially those in Sections 6.1 and 6.2, depend on the fact that all paths from the root to the sinks are complete.

ite

The operation to create a QOBDD node is called ite, short for *if-then-else*. Given a label i and QOBDD nodes u, v such that $\mathsf{var}(u) = \mathsf{var}(v) = i + 1$ and there is no node with label i in $\mathsf{V}(u)$ and $\mathsf{V}(v)$ then $\mathsf{ite}(i, u, v)$ returns a node w satisfying

$$\mathsf{var}(w) = i,\ \mathsf{then}(w) = u,\ \mathsf{else}(w) = v\,.$$

For implementation details and the running time of ite see Section 3.3 below.

Example 3.2. Let u and v denote the left and right node with label 2 in Figure 3.1 and let r denote the root of the BDD. Because there is neither a node with label 1 in $\mathsf{V}(u)$ nor in $\mathsf{V}(v)$, we could apply $\mathsf{ite}(1, u, v)$, which would return the root r, that is, the BDD depicted in Figure 3.1. □

3.2. Representation of Simple Games

As in the previous section, BDDs are usually used to represent Boolean functions $f : \mathbb{B}^n \to \mathbb{B}$. Our intention, however, is to represent simple games. Let $(N, \mathcal{W})$ be a n-person simple game with players $N = \{1, \ldots, n\}$. The *characteristic function of* $\mathcal{W}$ is the function $\chi_{\mathcal{W}} : 2^N \to \mathbb{B}$ such that:

$$\forall S \in 2^N : (S \in \mathcal{W} \iff \chi_{\mathcal{W}}(S) = 1)\,. \tag{3.1}$$

As described by Somenzi (1999, Section 8.1) to obtain a Boolean function, each subset S of N can be encoded as a Boolean vector $(x_1, \ldots, x_n) \in \mathbb{B}^n$ such that for each $i \in N$ it holds $x_i = 1$ if and only if $i \in S$. This is reasonable in most cases, but we prefer a direct interpretation of a QOBDD as a subset of 2^N.

set(v)

Definition 3.1. Let r be the root of a QOBDD. For a node $v \in \mathsf{V}(r) \cup \{\mathbb{O}, \mathbb{I}\}$, we define the *set represented by* v, denoted by $\mathsf{set}(v)$, as subset of $2^{\{\mathsf{var}(v),\ldots,n\}}$ inductively by:

$$\mathsf{set}(v) := \begin{cases} \emptyset & \text{if } v = \mathbb{O} \\ \{\emptyset\} & \text{if } v = \mathbb{I} \\ \{S + \mathsf{var}(v) \mid S \in \mathsf{set}(\mathsf{then}(v))\} \cup \mathsf{set}(\mathsf{else}(v)) & \text{otherwise} \end{cases}$$

We say that v *represents* $\mathcal{A} \subseteq 2^{\{\mathsf{var}(v),\ldots,n\}}$ if $\mathsf{set}(v) = \mathcal{A}$ and we say that v *represents the simple game* $(N, \mathcal{W})$ if v represents $\mathcal{W}$. Additionally we say that v *is (the root of) the* QOBDD *for* $\mathcal{A}$ *and* $(N, \mathcal{W})$, respectively. □

For example, the QOBDD r in Figure 3.1 represents $\{AB, AC, BC, ABC\}$, which are the winning coalitions of the weighted voting game $[5; 4, 3, 2]$ and A denotes the first player, B denotes the second player and so on.

Due to the canonicity of QOBDDs, for two QOBDD nodes v, w with label $i \in N$ it holds $v = w$ if and only if $\mathsf{set}(v) = \mathsf{set}(w)$.

The set $\mathsf{set}(\overline{v})$ represented by the complemented QOBDD of v, is generally not the same as the complement $\overline{\mathsf{set}(v)}$ of the set represented by v, since the latter set operation usually refers to the powerset 2^N as universe. $\overline{v}$

If for an inner node v with label i it holds $\mathsf{set}(v) = \emptyset$ or $\mathsf{set}(v) = 2^{\{i,\dots,n\}}$, then this node is redundant and it is very much related to the sinks $\mathbb{O}$ and $\mathbb{I}$, respectively. As a convenience, in the following we denote v by the symbol $\mathbb{O}_i$ if $\mathsf{set}(v) = \emptyset$ and we denote v by the symbol $\mathbb{I}_i$ if $\mathsf{set}(v) = 2^{\{i,\dots,n\}}$. To be consistent, we additionally define $\mathbb{O}_{n+1}$ as $\mathbb{O}$ and $\mathbb{I}_{n+1}$ as $\mathbb{I}$. For $1 \le i \le n$, both $\mathbb{I}_i$ and $\mathbb{O}_i$ can be absent, but if any of them exists, then it is unique, because no two different nodes represent the same set. It is easy to obtain these nodes by the recursive formula $\mathbb{O}_i = \mathsf{ite}(i, \mathbb{O}_{i+1}, \mathbb{O}_{i+1})$ and analogously for $\mathbb{I}_i$. In the following we therefore can safely refer to these nodes as if they always exist. For example, in the QOBDD depicted in Figure 3.1 the leftmost and rightmost nodes with label 3 are $\mathbb{I}_3$ and $\mathbb{O}_3$, respectively, but neither of the nodes $\mathbb{I}_1, \mathbb{O}_1$ and $\mathbb{I}_2, \mathbb{O}_2$ exists. $\mathbb{O}_i, \mathbb{I}_i$ $\mathbb{O}_{n+1}, \mathbb{I}_{n+1}$

For a label $i \in N$ and a subset of $\mathcal{A} \subseteq 2^{\{i,\dots,n\}}$ the set $\{S - i \mid i \in S,\ S \in \mathcal{A}\}$ is denoted by $\mathcal{A}_i$ and the set $\{S \in \mathcal{A} \mid i \notin S\}$ is denoted by $\mathcal{A}_{\neg i}$. The definition is motivated by the observation that if $\mathcal{A}$ is represented by a QOBDD v with label i, then it holds: $\mathcal{A}_i, \mathcal{A}_{\neg i}$

$$\mathcal{A}_i = \mathsf{set}(\mathsf{then}(v)) \quad \text{and} \quad \mathcal{A}_{\neg i} = \mathsf{set}(\mathsf{else}(v))\,.$$

In the context of sets, the if-then-else operation ite has the following meaning for a QOBDD inner node v with label i:

$$\mathsf{set}(v) = \{S + i \mid S \in \mathsf{set}(\mathsf{then}(v))\} \cup \mathsf{set}(\mathsf{else}(v)) = \mathsf{set}(\mathsf{ite}(i, \mathsf{then}(v), \mathsf{else}(v)))\,.$$

The notion of a path complements the concept of the set representation of a QOBDD node. It will play a prominent role for some counting algorithms in Section 6.2 and it will also facilitate a thorough understanding of QOBDDs representing sets and simple games in general.

Definition 3.2. For a QOBDD node v, a label $i \in N$ with $\mathsf{var}(v) \le i$ and a subset $S \subseteq \{\mathsf{var}(v), \dots, i-1\}$ the *node that is reached on level i starting at v by S*, denoted by $\mathsf{node}(v, i, S)$, is recursively defined by: $\mathsf{node}(v, i, S)$

$$\mathsf{node}(v,i,S) := \begin{cases} v & \text{if } \mathsf{var}(v) = i \\ \mathsf{node}(\mathsf{then}(v), i, S - \mathsf{var}(v)) & \text{if } \mathsf{var}(v) < i \text{ and } \mathsf{var}(v) \in S \\ \mathsf{node}(\mathsf{else}(v), i, S) & \text{if } \mathsf{var}(v) < i \text{ and } \mathsf{var}(v) \notin S \end{cases}.$$

□

The set S in the previous definition is sometimes called a *path*. As an example, for the QOBDD with root r in Figure 3.2 it holds $\mathsf{node}(r, 3, \{1\}) = v$ but also $\mathsf{node}(r, 3, \{2\}) = v$. For the former, at the root r we take the 1-edge, because the label 1 is in $\{1\}$. At the node $\mathsf{then}(r)$ we take the 0-edge, because $2 \notin \{1\}$. Now, the node $\mathsf{else}(\mathsf{then}(r))$ has label 3 and this v.

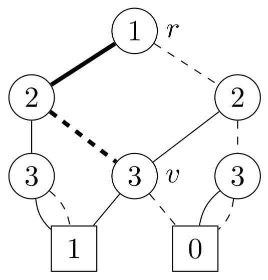


Figure 3.2.: Illustration how the node v is reached on level 3 starting at r by $\{1\}$.

The function node is total, but not necessarily injective. Having a starting node v and a path S but no label i is insufficient to uniquely identify a node in general. Consider for instance the empty set and the root r in Figure 3.2.

While the previous definition is convenient in proofs, the following notation is more intuitive in many situations.

$u \overset{S}{\to} v$

Definition 3.3. Let u, v be QOBDD nodes such that $\mathsf{var}(u) \leq \mathsf{var}(v)$ and let S be a subset of $\{\mathsf{var}(u), \dots, \mathsf{var}(v) - 1\}$. The fact that we *reach v from u by S*, denoted by $u \overset{S}{\to} v$, is defined by:

$$u \overset{S}{\to} v :\Longleftrightarrow \mathsf{node}(u, \mathsf{var}(v), S) = v\,.$$

$\mathsf{paths}(u, v)$

If $u \overset{S}{\to} v$ then we call S a *path from u to v*. By $\mathsf{paths}(u, v)$ we denote the set of all paths from u to v:

$$\mathsf{paths}(u, v) := \{S \subseteq \{\mathsf{var}(u), \dots, \mathsf{var}(v) - 1\} \mid u \overset{S}{\to} v\}\,. \qquad \Box$$

Figure 3.2 shows an example for the path $\{1\}$ from the root r to the node v. It does also hold $r \overset{\{2\}}{\to} v$. Other examples are $r \overset{\{1,2,3\}}{\to} \mathbb{I}$, $r \overset{\emptyset}{\to} \mathbb{O}$ and $w \overset{\emptyset}{\to} w$ for any node w. For the set of paths of the QOBDD in Figure 3.2 we have:

$$\begin{aligned}
\mathsf{paths}(r, v) &= \{\{1\}, \{2\}\}\,,\\
\mathsf{paths}(r, \mathbb{O}) &= \{\emptyset, \{1\}, \{2\}, \{3\}\}\,,\\
\mathsf{paths}(r, \mathbb{I}) &= \{\{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}\,.
\end{aligned}$$

As we will see now, it holds $\mathsf{paths}(r, \mathbb{I}) = \mathsf{set}(r)$ and consequently $\mathsf{paths}(r, \mathbb{O}) = \mathsf{set}(\overline{r})$.

Lemma 3.1. *For a* QOBDD *node v with label i and a subset S of $\{i, \dots, n\}$ it holds $S \in \mathsf{set}(v)$ if and only if $v \overset{S}{\to} \mathbb{I}$.*

Proof. The proof is by induction on the distance between the label of a node v and the label of the sinks $n+1$. In the base case, v is a sink and the distance is 0. It then holds $S = \emptyset$ and either $v = \mathbb{O}$ or $v = \mathbb{I}$. If $v = \mathbb{I}$ then by definition both statements $\emptyset \in \mathsf{set}(v)$ and $\mathbb{I} \overset{\emptyset}{\to} \mathbb{I}$ are true. Otherwise, if $v = \mathbb{O}$ then both statements $\emptyset \in \mathsf{set}(\mathbb{O})$ and $\mathbb{O} \overset{\emptyset}{\to} \mathbb{I}$ are false.

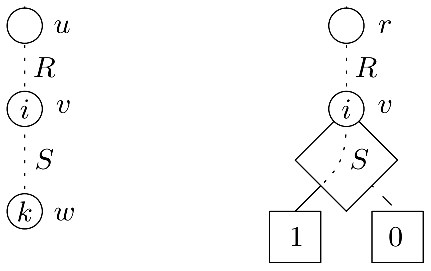


Figure 3.3.: Illustrations of the statements in Lemma 3.2 (left) and Theorem 3.3 (right). A dotted line represents a sequence of edges.

For the induction step we assume that the statement holds for $\mathsf{then}(v)$ and $\mathsf{else}(v)$. In the following we will make use of the fact that $\wedge$ has precedence over $\vee$. Then we have:

$$\begin{aligned}
& S \in \mathsf{set}(v) \\
\Longleftrightarrow\ & i \in S \wedge S - i \in \mathsf{set}(\mathsf{then}(v)) \vee S \in \mathsf{set}(\mathsf{else}(v)) && \text{(Def. } \mathsf{set}(v)\text{)} \\
\Longleftrightarrow\ & i \in S \wedge \mathsf{then}(v) \overset{S-i}{\to} \mathbb{I} \vee \mathsf{else}(v) \overset{S}{\to} \mathbb{I} && \text{(Ind. Hyp.)} \\
\Longleftrightarrow\ & i \in S \wedge \mathsf{node}(\mathsf{then}(v), n+1, S-i) = \mathbb{I} \vee \mathsf{node}(\mathsf{else}(v), n+1, S) = \mathbb{I} && \text{(Def. 3.3)} \\
\Longleftrightarrow\ & \mathsf{node}(v, n+1, S) = \mathbb{I} && \text{(Def. } \mathsf{node}\text{)} \\
\Longleftrightarrow\ & v \overset{S}{\to} \mathbb{I}\,. && \square
\end{aligned}$$

By the following lemma it will be possible to decompose a path with respect to a certain level i. The statement is rather intuitive and illustrated in Figure 3.3.

Lemma 3.2. *For* QOBDD *nodes u, w with $\mathsf{var}(u) \le \mathsf{var}(w)$, label $i \in \{\mathsf{var}(u), \dots, \mathsf{var}(w)\}$ and sets $R \subseteq \{\mathsf{var}(u), \dots, i-1\}$ and $S \subseteq \{i, \dots, \mathsf{var}(w)\}$ it holds*

$$u \overset{R\cup S}{\to} w \iff u \overset{R}{\to} v \wedge v \overset{S}{\to} w \tag{3.2}$$

where $v = \mathsf{node}(u, i, R)$.

Proof. By applying Def. 3.3 to the right-hand side of (3.2), the equivalence in (3.2) can be rewritten to

$$u \overset{R\cup S}{\to} w \iff \mathsf{node}(\mathsf{node}(u, i, R), k, S) = w$$

where $k = \mathsf{var}(w)$. We prove this equivalence by induction on i. For $i = \mathsf{var}(u)$ it holds $R = \emptyset$ and therefore by Def. 3.3 we have

$$u \overset{R\cup S}{\to} w \iff \mathsf{node}(u, k, R \cup S) = w \iff \mathsf{node}(\mathsf{node}(u, i, R), k, S) = w$$

because $\mathsf{node}(u, \mathsf{var}(u), \emptyset) = u$. For the induction step we assume that the statement is correct for $i \in \{\mathsf{var}(u), \dots, k-1\}$. Then for $i+1$ we have $R \subseteq \{\mathsf{var}(u), \dots, i\}$ and

$S \subseteq \{i+1,\dots,k\}$ and there are two cases. First, if $i \in R$ then set $R' := R - i$ and $S' := S + i$. Then for the induction step we have:

$$\begin{aligned}
u \overset{R\cup S}{\to} w &\iff u \overset{R'\cup S'}{\to} w && (R \cup S = R' \cup S')\\
&\iff \mathsf{node}(\mathsf{node}(u,i,R'),k,S') = w && \text{(Ind. Hyp.)}\\
&\iff \mathsf{node}(\mathsf{then}(\mathsf{node}(u,i,R')),k,S'-i) = w && \text{(Def. } \mathsf{node}\text{)}\\
&\iff \mathsf{node}(\mathsf{node}(u,i+1,R'+i),k,S) = w && \text{(Def. } \mathsf{node}\text{)}\\
&\iff \mathsf{node}(\mathsf{node}(u,i+1,R),k,S) = w\,.
\end{aligned}$$

The case $i \notin R$ can be shown similarly. □

The next important statement weaves together the previous two statements. It is the key for some results on counting later in Section 6.2. In the context of OBDDs representing Boolean functions, a very similar statement exists, which is sometimes referred to as *structure theorem* (Sieling and Wegener 1993). The statement is illustrated in Figure 3.3.

Theorem 3.3. *Let r be a* QOBDD *node and let $i \in \{\mathit{var}(r),\dots,n\}$ be a label. For sets $R \subseteq \{\mathit{var}(r),\dots,i-1\}$ and $S \subseteq \{i,\dots,n\}$ and $v := \mathsf{node}(r,i,R)$ it holds*

$$R \cup S \in \mathsf{set}(r) \iff r \overset{R}{\to} v \wedge S \in \mathsf{set}(v)\,.$$

Proof. By using Lemma 3.1 and Lemma 3.2 we have:

$$R \cup S \in \mathsf{set}(r) \iff r \overset{R\cup S}{\to} \mathbb{I} \iff r \overset{R}{\to} v \wedge v \overset{S}{\to} \mathbb{I} \iff r \overset{R}{\to} v \wedge S \in \mathsf{set}(v)\,.$$ □

3.3. Implementation Notes

When Dijkstra's algorithm for the single-source shortest path problem on an undirected graph is implemented, the actual running time depends on the choice of the data structure, that is used for the priority queue; see for instance Cormen, Leiserson, and Rivest (2001). Similarly, the running time for our algorithms later depends on some details in the implementation of QOBDDs. We will first discuss so-called shared OBDDs and the cost of creating an inner node using the operation ite whose running time is either expected or deterministic constant. The second detail is the cost of storing information for a BDD node. We begin by giving a brief overview on how OBDDs are implemented in practice.

Even though, most details in this section originally refer to reduced OBDDs, they do also apply to QOBDDs. An implementation of BDDs (regardless of the actual type such as ROBDDs or QOBDDs) is usually called a BDD *package*. The implementation of a BDD package is a topic on its own, because many design decisions have to be made (Brace, Rudell, and Bryant 1990; Ossowski and Baier 2008). Among the numerous BDD packages, CUDD (Somenzi 1998) supports many advanced features such as asynchronous reordering of variables and some variants of BDDs. QOBDDs are not supported. The

Operation	shared	non-shared
create inner node (ite)	exp. $\mathcal{O}(1)$	$\mathcal{O}(1)$
$\mathsf{set}(u) = \mathsf{set}(v)$ for nodes u, v	$\mathcal{O}(1)$	$\mathcal{O}(\min\{\mathsf{size}(u), \mathsf{size}(v)\})$

Table 3.1.: Running times for shared and non-shared QOBDD nodes for some operations.

following brief overview of the high-level design of a BDD package will facilitate the understanding of some subtle details.

A BDD package is usually divided into two kinds of objects: (BDD) *nodes* and *managers*. The data structure for a node has fields for the two successors, usually realized as pointers, the label and usually fields for reference counting and internal use. A Boolean function in a BDD package is usually implemented as a pointer to a node. Beside others, two main responsibilities of a manager are to maintain a set of non-equivalent inner nodes and to create new inner nodes on demand. By this division and because OBDDs are canonical for a fixed variable ordering, it is possible to let a set of nodes represent different Boolean functions. Therefore, OBDDs that can be used to represent more than a single Boolean function are sometimes called *shared* OBDD*s* (Minato, Ishiura, and Yajima 1990). The opposite is to use a single OBDD for each Boolean function and then one speaks of *non-shared* OBDD*s*.

When using shared nodes, the manager uses a so-called *unique table* to maintain a set of non-equivalent nodes that is implemented by a hash table. If the manager M is requested to create a node by the call

$$\mathsf{ite}_M(i, u, v) \tag{3.3}$$

where ite now has to refer to the manager M for the unique table, it queries M's unique table and does only create a new node if no such node with successors u and v and label i exists. Querying the unique table costs expected time $\mathcal{O}(1)$. In some situations, however, we will never ask to create a QOBDD node twice and having non-shared nodes is perfectly fine. In these situations that we will encounter at least once in the remaining text, we do not use shared OBDDs and we will assume that the call in (3.3) does not query the unique table. For these algorithms we will then have deterministic constant time for creating a node by using the operation ite. See Table 3.1 for the differences of shared and non-shared OBDDs for some operations.

The second detail is to store information for a BDD node. The data structure that is used to store a BDD node in a BDD package is usually immutable by the user of such a package. For example, if we would like to store an additional integer with each node v for the value $|\mathsf{set}(v)|$, this cannot easily be done. This restriction is mainly due to technical reasons and other implementations are possible. The approach in BDD packages such as CUDD is to use a hash table with the BDD nodes as the keys and the additional information as the values. The drawback of such a solution is the expected time $\mathcal{O}(1)$ to access these values. In this thesis (especially in Section 6.2) we assume that no such restriction exists and therefore information for a node can be accessed in (deterministic) constant time.

3.4. Synthesis

The process of applying a binary Boolean function $\otimes : \mathbb{B}^2 \to \mathbb{B}$ to two QOBDD nodes is called *binary synthesis* and it is realized by a recursive algorithm and a computed table to avoid redundant computations. The algorithm is usually called apply and it takes $\otimes$ as an argument. We use apply as listed in Algorithm 1. The algorithm traverses the QOBDDs with roots u, v and applies $\otimes$ in the case that both nodes are a sink. For nodes u, v with label i and a set $S \subseteq \{i, \dots, n\}$ the term $S \in \mathsf{set}(u) \otimes S \in \mathsf{set}(v)$ in the postcondition of algorithm apply refers to the meta level where $S \in \mathsf{set}(u)$ and $S \in \mathsf{set}(v)$ are considered as predicates.

We use shared nodes here and therefore, ite is assumed to have expected constant time throughout this section. The computed table is assumed to be in a global scope and can be reused in subsequent calls to apply. It is usually implemented by a hash table and we therefore assume expected constant time for the operations *insertion* and *lookup*. If *lookup* is called with a key that has not already been stored, then $\perp$ is returned. Sometimes, as in Wegener (2000), the operator $\otimes$ is not used in the key of the computed table and each operator is assumed to possess a dedicated computed table. Implementations like CUDD (Somenzi 1998), however, use it for the key. Which method is used is of no relevance in our further considerations, though.

Algorithm 1 apply$(i, u, v, \otimes)$

```
Require: $u, v$ are QOBDD nodes with label $i$ and $\otimes : \mathbb{B}^2 \to \mathbb{B}$.
Ensure: $\mathsf{set}(\mathsf{apply}(i, u, v, \otimes)) = \{S \in 2^{\{i,\dots,n\}} \mid S \in \mathsf{set}(u) \otimes S \in \mathsf{set}(v)\}$
    if $(u = \mathbb{I}) \otimes (v = \mathbb{I})$ then return $\mathbb{I}$
    else if $u \in \{\mathbb{I}, \mathbb{O}\}$ then return $\mathbb{O}$
    else if $e \neq \perp$ then return $e$ where $e = lookup(T, (\otimes, u, v))$
    else
        $r \leftarrow \mathsf{ite}(i, \mathsf{apply}(i+1, \mathsf{then}(u), \mathsf{then}(v), \otimes), \mathsf{apply}(i+1, \mathsf{else}(u), \mathsf{else}(v), \otimes))$
        $insert(T, (\otimes, u, v), r)$
        return $r$
```

$u \otimes v$

For QOBDD nodes u, v with $\mathsf{var}(u) = \mathsf{var}(v)$ we will usually write $u \otimes v$ in the following instead of $\mathsf{apply}(\mathsf{var}(u), u, v, \otimes)$. The correctness and running time of algorithm apply in Algorithm 1 can rather easily be shown. For details see Wegener (2000, Section 3.3) or (Bryant 1986, Section 4.3).

Proposition 3.4. *Let u, v be QOBDD nodes with label i. The binary synthesis using $\mathsf{apply}(i, u, v, \otimes)$ is possible in expected running time $\mathcal{O}(\sum_{j=i,\dots,n} \mathsf{width}_j(u)\mathsf{width}_j(v))$ and on each level $j \in \{i, \dots, n\}$ there are at most $\mathsf{width}_j(u) \cdot \mathsf{width}_j(v)$ nodes in the result.* $\square$

Bryant (1986) has shown, that for ROBDDs with roots u, v the binary synthesis takes expected time $\mathcal{O}(\mathsf{size}(u) \cdot \mathsf{size}(v))$. Wegener (2000) has presented an example, for which this bound is tight regardless of the ordering of the labels. In the case of quasi-reduced OBDDs we can expect a similar result w.r.t. to the complexity stated in Proposition 3.4.

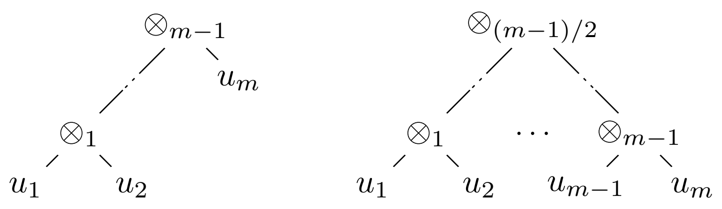


Figure 3.4.: Expression tree for the sequential m-ary synthesis (left) and an expression tree with height $\log_2(m)$ (right).

To compute the QOBDD for an arbitrary expression we consider an expression tree with QOBDD nodes $u_1, \ldots, u_m$ for the leaves (operands) and Boolean binary functions $\otimes_1, \ldots, \otimes_{m-1}$ for the inner nodes (operators). Figure 3.4 shows two examples for an expression tree. For sake of simplicity, in the following we assume $\mathsf{var}(u_k) = 1$ for $k \in \{1, \ldots, m\}$. A node v in such an expression tree is either a leave u_k for some k or a triple $(t_l, \otimes, t_r)$ with left (resp. right) child t_l (resp. t_r) and $\otimes$ is one of the operators. The expression is usually assumed to have constant size, so that the complexity to compute the result is not considered. For our application to simple games, the formula is part of the input, though. The leaves and the number of operators (also called *size*) for a node t are denoted by $\mathsf{leaves}(t)$ and $|t|$, respectively. Because each operator node t has two children, it holds $|t| = |\mathsf{leaves}(t)| - 1$. $\mathsf{leaves}(t)$ $|t|$

Theorem 3.5. *For an expression tree with leaves $u_1, \ldots, u_m$ and root node $s = (s_l, \otimes, s_r)$ it takes expected time*

$$\mathcal{O}(n(\prod_{k=1}^{m} \mathit{width}(u_k) + |s_l| \cdot \prod_{v \in \mathit{leaves}(s_l)} \mathit{width}(v) + |s_r| \cdot \prod_{v \in \mathit{leaves}(s_r)} \mathit{width}(v)))$$

to compute its results and the result QOBDD *has size*

$$\sum_{i=1}^{n} \prod_{k=1}^{m} \mathit{width}_i(u_k) \leq n \prod_{k=1}^{m} \mathit{width}(u_k)\,.$$

Proof. The QOBDD for an expression tree with root t, which we denote by $f(t)$, can be recursively computed by

$$f(t) = \begin{cases} t & \text{if } t \text{ is a leave} \\ f(t_l) \otimes f(t_r) & \text{if } t = (t_l, \otimes, t_r)\,. \end{cases}$$

By using Proposition 3.4 and induction on the structure of the expression tree for each level $i \in N$ it can easily be seen that

$$\mathsf{width}_i(f(t)) \leq \prod_{v \in \mathsf{leaves}(t)} \mathsf{width}_i(u)\,.$$

This shows the size of the result QOBDD for $t = s$. For the running time we have to accumulate the running times for the applications of the operators $\otimes_1, \ldots, \otimes_{m-1}$. For a node $t = (t_l, \otimes, t_r)$ in the expression tree with root s

$$\mathcal{O}(\sum_{i=1}^{n} \prod_{v \in \mathsf{leaves}(t_l)} \mathsf{width}_i(v))$$

and therefore

$$\mathcal{O}(n \cdot \prod_{v \in \mathsf{leaves}(t_l)} \mathsf{width}(v)) \tag{3.4}$$

is an upper bound for the expected running time for each single application of an operator in t_l. Because the number of operators is $|t_l|$, the overall expected running time to obtain $f(t_l)$ is therefore bounded by

$$\mathcal{O}(|t_l| n \cdot \prod_{v \in \mathsf{leaves}(t_l)} \mathsf{width}(v))$$

Analogously for t_r. The expected running time for the application of the operator at the top is similarly to (3.4) bounded by $\mathcal{O}(n \prod_{k=1,\ldots,m} \mathsf{width}(u_k))$. □

Because the expression tree is usually much smaller than the widths of the QOBDDs, in the expected running time of Theorem 3.5 the term $n \prod_{k=1,\ldots,m} \mathsf{width}(u_k)$ is dominant in most cases.

For the *sequential m-ary synthesis* $((u_1 \otimes_1 u_2) \otimes_2 \cdots) \otimes_{m-1} u_m$ which is also depicted in Figure 3.4 (left), we can show an improved upper bound for the expected running time.

Theorem 3.6. *Let* $\mathsf{width}(u_1) \leq \ldots \leq \mathsf{width}(u_m)$ *and let* W *denote* $\mathsf{width}(u_m)$. *Then the* QOBDD *for* $((u_1 \otimes_1 u_2) \otimes_2 \cdots) \otimes_{m-1} u_m$ *can be computed in expected time*

$$\mathcal{O}(n(1 + \frac{m}{W}) \prod_{j=1}^{m} \mathsf{width}(u_j)) .$$

Proof. Let $v_1 = u_1$ and for $k = 1, \ldots, m-1$ let v_{k+1} denote the result of $v_k \otimes_k u_{k+1}$. By induction on k it holds $\mathsf{width}(v_k) \leq \prod_{j=1}^{k} \mathsf{width}(u_j)$ and all the results $v_1, \ldots, v_k$ can be computed in expected time $\mathcal{O}(\sum_{p=2}^{k} n \prod_{j=1}^{p} \mathsf{width}(v_j))$. The overall result v_m can therefore be computed in expected time

$$\mathcal{O}(\sum_{p=2}^{m} n \prod_{j=1}^{p} \mathsf{width}(v_j)) . \tag{3.5}$$

For $k = 1, \ldots, m$ we use w_k short for $\mathsf{width}(v_k)$. We rewrite (3.5) to:

$$\sum_{p=2}^{m} n \prod_{j=1}^{p} w_j = n \sum_{p=2}^{m} \frac{w_1 \cdot \cdots \cdot w_m}{\prod_{j=p+1}^{m} w_j} = n(\prod_{j=1}^{m} w_j) \cdot \sum_{p=2}^{m} \prod_{j=p+1}^{m} w_j^{-1} . \tag{3.6}$$

Because $\prod_{j=m+1}^{m} w_j^{-1} = 1$ and $w_m^{-1} = W^{-1}$ appears in each of the products for $p = 2, \dots, m$, the sum on the right-hand side in (3.6) can be bounded by:

$$\sum_{p=2}^{m} \prod_{j=p+1}^{m} w_j^{-1} = 1 + \sum_{p=2}^{m-1} \prod_{j=p+1}^{m} w_j^{-1} \leq 1 + \sum_{p=2}^{m-1} W^{-1} \leq 1 + \frac{m}{W}$$

By applying this to (3.5) we can conclude with:

$$\mathcal{O}(\sum_{p=2}^{m} n \prod_{j=1}^{p} \mathsf{width}(v_j)) \leq \mathcal{O}(n \prod_{j=1}^{m} \mathsf{width}(v_j) \cdot (1 + \frac{m}{W}))\,. \qquad \square$$

Part II.

Representation of Simple Games

4. QOBDDs representing Simple Games

Weighted voting games are the most important and most frequently used subclass of simple games in practice. For that reason, in this chapter we will spend most time on the analysis of QOBDDs representing weighted simple games. However, in the context of QOBDDs, this restriction can also be justified by looking at WVGs as building blocks for general simple games. In this respect, as we have already mentioned in Section 2.3, it is well-known that every simple game has a vector-weighted representation with at most $|\mathcal{L}_{\max}|$ rules. Even though this result is more of theoretical interest, in practice, if a simple game is represented by multiple rules, then the rules often correspond to a certain paradigm. For instance:

- **Qualified majority or blocking minority.** For n players these rules look like $[Q; 1, \ldots, 1]$ where Q is usually greater than $\lceil n/2 \rceil$ for a qualified majority, and Q is usually near n to avoid a blocking minority.

- **Proportion.** For n players let $p_i \in [0,1]$ be player ith proportion of a measure, say population, contingent, area and so on, such that $\sum_{i=1,\ldots,n} p_i = 1$ and let $r \in [0,1]$ be a rate such as 50% or 66%. The weight w_i for player i is then chosen as the rounded value of $p_i L$ and the quota Q is chosen as the rounded value of rL where L is a sufficiently large positive integer, say 10 or 100, such that all the weights and the quota are nearly integers. This type of a rule is used very often in practice. For instance, the population is used in the Treaty of Nice, the US Electoral College and the German Bundesrat.

In this chapter we will show that the complexity of a simple game, which is made of rules, is determined by the complexity of these rules. And, as we have just seen, those rules often have a simple structure in practice. This is one of the main reasons, why QOBDDs are well suited to represent simple games, even if the number of players or the quota is huge and despite the fact that the size of QOBDDs in general grows rapidly. This chapter is structured as follows. The first section presents notation and elementary results for WVGs represented by QOBDDs. Based on that, Sections 4.2 and 4.3 present algorithms to obtain the QOBDD representation of a simple game from a weighted representation and multiple weighted representation with a formula, respectively. Parts of the chapter are based on the article Bolus (2011b).

4.1. QOBDDs representing Weighted Voting Games

In this section we present elementary results regarding the structure of a QOBDD that represents the winning coalitions of a weighted voting game.

Threshold functions, and therefore, WVGs, have been studied extensively in the 1960s; see Sheng (1969) for an overview and Coates and Lewis (1961) for an application. Most ideas in this section can be attributed to results stated then. However, at that time, OBDDs were unknown and it took 30 years before Hosaka, Takenaga, and Yajima (1994) (implicitly) applied some of the ideas to prove upper bounds for the size of QOBDDs representing threshold functions. Quite recently, Behle (2008) used QOBDDs to represent threshold functions in the context of 0-1 knapsack problems and thereby implicitly used some of the results, too. The aim of this section is to lay a solid foundation for the study of QOBDDs representing WVGs. The results in this section are used throughout the thesis.

$l_w(\mathcal{A}), u_w(\mathcal{A})$
$\Delta_w(\mathcal{A})$

Definition 4.1. For finite and maybe empty sets N, M with $N \subseteq M \subseteq \mathbb{N}$ and a weight function $w : M \to \mathbb{R}_{\geq 0}$ we define mappings $l_w, u_w, \Delta_w : 2^{2^N} \to \mathbb{R} \cup \{-\infty, \infty\}$ by

$$\begin{aligned} l_w(\mathcal{A}) &:= \sup\{w(S) \mid S \in 2^N \setminus \mathcal{A}\} \\ u_w(\mathcal{A}) &:= \inf\{w(S) \mid S \in \mathcal{A}\} \\ \Delta_w(\mathcal{A}) &:= u_w(\mathcal{A}) - l_w(\mathcal{A}) \end{aligned}$$

where $\mathcal{A} \subseteq 2^N$ and $\infty - (-\infty) = \infty$. The value $\Delta_w(\mathcal{A})$ is called the *gap of $\mathcal{A}$ w.r.t. w*. If there is no risk of confusion, the subscript w is omitted. □

If a vector $(w_1, \dots, w_n) \in \mathbb{R}^n_{\geq 0}$ is given instead of weight function, then we suppose that the weight function w is implicitly defined by $w(i) := w_i$ and the players are $N := \{1, \dots, n\}$.

In the context of a weighted voting game $[Q; w_1, \dots, w_n]$ with winning coalitions $\mathcal{W}$, the value $l_w(\mathcal{W})$ is the maximum weight of all losing coalitions or $-\infty$ if all coalitions are winning and $u_w(\mathcal{W})$ is the minimum weight of all winning coalitions or ∞ if all coalitions are losing. It holds $l(2^{\{1,\dots,n\}}) = -\infty$ and $u(\emptyset) = \infty$.

Later, we will use these mappings mainly for QOBDD nodes that represent sets of subsets. Let v be a QOBDD node with label i. We will always assume that the labels of the nodes in $\mathsf{V}(v)$ are $\{i, \dots, n\}$ for some n. Therefore, for a weight function w we will implicitly assume, that w has domain at least $\{i, \dots, n\}$. We define $l_w(v), u_w(v)$ and $\Delta_w(v)$ by $l_w(\mathsf{set}(v)), u_w(\mathsf{set}(v))$ and $\Delta_w(\mathsf{set}(v))$, respectively, and omit the subscript w if there is no risk of confusion. It holds $v = \mathbb{I}_i$ if $l(v) = -\infty$ and $v = \mathbb{O}_i$ if $u(v) = \infty$.

$l_w(v), u_w(v)$
$\Delta_w(v)$

Figure 4.1 shows an example using the vector of weights $(w_1, w_2, w_3) = (3, 3, 1)$. The values of l and u are annotated as half open intervals $(l(v), u(v)]$ or as an open interval if $u(v) = \infty$. The reason for the use of half-open intervals will become clear in Lemma 4.2.

The following statement provides a recursive formula to compute the values $l(v)$ and $u(v)$ for a QOBDD with root v bottom-up, starting at the sinks. It is a suggested to re-enact the values for l and u in Figure 4.1 again to memorize the formula.

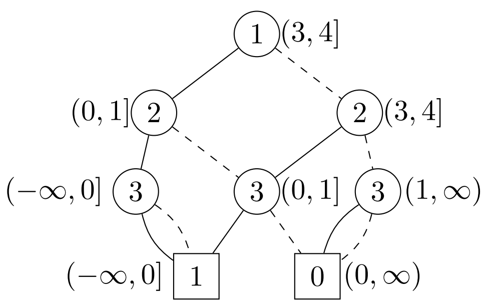


Figure 4.1.: Values of the functions l and u for weights $(w_1, w_2, w_3) = (3, 3, 1)$.

Lemma 4.1. *For a weight function w it holds*

$$l_w(\mathbb{I}) = -\infty, \quad u_w(\mathbb{I}) = 0, \quad l_w(\mathbb{O}) = 0, \quad u_w(\mathbb{O}) = \infty$$

and for any QOBDD *node $v \notin \{\mathbb{O}, \mathbb{I}\}$ with label i it holds:*

$$\begin{aligned} l_w(v) &= \max\{l_w(\mathsf{then}(v)) + w(i), l_w(\mathsf{else}(v))\} \\ u_w(v) &= \min\{u_w(\mathsf{then}(v)) + w(i), u_w(\mathsf{else}(v))\} \end{aligned}$$

where $\infty + x = \infty$ and $-\infty + x = -\infty$ for $x \in \mathbb{R}$.

Proof. The proof is by induction on the structure of the QOBDD. In the base case we consider the sinks. For the 0-sink $\mathbb{O}$ we have:

$$\begin{aligned} l(\mathbb{O}) &= \sup\{w(S) \mid S \in 2^{\emptyset} \setminus \mathsf{set}(\emptyset)\} = w(\emptyset) = 0 \\ u(\mathbb{O}) &= \inf\{w(S) \mid S \in \mathsf{set}(\mathbb{O})\} = \sup \emptyset = \infty\,. \end{aligned}$$

A similar argumentation can be used for the 1-sink $\mathbb{I}$.

Now, let v be an inner node with label i and assume that the statement holds for $\mathsf{then}(v)$ and $\mathsf{else}(v)$. In the induction step for $u(v)$ we have:

$$\begin{aligned} u(v) &= \inf\{w(S) \mid S \in \mathsf{set}(v)\} && \text{(Def. } u\text{)} \\ &= \inf\{w(S) \mid S \in \{T + i \mid T \in \mathsf{set}(\mathsf{then}(v))\} \cup \mathsf{set}(\mathsf{else}(v))\}\} && \text{(Def. } \mathsf{set}\text{)} \\ &= \min\{\inf_{T \in \mathsf{set}(\mathsf{then}(v))} w(T) + w(i), \inf_{T \in \mathsf{set}(\mathsf{else}(v))} w(T)\} && \text{(Rearrange)} \\ &= \min\{u(\mathsf{then}(v)) + w(i), u(\mathsf{else}(v))\}\,. && \text{(Ind. Hyp.)} \end{aligned}$$

The remaining case for $l(v)$ can be shown analogously. □

The next statements shed light on the relation between the quota of a weighted voting game with winning coalitions $\mathcal{W}$ and the values $l(\mathcal{W})$, $u(\mathcal{W})$ and $\Delta(\mathcal{W})$.

Lemma 4.2. *Let $[Q; w]$ be a weighted voting game with winning coalitions $\mathcal{W}$. Then $l_w(\mathcal{W}) < Q \leq u_w(\mathcal{W})$ and $\Delta_w(\mathcal{W}) > 0$.*

Proof. Let N denote the set of players. If no coalition is losing, that is, $\mathcal{W} = 2^N$, then $l(\mathcal{W}) = -\infty$ and $l(\mathcal{W}) < Q$ is trivially true. Otherwise, by Def. 4.1 there is a losing coalition with weight $l(\mathcal{W})$ and hence, $l(\mathcal{W}) < Q$.

Analogously, if all coalitions are losing, that is, $\mathcal{W} = \emptyset$, then $u(\mathcal{W}) = \infty$ and $Q \leq u(\mathcal{W})$ is true. Otherwise, there is a winning coalition with weight $u(\mathcal{W})$, so $Q \leq u(\mathcal{W})$. □

If a QOBDD with root r represents a weighted voting game with weight function w and players $N = \{1, \dots, n\}$, then every node $v \in \mathsf{V}(r)$ represents a WVG with weight function w and a subset of the players. The formal restriction of the domain of the weight function is omitted to keep the presentation easy. Unused weights are ignored.

Lemma 4.3. *Let v be a QOBDD node with label i and let I denote the labels of the nodes $\mathsf{V}(v)$, which are $I = \{i, \dots, n\}$. If $(I, \mathsf{set}(v))$ is a weighted voting game, then also $(I - i, \mathsf{set}(\mathsf{then}(v)))$ and $(I - i, \mathsf{set}(\mathsf{else}(v)))$ are WVGs.*

Proof. If $[Q; w]$ is a weighted representation of $(I, \mathsf{set}(v))$, then it can easily be seen that $[Q - w(i); w]$ is a weighted representation of $(I - i, \mathsf{set}(\mathsf{then}(v)))$ and $[Q; w]$ is a weighted representation of $(I - i, \mathsf{set}(\mathsf{else}(v)))$. Note that the weight of player i is not used. □

The following property will lead us to the idea of so-called flat QOBDDs in Section 5.2. See Figure 4.1 for an illustration.

Lemma 4.4. *Let v, v' be different QOBDD nodes with label i and let I denote the common set of labels of the nodes $\mathsf{V}(v)$ and $\mathsf{V}(v')$, which is $I = \{i, \dots, n\}$. If for a weight function $w : I \to \mathbb{R}_{\geq 0}$ both nodes represent WVGs $[Q_v; w]$ and $[Q_{v'}; w]$, respectively, then it holds:*

$$u(v) \leq l(v') \iff \mathsf{set}(v) \supset \mathsf{set}(v') \,.$$

Proof. Because we have WVGs, for $S \in 2^I$ it holds $S \in \mathsf{set}(v)$ if and only if $w(S) \geq Q_v$ and analogously for $\mathsf{set}(v')$ and $Q_{v'}$. For the direction "$\Longrightarrow$" assume $u(v) \leq l(v')$. If $\mathsf{set}(v') = \emptyset$, then the statement is true, because $v \neq v'$ and therefore, $\mathsf{set}(v) \neq \mathsf{set}(v')$. Otherwise let $S \in \mathsf{set}(v')$. By the definition of u, Lemma 4.2 and our assumption it holds

$$w(S) \geq u(v') > l(v') \geq u(v) \geq Q_v$$

and hence, $S \in \mathsf{set}(v)$ and $\mathsf{set}(v) \supseteq \mathsf{set}(v')$. The inclusion is proper due to $v \neq v'$.

For the direction "$\Longleftarrow$" assume $\mathsf{set}(v) \supset \mathsf{set}(v')$. Then there is a set $S \in \mathsf{set}(v) \backslash \mathsf{set}(v')$. From $S \in \mathsf{set}(v)$ it follows $u(v) \leq w(S)$ and from $S \notin \mathsf{set}(v')$ it follows $w(S) \leq l(v')$. Hence, $u(v) \leq w(S) \leq l(v')$. □

Let $(N, \mathcal{W})$ be a simple game and let w be a weight function for the players. The simple game has a weighted representation with w only if $l_w(\mathcal{W}) < u_w(\mathcal{W})$. If $l(\mathcal{W}) \geq u(\mathcal{W})$ then either $(N, \mathcal{W})$ is not weighted, or there is no weighted representation with w. For example, the QOBDD in Figure 4.1 represents the WVG $[4; 3, 3, 1]$. For the weights $(w_1, w_2, w_3) = (3, 2, 1)$, however, the root r of the QOBDD would have $l(r) = 3 = u(r)$. Hence, the represented simple game does not possess a weighted representation with weights $(3, 2, 1)$. Even though we have restricted the quota Q of a Wvg to be non-negative, it does not make a difference whether $Q = 0$ or $Q < 0$ if the weights are non-negative. For technical reasons, we will sometimes allow the quota to be negative as in the following statement.

Lemma 4.5. *Let v be* Qobdd *node with label i and let I denote the labels of $\mathsf{V}(v)$, which are $I = \{i, \dots, n\}$. Let $w : I \to \mathbb{R}_{\geq 0}$ be a weight function. For any $Q \in \mathbb{R}$ it holds:*

$$l(v) < Q \leq u(v) \iff \mathsf{set}(v) = \{S \subseteq I \mid w(S) \geq Q\}\,. \tag{4.1}$$

Proof. The direction "$\Longleftarrow$" is for free, because if the right-hand side in (4.1) is satisfied, then $(I, \mathsf{set}(v))$ is a Wvg with representation $[Q; w]$ and Lemma 4.2 can be used.

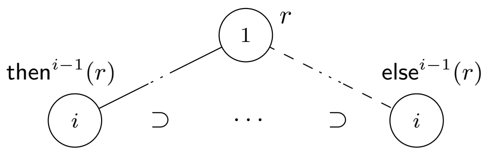
For the direction "$\Longrightarrow$" we assume $l(v) < Q \leq u(v)$. By Def. 4.1 for any $S \in \mathsf{set}(v)$ it holds $w(S) \geq u(v)$ and thus, $w(S) \geq Q$. Now, let $S \subseteq I$ such that $w(S) \geq Q$. Assume to the contrary that $S \notin \mathsf{set}(v)$. By the definition of l in Def. 4.1 then it holds $w(S) \leq l(v)$ and therefore, $w(S) < Q$. This is a contradiction, because $w(S) \geq Q$. □

In the remainder of this section we prove two simple properties regarding the size of Qobdds representing weighted voting games. Both will be used in Chapter 5.

Lemma 4.6. *For a weighted voting game $[Q; w_1, \dots, w_n]$ represented by a* Qobdd *with root r it holds for each $i \in \{1, \dots, n\}$:*

$$\min_{v \in \mathsf{V}_i(r)} u(v) \geq \max\{0, Q - w(\{1, \dots, i-1\})\}$$
$$\max_{v \in \mathsf{V}_i(r)} l(v) \leq \min\{Q - 1, w(\{i, \dots, n\})\}\,.$$

Proof. Let $v \in \mathsf{V}_i(r)$. Because there are no negative weights, it holds $u(v) \geq 0$ and thus, $\min_{x \in \mathsf{V}_i} u(x) \geq 0$. It holds $\max_{x \in \mathsf{V}_i} l(x) \leq w(\{i, \dots, n\})$, because in the worst case, $\{i, \dots, n\} \notin \mathsf{set}(v)$ and hence, $l(v) \leq w(\{i, \dots, n\})$.



For any node $v \in \mathsf{V}(r)$ the set $\mathsf{set}(v)$ is an up-set so $\mathsf{set}(\mathsf{then}(v)) \supseteq \mathsf{set}(\mathsf{else}(v))$. By induction starting at the root r it follows that $\mathsf{then}^{i-1}(r)$ is the maximal and $\mathsf{else}^{i-1}(r)$ is the minimal element in V_i w.r.t. $\supseteq$ and set, formally:

$$\forall v \in \mathsf{V}_i : \mathsf{set}(\mathsf{then}^{i-1}(r)) \supseteq \mathsf{set}(v) \supseteq \mathsf{set}(\mathsf{else}^{i-1}(r))\,.$$

By Lemma 4.4 it is rather easy to see:

$$\min_{v \in \mathsf{V}_i} u(v) = u(\mathsf{then}^{i-1}(r)) \quad \text{and} \quad \max_{v \in \mathsf{V}_i} l(v) = l(\mathsf{else}^{i-1}(r))\,. \tag{4.2}$$

With Lemma 4.5 it holds $Q \in \{l(r)+1, \dots, u(r)\}$ because the weights and the quota Q are integers. Hence, $l(r) \leq Q-1$ and $u(r) \geq Q$. Let I denote the set $\{1, \dots, i-1\}$. Again by induction and Lemma 4.1 it can be seen $u(\mathsf{then}^{i-1}(r)) \geq u(r) - w(I)$ by

$$u(r) \leq u(\mathsf{then}(r)) + w_1 \leq \cdots \leq u(\mathsf{then}^{i-1}(r)) + w(I) \tag{4.3}$$

and similarly, $l(\mathsf{else}^{i-1}(r)) \leq l(r)$ by

$$l(r) \geq l(\mathsf{else}(r)) \geq \cdots \geq l(\mathsf{else}^{i-1}(r))\,. \tag{4.4}$$

We can now conclude with

$$\min_{v \in \mathsf{V}_i} u(v) \overset{(4.2)}{=} u(\mathsf{then}^{i-1}(r)) \overset{(4.3)}{\geq} u(r) - w(I) \geq Q - w(I)$$

and

$$\max_{v \in \mathsf{V}_i} l(v) \overset{(4.2)}{=} l(\mathsf{else}^{i-1}(r)) \overset{(4.4)}{\leq} l(r) \leq Q-1\,. \qquad \square$$

The previous result can now be used in conjunction with the gap $\Delta(v)$ for a node v to establish an upper bound for the number of nodes on each level. Because the weights and the quota in weighted representations are non-negative integers, the gap is at least 1.

Lemma 4.7. *For a weighted voting game $[Q; w_1, \dots, w_n]$ represented by a* QOBDD *with root r it holds for each $i \in \{1, \dots, n\}$:*

$$|\mathsf{V}_i(r)| \leq \frac{\max\limits_{v \in \mathsf{V}_i(r)} l(v) - \min\limits_{v \in \mathsf{V}_i(r)} u(v)}{\min\limits_{v \in \mathsf{V}_i(r)} \Delta(v)} + 2\,.$$

Proof. Let $i \in \{1, \dots, n\} =: N$. Because $\mathsf{set}(r)$ is an up-set, it can be shown that if $\mathbb{I}_i \in \mathsf{V}_i$ then $\mathsf{then}^{i-1}(r) = \mathbb{I}_i$ and analogously, if $\mathbb{O}_i \in \mathsf{V}_i$ then $\mathsf{else}^{i-1}(r) = \mathbb{O}_i$. We define X by

$$X := \mathsf{V}_i \setminus \{\mathsf{then}^{i-1}(r), \mathsf{else}^{i-1}(r)\} \tag{4.5}$$

and the function $f : \mathsf{V}_i \to 2^{\mathbb{R} \cup \{-\infty, \infty\}}$ by $v \mapsto \{x \in \mathbb{R} \mid l(v) < x \leq u(v)\}$. Because all nodes $v \in \mathsf{V}_i$ expect $\mathbb{O}_i, \mathbb{I}_i$ have finite values for $l(v)$ and $u(v)$ and thus, $\Delta(v)$, we have $f(v) \subseteq \mathbb{R}$ for $v \in X$. Because all weights $w_1, \dots, w_n$ are non-negative integers, it furthermore holds $f(v) \subseteq \mathbb{N}_0$ for $v \in X$. By Lemma 4.4 for all $v \in \mathsf{V}_i$ it follows $\min_{x \in \mathsf{V}_i} u(x) \leq l(v)$ and $u(v) \leq \max_{x \in \mathsf{V}_i} l(x)$ and therefore we get:

$$\bigcup_{v \in X} f(v) \subseteq \{\min_{v \in \mathsf{V}_i} u(v) + 1, \dots, \max_{v \in \mathsf{V}_i} l(v)\}\,. \tag{4.6}$$

For any two different nodes $v, v' \in \mathsf{V}_i$, by Lemma 4.4 we have $f(v) \cap f(v') = \emptyset$ and thus:

$$|\bigcup_{v \in X} f(v)| = \sum_{v \in X} |f(v)|\,.$$

Applying this to (4.6) yields:

$$\sum_{v \in X} |f(v)| \leq \max_{v \in \mathsf{V}_i} l(v) - \min_{v \in \mathsf{V}_i} u(v)\,. \tag{4.7}$$

By the definition of Δ for each $v \in X$ it holds $\Delta(v) = |f(v)|$. Therefore:

$$|\mathsf{V}_i| - 2 \leq |X| = |\sum_{v \in X} \frac{|f(v)|}{\Delta(v)}| \leq \sum_{v \in X} \frac{|f(v)|}{\min_{z \in \mathsf{V}_i} \Delta(z)} \leq \frac{\sum_{v \in X} |f(v)|}{\min_{v \in \mathsf{V}_i} \Delta(v)}\,.$$

The missing step is the application of (4.7). □

In the next section we show how the QOBDD for a weighted voting game can be obtained from a weighted representation.

4.2. From Weighted Voting Games to QOBDDs

In this section we present an output sensitive algorithm for building the QOBDD representation of a weighted voting game. The algorithm is used later as a building block to build QOBDDs for arbitrary simple games. The algorithm has originally been presented by Behle (2008) in the context of 0-1 knapsack problems and the optimal variable ordering problem for QOBDDs representing threshold functions. Its theory, however, goes back to at least Coates and Lewis (1961), whose results for linear separable functions can be used to prove the optimality of the caching strategy, which is at the heart of the algorithm. If a QOBDD with root r is built from a weighted voting game, then the algorithm has (expected) running time $\mathcal{O}(\mathsf{size}(r) \log \mathsf{width}(r))$ if AVL trees are used for the caches. Whether the running time is expected or not depends on the implementation of ite to create QOBDD nodes.

We start with a naive approach for illustration. Let $[Q; w_1, \ldots, w_n]$ be a WVG. Its QOBDD representation can be obtained by the following recursive function f where i is the current player (initially 1) and q is the current remaining quota (initially Q):

$$f(i,q) = \begin{cases} \mathbb{O} & \text{if } i > n \text{ and } q > 0 \\ \mathbb{I} & \text{if } i > n \text{ and } q \leq 0 \\ \mathsf{ite}(i, f(i+1, q-w_i), f(i+1,q)) & \text{otherwise} \end{cases}\,.$$

Intuitively, the recursive call $f(i+1, q-w_i)$ corresponds to the case in which player i says “yes”, so that in the remaining game with players $i+1, \ldots, n$ the quota is reduced by the weight of player i. In the other recursive call $f(i+1,q)$, player i says “no” and the quota remains unchanged. If $i > n$, then no player is left in the remaining game. The correctness can be seen as follows.

Proposition 4.8. *For $i \in \{1, \ldots, n+1\}$ and $q \in \mathbb{Z}$ it holds:*

$$\mathsf{set}(f(i,q)) = \{S \subseteq \{i, \ldots, n\} \mid w(S) \geq q\}\,. \tag{4.8}$$

Proof. The proof is by induction on the number of remaining players $n+1-i$ and the induction starts at $i = n+1$. Then no players remain and in case $q \leq 0$ we have $\mathsf{set}(\mathbb{I}) = 2^{\emptyset} = \{S \in 2^{\emptyset} \mid w(S) \geq q\}$. In case $q > 0$ we have $\mathsf{set}(\mathbb{O}) = \emptyset = \{S \in 2^{\emptyset} \mid w(S) \geq q\}$.

For the induction hypothesis let $i \in \{1, \dots, n\}$ and assume that (4.8) is true for $i+1$ and any q. Because $i \leq n$ we only consider the third case of f. For $S \subseteq \{i, \dots, n\}$ it holds:

$$\begin{aligned}
& S \in \mathsf{set}(f(i,q)) \\
\iff & S \in \mathsf{set}(\mathsf{ite}(i, f(i+1, q-w_i), f(i+1,q))) && \text{(Def. } f\text{)} \\
\iff & \big(i \in S \land S - i \in \mathsf{set}(f(i+1, q-w_i))\big) \\
& \lor \big(i \notin S \land S \in \mathsf{set}(f(i+1,q))\big) && \text{(Def. } \mathsf{set}, \mathsf{ite}\text{)} \\
\iff & \big(i \in S \land w(S-i) \geq q - w_i\big) \lor \big(i \notin S \land w(S) \geq q\big) && \text{(Ind. Hyp.)} \\
\iff & w(S) \geq q\,.
\end{aligned}$$

The last equivalence it due to $w(S-i) + w_i = w(S)$ if $i \in S$. □

This approach has the obvious shortcoming that it requires exponentially many recursive calls. As in the case of the knapsack problem, dynamic programming could be used to reduce the running time to something like $\mathcal{O}(nQ)$. The main idea for this approach would be to store and reuse the return value of f for arguments i, q, because the algorithm returns the same QOBDD node for the same input. But, it sometimes returns the same node for different inputs as well. For example, consider the WVGs $[4; 3, 2]$ and $[5; 3, 2]$. Even though the quotas are different, the games have the same winning coalitions.

In Lemma 4.5 in the previous section we have seen when WVGs with the same weights $w_i, \dots, w_n$ have the same winning coalitions $\mathcal{W}$. For any quota $Q \in \mathbb{R}$ it holds

$$l_w(\mathcal{W}) < Q \leq u_w(\mathcal{W}) \iff \mathcal{W} = \{S \subseteq \{i, \dots, n\} \mid w(S) \geq Q\}$$

where w is the implicitly defined weight function with $w(i) = w_i$. We will use this result to avoid multiple computations in f here. In Lemma 4.1 we have shown, how the values $l_w(v)$ and $u_w(v)$ can be computed recursively for a QOBDD node v. The new algorithm, called *BuildRec*, that incorporates this computation, is listed in Algorithm 2. To argue that the computation of the values for l_w and u_w is correct, we will ignore the lines 3 and 9 and, respectively, the operations *lookup* and *insert* for now.

In the remaining section we will assume that $[Q; w_1, \dots, w_n]$ is a weighted voting game with players $N = \{1, \dots, n\}$ for which the weight function w is implicitly defined. Furthermore, we will assume that in the context of an algorithm the weights are in a global scope.

As for the recursive function f, *BuildRec* is initially called with i as 1 and q as Q. The following lemma states the correctness without the reuse of previously computed results. The latter is taken into account in Theorem 4.10 below.

Lemma 4.9. *Let $i \in \{1, \dots, n+1\}$ and let $q \in \mathbb{Z}$ be a remaining quota. If we ignore the lines 3 and 9 in Algorithm 2, then for $(v, x, y) := BuildRec(i, q)$ it follows $x = l_w(v)$, $y = u_w(v)$ and $\mathsf{set}(v) = \{S \in 2^{\{i, \dots, n\}} \mid w(S) \geq q\}$.*

Algorithm 2 *BuildRec*(i, q)

```
1:  if $i > n$ and $q > 0$ then return $(\mathbb{O}, 0, \infty)$
2:  else if $i > n$ and $q \leq 0$ then return $(\mathbb{I}, -\infty, 0)$
3:  else if $e \neq \perp$ then return $e$ where $e = \mathit{lookup}(i, q)$
4:  else
5:      $(v_T, x_T, y_T) \leftarrow \mathit{BuildRec}(i + 1, q - w_i)$
6:      $(v_E, x_E, y_E) \leftarrow \mathit{BuildRec}(i + 1, q)$
7:      $(x, y) \leftarrow (\max\{x_T + w_i, x_E\}, \min\{y_T + w_i, y_E\})$
8:      $v \leftarrow \mathsf{ite}(i, v_T, v_E)$
9:      $\mathit{insert}(i, (v, x, y))$
10:     return $(v, x, y)$
```

Proof. The correctness of the node v is a direct consequence of Proposition 4.8. The values x and y are correct by Lemma 4.1. □

What remains is to reuse previously computed results. We use n balanced search trees $T_1, \ldots, T_n$ where T_i is used if the current player in *BuildRec* is i. The search trees are in a global scope and it is assumed that they are initially empty. We use AVL trees (Adelson-Velskii and Landis 1963) for the balanced search trees, which can insert and look up elements in time $\mathcal{O}(\log m)$ where m is the number of elements in the tree. The elements that we store in the trees are triples (v, x, y) where x, y are integers and v is a QOBDD node. A search tree compares two entries (v_1, x_1, y_1) and (v_2, x_2, y_2) by $y_1 < y_2$. The *insert*$(i, (v, x, y))$ call in line 9 of Algorithm 2 is assumed to insert (v, x, y) into T_i while *lookup*(i, q) is assumed to perform a lookup in T_i. For a lookup, a single integer argument q is used and an element (v, x, y) is returned if and only if $x < q \leq y$. In case that no such element exists, $\perp$ is returned, which has the meaning of *undefined*.

From now on, we assume that the operations *lookup* and *insert* are implemented as just described. The correctness of *BuildRec* with caching is shown next.

Theorem 4.10. *For $i \in \{1, \ldots, n+1\}$, $I := \{i, \ldots, n\}$, remaining quota $q \in \mathbb{Z}$ and $(v, x, y) :=$ BuildRec(i, q) it holds $\mathsf{set}(v) = \{S \in 2^I \mid w(S) \geq q\}$, $x = l(v)$ and $y = u(v)$.*

Proof. The proof is by induction and similar to that of Lemma 4.9. The case for $i > n$ is omitted, because it is the same as in Lemma 4.9. Assume $i \leq n$ and the statement holds for $i + 1$. The case where *lookup*$(i, q) = \perp$ is analogous to that in Lemma 4.9 again. Hence, assume *lookup*$(i, q) = (v, x, y) \neq \perp$. Then there is a $p \in \mathbb{Z}$ for which (v, x, y) has been inserted into T_i in the call *BuildRec*(i, p). From the correctness of that call it holds $\mathsf{set}(v) = \{S \in 2^I \mid w(S) \geq p\}$, $x = l(v)$ and $y = u(v)$. By the definition of *lookup*(i, q) it holds $x < q \leq y$ and hence, $l(x) < q \leq u(v)$. By using Lemma 4.5 it follows $\mathsf{set}(v) = \{S \in 2^I \mid w(S) \geq q\}$. □

To illustrate this result, consider the WVGs $[5; 3, 2]$ and $[4; 3, 2]$ again. As we have already mentioned, both games have the same set of winning coalitions and therefore, the same QOBDD representation, say, the root is r. It holds $l(r) = 3$, $u(r) = 5$ and therefore, $l(r) < 4 \leq u(r)$ and also $l(r) < 5 \leq u(r)$.

For the running time, we assume that $T_1, \ldots, T_n$ are initially empty and that the QOBDD representation of $[Q; w_1, \ldots, w_n]$ is being built. The root of the QOBDD is denoted by r. The initial call is *BuildRec*$(1, Q)$.

Lemma 4.11. *When (v, x, y) is inserted into T_i in line 9 in BuildRec(i, q), then for each $(v', x', y') \in T_i$ it holds $v \neq v'$.*

Proof. At the moment when (v, x, y) is inserted into T_i it holds *lookup*$(i, q) = \perp$. Otherwise we would not be in line 9. For that reason, by the definition of the operation *lookup* there is no element $(v', x', y') \in T_i$ with $x' < q \leq y'$. Assume to the contrary that there is an element $(v', x', y') \in T_i$ with $v = v'$. By the correctness of the algorithm it holds $\mathsf{set}(v) = \{S \subseteq \{i, \ldots, n\} \mid w(S) \geq q\}$, $x = l(v)$ and $y = u(v)$. For the same reason it holds $x' = x$ and $y = y'$. Due to Lemma 4.2 it now holds $x' < q \leq y'$ which is a contradiction to *lookup*$(i, q) = \perp$. □

Because for each QOBDD node v in $\mathsf{V}(r)$ there is an entry in $T_{\mathsf{var}(v)}$, it follows:

Corollary 4.12. *For $i = 1, \ldots, n$ it holds $|T_i| = |\mathsf{V}_i(r)|$ and $\sum_{i=1}^n |T_i| = \mathsf{size}(r)$.* □

There is one initial call to *BuildRec* and for each element in $T_1, \ldots, T_n$ there are two recursive calls. Hence, the number of calls to *BuildRec* is exactly:

$$1 + 2\sum_{i=1}^{n} |T_i| = 1 + 2\mathsf{size}(r)\,.$$

A call to *BuildRec*(i, q) with $i > n$ for any $q \in \mathbb{Z}$ takes time $\mathcal{O}(1)$. Taking into account the time to lookup and insert an element into a search tree, every other call with $i \leq n$ takes time

$$\mathcal{O}(\log |T_i|) = \mathcal{O}(\log |\mathsf{V}_i(r)|) \leq \mathcal{O}(\log \mathsf{width}(r))\,.$$

Additionally, a node is created exactly $\mathsf{size}(r)$ times using the operation ite, which costs either deterministic or expected constant time, depending on the implementation. In conclusion, the running time is bounded in the size of the result QOBDD. Such an algorithm is called *output sensitive*. We note this in the following theorem for future reference.

Theorem 4.13. *The QOBDD with root r for $[Q; w_1, \ldots, w_n]$ can be build in (expected) time $\mathcal{O}(\mathsf{size}(r) \log \mathsf{width}(r))$. Whether the running time is expected or not depends on the implementation of the operation ite.* □

Algorithm 3 *Build*$([Q; w_1, \ldots, w_n])$

```
    initialize the AVL trees $T_1, \ldots, T_n$
    set up the variables in the global scope
    $(r, x, y) \leftarrow BuildRec(1, Q)$
    return $r$
```

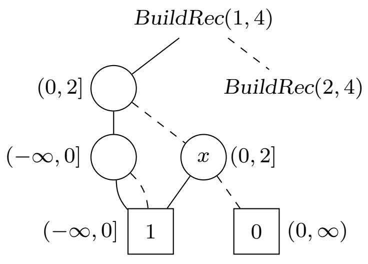


Figure 4.2.: Example of algorithm *BuildRec*.

The algorithm that builds the QOBDD for a WVG is listed in Algorithm 3. It accepts a weighted representation $[Q; w_1, \ldots, w_n]$ and it returns the root of the QOBDD for the game. The running time of algorithm *Build* has been stated in Theorem 4.13. We illustrate the approach in the following example.

Example 4.1. We build the QOBDD for the WVG $[4; 3, 2, 2]$ with winning coalitions $\{AB, AC, BC, ABC\}$. In what follows, we use the letters A, B, C and the numbers $1, 2, 3$ interchangeably.

Figure 4.2 shows our point of departure. At the very beginning, *BuildRec*$(1, 4)$ is called by *Build*$([4; 3, 2, 2])$. Because all the search trees are initially empty, the algorithm *BuildRec* recursively calls *BuildRec*$(2, 4 - w(A))$ in line 5. This is the situation in which player A says "yes" and joins the coalition. The remaining quota is reduced by $w(A)$ and A withdraws from the remaining game. Thus, the recursive call considers the game $[4 - w(A); w(B), w(C)]$. After some recursive calls, *BuildRec* returns and *BuildRec*$(2, 4)$ is called in line 6. This is the situation in which A says "no" and the remaining quota does not change. Again, there is no entry in the corresponding search tree for this call. We first consider the situation in which player B says "yes" and joins the currently empty coalition (A said "no"). The recursive call is *BuildRec*$(3, 4 - w(B))$ in line 5, i.e., the game $[4 - w(B); w(C)]$ is considered. This time, there is an entry in the search tree for the remaining quota $4 - w(B) = 2$ on level 3 (player C). The algorithm returns the node x and its values $l(x)$ and $u(x)$ which are 0 and 2, respectively. Back on level 2 (player B), it remains the case when B says "no". □

The fact that the procedure is output sensitive w.r.t. the result QOBDD is one of its strengths. If we find a class of weighted voting games with small QOBDD representations, then, without further proof, we have a "fast" algorithm to build the QOBDD for a game in that class. For instance, in Section 5.4 we will show that homogeneous simple games have QOBDD representations with size $\mathcal{O}(n^2)$ and width $\mathcal{O}(n)$ if the weights are ordered by $w_1 \geq \cdots \geq w_n$. For such a game we can therefore build the QOBDD representation in time $\mathcal{O}(n^2 \log n)$. The size of QOBDDs for general WVGs and classes of WVGs is discussed in Chapter 5.

We have used balanced search trees and AVL trees, respectively, in *BuildRec* for the caches. These added a factor of $\log \mathsf{width}(r)$ to the running time where r is the result QOBDD. It is an open question if this factor can be reduced or avoided.

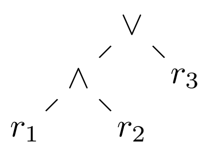


Figure 4.3.: Expression tree for the formula $(1 \wedge 2) \vee 3$ and QOBDDs with roots r_1, r_2, r_3 for the rules.

4.3. From Multiple Weighted Voting Games to QOBDDs

In Section 2.3 we have seen, that weighted representations are sometimes inconvenient and sometimes insufficient to model a simple game in practice. As an example we have discussed the UN Security Council, which has the weighted representation

$$[39; \underbrace{7, \ldots, 7}_{\text{5-times}}, \underbrace{1, \ldots, 1}_{\text{10-times}}]$$

but which can more intuitively be represented by

$$[5; \underbrace{1, \ldots, 1}_{\text{5-times}}, \underbrace{0, \ldots, 0}_{\text{10-times}}] \wedge [9; \underbrace{1, \ldots, 1}_{\text{15-times}}] .$$

The structure of the game is apparent from the latter and this representation is therefore preferable. In this section, we present a general purpose approach to obtain the QOBDD from a multiple weighted representation with a formula. By using the binary synthesis for QOBDDs and the algorithm from the previous section this is nearly for free.

Throughout this section let $(\varphi, \{g_1, \ldots, g_m\})$ be a multiple weighted representation with formula φ, $n \geq 1$ players and $m \geq 1$ rules $g_1, \ldots, g_m$. The kth rule is $g_k = [Q_k; w_{k,1}, \ldots, w_{k,n}]$. It is a two stage process to obtain the QOBDD for $(\varphi, \{g_1, \ldots, g_m\})$:

1. For each $k \in \{1, \ldots, m\}$ obtain the QOBDD with root r_k for g_k by using the algorithm *Build* from the previous section.

2. Transform the formula φ into an expression tree where the variables $1, \ldots, m$ in the formula become the QOBDD nodes $r_1, \ldots, r_m$ in the expression tree. Then use the algorithm apply from Section 3.4 to obtain the QOBDD for $(\varphi, \{g_1, \ldots, g_m\})$ using the tree.

Example 4.2. Consider the multiple weighted representation with rules g_1, g_2, g_3 and formula $\varphi = (1 \wedge 2) \vee 3$. After the first step we have the QOBDD roots r_1, r_2, r_3 such that $\mathsf{set}(r_k)$ are the winning coalitions of g_k for $k = 1, 2, 3$. The expression tree for φ is shown in Figure 4.3. In the second step, the QOBDD for $r_1 \wedge r_2$ is computed first, using apply with $\wedge$. Afterwards apply with $\vee$ is used with the result of the previous application for $r_1 \wedge r_2$ and r_3. The result QOBDD now represents the simple game associated with $(\varphi, \{g_1, g_2, g_3\})$. □

We omit a formal proof for the correctness of this approach, because it can easily be derived from the correctness of the synthesis operation apply in Section 3.4 and that of *Build* in Theorem 4.10. The more interesting part is the running time and the size of the result QOBDD.

Theorem 4.14. *Let $g_1, \dots, g_m$ be the $m \geq 2$ rules of a multiple weighted representation with formula $\varphi = \alpha \diamond \beta$ where $\diamond \in \{\wedge, \vee\}$. W.l.o.g. we assume that each variable in α (resp. β) is less or equal (resp. greater) than j for some $j \in \{1, \dots, m-1\}$. The* QOBDD *for the associated simple game can be computed in expected time*

$$\mathcal{O}(\sum_{k=1}^{m} \mathit{size}(r_k) \log \mathit{width}(r_k) + n(\prod_{k=1}^{m} \mathit{width}(r_k) + (j-1) \prod_{k=1}^{j} \mathit{width}(r_k) + (m-j-1) \prod_{k=j+1}^{m} \mathit{width}(r_k)))$$

where $r_1, \dots, r_m$ are the QOBDD*s representing $g_1, \dots, g_m$ and the result* QOBDD *has size*

$$\sum_{i=1}^{n} \prod_{k=1}^{m} \mathit{width}_i(r_k) \tag{4.9}$$

and thus, $n \prod_{k=1}^{m} \mathit{width}(r_k)$.

Proof. To obtain the QOBDD r_k for the game g_k, $k = 1, \dots, m$, requires expected time $\mathcal{O}(\mathsf{size}(r_k) \log \mathsf{width}(r_k))$ by Theorem 4.13. The overall expected time to obtain all the QOBDDs $r_1, \dots, r_m$ is therefore $\mathcal{O}(\sum_{k=1}^{m} \mathsf{size}(r_k) \log \mathsf{width}(r_k))$. The remaining parts are a direct consequence of Theorem 3.5 when the formula φ is used for the expression tree as described above. The partitioning of the variables in φ by j is used as upper bound for the number of operators in the left and right children of the expression tree in Theorem 3.5. □

The upper bound for the running time can be improved when we have a vector-weighted representation.

Theorem 4.15. *Let $g_1, \dots, g_m$ be the rules of a m-vector-weighted representation. The* QOBDD *for the associated simple game can be computed in expected time*

$$\mathcal{O}(\sum_{k=1}^{m} \mathit{size}(r_k) \log \mathit{width}(r_k) + m \log m + n(1 + \frac{m}{W}) \prod_{k=1}^{m} \mathit{width}(r_k)) \tag{4.10}$$

where $r_1, \dots, r_m$ are the QOBDD*s representing $g_1, \dots, g_m$ and W is the maximum width $\max_{k=1,\dots,m} \mathit{width}(r_k)$.*

Proof. As in the previous Theorem 4.14, we compute the QOBDDs $r_1, \dots, r_m$ for all rules in expected time $\mathcal{O}(\sum_{k=1}^{m} \mathsf{size}(r_k) \log \mathsf{width}(r_k))$.

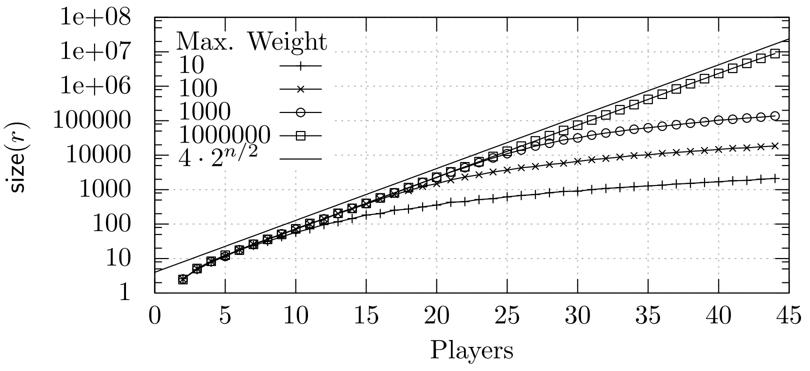


Figure 4.4.: Size of the QOBDDs for randomly generated WVGs with quota chosen as 50% of the sum of weights and 100 samples per point.

Let π be an ordering of the QOBDDs such that

$$\mathsf{width}(\pi(1)) \leq \cdots \leq \mathsf{width}(\pi(m))\,.$$

The width of r_k can be obtained in $\mathcal{O}(\mathsf{size}(r_k))$ steps by a single traversal of r_k. It takes additional $\mathcal{O}(m \log m)$ steps to obtain the ordering π. Now, Theorem 3.6 can be applied to $r_1, \ldots, r_m$ and the ordering π. The result QOBDD can be computed in expected time $\mathcal{O}(n(1 + m/W) \prod_{k=1}^{m} \mathsf{width}(r_k))$. □

In Section 5.3 we will see that the QOBDD for a WVG $[Q; w_1, \ldots, w_n]$ has width at most $Q + 1$. Therefore, for m rules with quotas $Q_1, \ldots, Q_m$ the width of the result QOBDD is bounded by

$$(Q_1 + 1) \cdots (Q_m + 1)\,. \tag{4.11}$$

If for each Q_k we know that Q_k is polynomially bounded in n and m is a constant, then the size of the result is also polynomially bounded in n and we therefore have an expected polynomial time algorithm to build the QOBDD for the associated simple game. In general, however, (4.11) grows exponentially in m. As already mentioned in the introduction to this chapter, the number of rules in real world voting systems is rarely above 3, though.

4.4. Conclusions

In this chapter we have studied the structure of QOBDDs representing weighted voting games. Based on that, we have developed an output sensitive algorithm to obtain the QOBDD for a WVG with running time $\mathcal{O}(\mathsf{size}(r) \log \mathsf{width}(r))$ where r is the result QOBDD. The algorithm uses balanced search trees to identify results that we have already computed earlier. This adds the factor of $\log \mathsf{width}(r)$ to the running time, when AVL trees are used in the implementation. It would be very interesting to see if this factor can be reduced.

Some real world simple games cannot be represented by WVGs and sometimes it is more convenient to use more than one rule to represent a voting system as in the case of the UN Security Council. Therefore, it is important to be able to obtain the QOBDD representation from multiple weighted representations with a formula. We have used binary synthesis for QOBDDs to this end. This approach permits to obtain the QOBDD for any simple game, because each simple game has a vector-weighted representation.

	Players	Quota(s)	QOBDD size
Canadian Constitution (1995)	10	$(7, 50)$	37
Canadian Constitution (2005)	10	$(7, 50)$	44
US Electoral College (2004-2008)	51	270	4558
German Bundesrat (2012)	16	35	162
Treaty of Nice	27	$(255, 14, 620)$	635
Treaty of Lisbon	27	$(15, 32400, 24)$	4134
UN Security Council	15	$(9, 5)$	53
US Federal Legislative System	537	$(1, 51, 128, 1, 50, 67, 290)$	141 650
International Monetary Fund (85%, 2009)	186	1 884 478	15 712 104

Table 4.1.: QOBDD sizes for some real world simple games.

Figure 4.4 shows the size of QOBDDs for random WVGs with different maximum weights. We have obtained the sizes by a small program. As can be seen from the figure, the size of random WVGs grows exponentially if the number of players increases and if the weights are large enough. If the weights of the players are small and therefore, the quota is small, then the size of the QOBDD is small, too. The connection between the quota and the size is studied in Section 5.3. We will also see, that a QOBDD for a WVG has size at most $\mathcal{O}(2^{n/2})$. Fortunately, real world voting systems usually have a simple structure and therefore QOBDDs of moderate size. The size of some real world voting systems is listed in Table 4.1. For instance, even though the QOBDD for the International Monetary Fund is rather large, in comparison to the worst case size it is still small. The same holds for the US Federal Legislative System which has 7 rules and 537 players.

5. Size and Structure of QOBDDs for Classes of Simple Games

In Section 4.1 we have seen that QOBDDs for WVGs have structural properties that could be used in Section 4.2 to develop an output sensitive algorithm to build the QOBDD for a WVG. In this chapter we study the structure and size of QOBDDs for some more classes of simple games. The main contributions in this chapter are:

1. The definition of so-called flat QOBDDs and the proof that already flat QOBDDs have size at most $\mathcal{O}(2^{n/2})$.
2. The proof that a QOBDD representing a weighted voting game with quota Q has size $\mathcal{O}(\max\{n - \log Q, 1\}Q)$. So far, the best known upper bounds are $\mathcal{O}(nQ)$ and $\mathcal{O}(2^{n/2})$. Both will be discussed below.
3. The proof that QOBDDs representing directed and homogeneous simple games have size $\mathcal{O}(n^2)$, but that there are homogeneous simple games that have QOBDD representations with exponential size for at least one ordering of the players and weights, respectively.
4. A very similar result for WVGs with sequential weights.

Section 5.4 is based on the article Bolus (2011c).

5.1. Complete and Directed Simple Games

Intuitively, in a complete simple game every pair of players can be compared by the desirability relation on individuals $\preceq_I$. Having a complete simple game is very common in practice. Not only because every WVG is complete, but also because if more than one rule is used, then the weights of a player are often related. For instance, if there are two rules and one is the amount of money that everybody pays for a system (e.g., the European Union) and the other is the proportion of the population, then usually both numbers correlate, because countries with a large population pay more money.

Complete simple games have been studied extensively. Carreras and Freixas (1996) have presented a characterization of complete simple games by so-called models of shift-minimal winning coalitions; see also Section 6.8. Kurz and Tautenhahn (2012) have studied the Dedekind problem for complete simple games and they have shown that complete simple games can be counted and enumerated by cliques in a suitable graph, a problem for which there are sophisticated algorithms already (Östergård 2002). For instance, the number of complete simple games without the cases $\mathcal{W} = \emptyset$ and $\mathcal{W} = 2^N$ for 9 players (up to isomorphism) is 284,432,730,174.

In this short section, we show two auxiliary statements, that are used later in Section 6.5 to simplify the computation of the QOBDDs for the minimal winning and the maximal losing coalitions. The first one shows a simple statement for up- and down-sets. A proof is omitted, because the statement can easily be verified. Note that the subscripts i and $\neg i$ have precedence over the application of min and max, respectively.

Lemma 5.1. *Let $i \in N$ and let $\mathcal{A} \subseteq 2^{\{i,\dots,n\}}$. If $\mathcal{A}$ is an up-set, then*

$$\min \mathcal{A} = \{S + i \mid S \in (\min \mathcal{A}_i) \setminus \min \mathcal{A}_{\neg i}\} \cup \min \mathcal{A}_{\neg i}$$

and if $\mathcal{A}$ is a down-set then

$$\max \mathcal{A} = \{S + i \mid S \in \max \mathcal{A}_i\} \cup ((\max \mathcal{A}_{\neg i}) \setminus \max \mathcal{A}_i). \qquad \square$$

Our second statement exploits a structural property of directed simple games.

Lemma 5.2. *Let $i \in N$ and let $\mathcal{A} \subseteq 2^{\{i,\dots,n\}}$ be a set of coalitions. If $(\{i,\dots,n\}, \mathcal{A})$ is a directed simple game with $\mathcal{A}_i \neq \mathcal{A}_{\neg i}$ then*

$$(\min \mathcal{A}_i) \setminus \min \mathcal{A}_{\neg i} = \min \mathcal{A}_i$$

and for $\overline{\mathcal{A}}$ being $2^{\{i,\dots,n\}} \setminus \mathcal{A}$ it holds

$$\max(\overline{\mathcal{A}})_{\neg i} \setminus \max(\overline{\mathcal{A}})_i = \max(\overline{\mathcal{A}})_{\neg i}\,.$$

Proof. We only show the first claim for the minimal subsets. The remaining claim can be shown similarly. Assume to the contrary that there is a set $S \in \min \mathcal{A}_i \cap \min \mathcal{A}_{\neg i}$. Then we have that $i \notin S$ and that $\mathcal{A}$ is an up-set. It also holds $\emptyset \neq S$, because otherwise $\emptyset \in \mathcal{A}_{\neg i}$ would imply $2^{\{i+1,\dots,n\}} = \mathcal{A}_{\neg i}$ and due to the up-set property $\mathcal{A}_{\neg i} \subseteq \mathcal{A}_i$, both sets were equal in contradiction to $\mathcal{A}_{\neg i} \neq \mathcal{A}_i$. Hence, there is a player j in S. Since the game is directed, it holds $j \preceq_I i$ and thus, $(S - j) + i \in \mathcal{A}$ and $S - j \in \mathcal{A}_i$. This is a contradiction to the choice of S from $\min \mathcal{A}_i$. $\square$

By this result, for a directed simple game $(\{i,\dots,n\}, \mathcal{A})$ the equations in Lemma 5.1 can be rewritten to:

$$\begin{aligned}\min \mathcal{A} &= \{S + i \mid S \in \min \mathcal{A}_i\} \cup \min \mathcal{A}_{\neg i}\,,\\ \max \overline{\mathcal{A}} &= \{S + i \mid S \in \max(\overline{\mathcal{A}})_i\} \cup \max(\overline{\mathcal{A}})_{\neg i}\,.\end{aligned}$$

As already mentioned, we will come back to this result in Section 6.5.

5.2. Simple Games with flat QOBDDs

Hosaka, Takenaga, and Yajima (1994) have shown, that every QOBDD for a weighted voting game, regardless of the ordering of the players, has size at most $\mathcal{O}(2^{n/2})$. This seminal work is the origin of some interesting complexity results in the context of simple games. For instance, in Section 6.7 we will develop an algorithm to compute the so-called Banzhaf power index (Banzhaf 1965) of all players in time $\mathcal{O}(\mathsf{size}(r))$ where the QOBDD r represents a simple game. Hence, by the previous observation and the complexity of the algorithm to build the QOBDD for a weighted voting game in Section 4.2, we have an $\mathcal{O}(n2^{n/2})$ algorithm for this problem. This is a factor n better than the algorithm by Klinz and Woeginger (2005) for this problem.

The result by Hosaka et al. (1994) is based on the values $l(v)$ and $u(v)$ and uses a result similar to Lemma 4.4 to obtain an ordering of the nodes on a level in the QOBDD. In this section we show, that much weaker conditions have to be fulfilled in order to obtain this bound. Throughout this section let r represent a simple game $(N, \mathcal{W})$ with players $N = \{1, \dots, n\}$.

Definition 5.1. For nodes u, v with $\mathsf{var}(u) = \mathsf{var}(v)$ we define $u \subset v$ and $u \subseteq v$ by $\mathsf{set}(u) \subset \mathsf{set}(v)$ and $\mathsf{set}(u) \subseteq \mathsf{set}(v)$, respectively. We say a QOBDD with root r is *flat* if for each level $i \in N$ the set $\mathsf{V}_i(r)$ is a chain[1] w.r.t. $\subset$. □ $u \subset v, u \subseteq v$

Figure 5.1 shows a non-flat (left) and a flat QOBDD (right) for the same simple game

$$[2; 1, 2, 0, 1] \wedge [3; 2, 1, 2, 1]$$

but different orderings of the players. To see that the QOBDD with ordering A, B, C, D for the players is not flat, consider the nodes u, v. It neither holds $u \subset v$ nor $u \supset v$, because $\mathsf{set}(u) = \{D, CD\}$ and $\mathsf{set}(v) = \{C, CD\}$. However, if we interchange the players B and C in the ordering for the QOBDD, then the result on the right-hand side in Figure 5.1 is flat. To see this, one could obtain the set represented by each node. Therefore, whether a QOBDD is flat, can depend on the ordering of the players in general. As we will see below, QOBDDs for WVGs are flat for every ordering of the players. This fact will be used in Section 7.3 for a heuristic to identify QOBDDs that do not represent WVGs.

Using Def. 5.1, we can now generalize the result in Hosaka et al. (1994) from QOBDDs for WVGs to flat QOBDDs.

Lemma 5.3. *Let r be the root of a QOBDD. For each level $i \in N$ and every two nodes $u, v \in \mathsf{V}_i$ with $u \subset v$ it holds $\mathsf{then}(u) \subseteq \mathsf{then}(v)$ and $\mathsf{else}(u) \subseteq \mathsf{else}(v)$ and at least one inequality is strict.*

Proof. This is rather easy to see. Assume that $S \in \mathsf{set}(\mathsf{then}(u))$. Then $i \notin S$ and $S + i \in \mathsf{set}(u)$. Hence, by $u \subset v$ we have $S + i \in \mathsf{set}(v)$ and it follows $(S + i) - i = S \in \mathsf{set}(\mathsf{then}(v))$. A similar argumentation can be used to see $\mathsf{else}(u) \subseteq \mathsf{else}(v)$. If both inequalities were not strict, then it would hold $\mathsf{set}(u) = \mathsf{set}(v)$ in contradiction to $u \subset v$. □

[1] In a chain, each two elements are comparable with respect to the order.

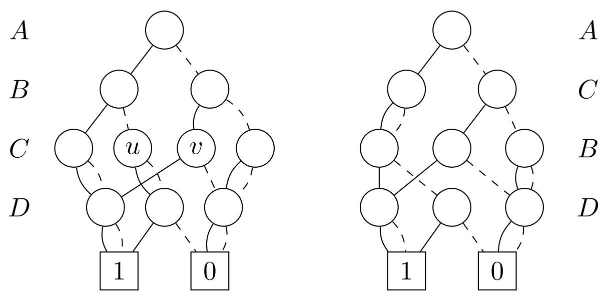


Figure 5.1.: A non-flat (left) and a flat QOBDD (right) for the same simple game.

The next result is a generalization of Lemma 3 in Hosaka et al. (1994).

Lemma 5.4. *If the* QOBDD *with root r is flat, then for each $i \in \{2, \dots, n\}$ it holds* $|\mathsf{V}_i| \le \min\{2|\mathsf{V}_{i-1}|, 2|\mathsf{V}_{i+1}| - 1\}$.

Proof. Let $i \in N$ with $i \ge 2$. Because each node on level $i-1$ has two outgoing edges, the number of nodes from level $i-1$ to level i can at most double. Hence, $|\mathsf{V}_i| \le 2|\mathsf{V}_{i-1}|$.

Let i be an element from N. For the second part, we define two functions $ord : \mathsf{V}_i \to \{0, \dots, |\mathsf{V}_i| - 1\}$ and $sumord : \mathsf{V}_i \to \{0, \dots, 2(|\mathsf{V}_{i+1}| - 1)\}$ and we show that the latter one is an injective mapping. The functions are defined as

$$ord(v) := |\{u \in \mathsf{V}_i \mid \mathsf{set}(u) \subset \mathsf{set}(v)\}|$$

and, respectively,

$$sumord(v) := ord(\mathsf{then}(v)) + ord(\mathsf{else}(v))\,.$$

To justify the injectivity, let $u, v \in \mathsf{V}_i$ such that w.l.o.g. $u \subset v$. Then we have $ord(u) < ord(v)$ and by Lemma 5.3 it holds

$$ord(\mathsf{then}(u)) \le ord(\mathsf{then}(v)) \text{ and } ord(\mathsf{else}(u)) \le ord(\mathsf{else}(v))$$

and at least one of these inequalities is strict. Thus we get:

$$ord(\mathsf{then}(u)) + ord(\mathsf{else}(u)) < ord(\mathsf{then}(v)) + ord(\mathsf{else}(v))$$

and this yields $sumord(u) < sumord(v)$. The injectivity of $sumord$ now implies

$$|\mathsf{V}_i| \le |\{0, \dots, 2(|\mathsf{V}_{i+1}|) - 1\}| = 2|\mathsf{V}_{i+1}| - 1\,. \qquad \square$$

The next result is a generalization of Lemma 4 in Hosaka et al. (1994) and establishes the upper bound for the size of a flat QOBDD. The proof is omitted, because the main work has already been done in Lemma 5.4. In Theorem 5.11 in Section 5.3 we will see a similar proof for the case of QOBDDs representing WVGs.

Theorem 5.5. *If the* QOBDD *with root r is flat, then* $\mathsf{size}(r) \in \mathcal{O}(2^{n/2})$. $\square$

This result rises two questions. First, now that we do not need a weighted voting game in the proof for the upper bound $\mathcal{O}(2^{n/2})$ anymore, it is an open question whether we can prove a better upper bound for the size of QOBDDs representing WVGs. We postpone this question to Section 5.3. Second, do flat QOBDDs have a comparable concept in the context of simple games?

For the transition from flat QOBDDs to simple games we make use of Lapidot's desirability relation on coalitions (Lapidot 1972). A similar desirability relation has been defined by Winder (1962), which is not considered here. See, for instance, Taylor and Zwicker (1999) for a discussion and a comparison.

Definition 5.2. Let $S, T \in 2^N$ be coalitions. We say T *is at least as desirable as* S, denoted by $S \preceq_L T$, if $S \preceq_L T$

$$\forall U \subseteq N \setminus (S \cup T) : (S \cup U \in \mathcal{W} \implies T \cup U \in \mathcal{W})\,. \qquad \square$$

In contrast to the desirability relation on individuals $\preceq_I$, the relation is not necessarily transitive (Taylor and Zwicker 1999) and therefore not even a preorder. Totality of $\preceq_L$ on 2^N is a necessary condition for being a weighted simple game, because for any pair of coalitions $S, T \in 2^N$, $w(S) \leq w(T)$ implies $S \preceq_L T$.

Proposition 5.6. *If $(N, \mathcal{W})$ is weighted, then $\preceq_L$ is total on 2^N.* $\square$

The converse does not hold as has been shown by Einy (1985). Taylor and Zwicker (1995) present a class of non-weighted simple games for which $\preceq_L$ is total on 2^N. However, these games are "nearly" weighted (Taylor and Zwicker 1996). For a given ordering π of players we can show the following fact:

Lemma 5.7. *Let r be a π-QOBDD representing the simple game $(N, \mathcal{W})$. If for each level $i \in N$ and each $S, T \subseteq \pi(\{1, \ldots, i-1\})$ it holds $S \preceq_L T$ or $T \preceq_L S$, then r is flat.*

Proof. Let $i \in N$ be a level and let $u, v \in \mathsf{V}_{\pi(i)}$ be different nodes. By I we denote the set $\{i, \ldots, n\}$. Then there are $S, T \subseteq \pi(N \setminus I)$ such that $r \overset{S}{\to} u$ and $r \overset{T}{\to} v$, respectively. W.l.o.g. we assume $S \preceq_L T$. It holds:

$$\begin{aligned}
& S \preceq_L T \\
\iff & \forall U \subseteq N \setminus (S \cup T) : (S \cup U \in \mathcal{W} \Rightarrow T \cup U \in \mathcal{W}) && (\text{Def. } \preceq_L) \\
\implies & \forall U \subseteq \pi(I) : (S \cup U \in \mathcal{W} \Rightarrow T \cup U \in \mathcal{W}) && (N \setminus (S \cup T) \supseteq \pi(I)) \\
\iff & \forall U \subseteq \pi(I) : (U \in \mathsf{set}(u) \Rightarrow U \in \mathsf{set}(v)) && (\text{Thm. } 3.3) \\
\iff & \mathsf{set}(u) \subseteq \mathsf{set}(v) && (\mathsf{set}(u), \mathsf{set}(v) \subseteq 2^{\pi(I)}) \\
\iff & u \subset v\,. && (u \neq v)
\end{aligned}$$

This completes the proof. $\square$

A QOBDD is therefore flat, if just some coalitions can be compared w.r.t. $\preceq_L$ which is a much weaker condition than that $\preceq_L$ is total on 2^N. We can therefore expect to find flat QOBDDs for simple games, that are far from being weighted. For instance, the QOBDD representing the Treaty of Nice is flat, if the players are ordered by $1 \succeq_I \cdots \succeq_I n$ even though this game has no weighted representation.

Theorem 5.8. *The relation $\preceq_L$ is complete on 2^N for $(N, \mathcal{W})$ if and only if for every ordering π of the players the π-QOBDD for $(N, \mathcal{W})$ is flat.*

Proof. The direction "$\Longrightarrow$" is a direct consequence of Lemma 5.7. For the direction "$\Longleftarrow$" let $S, T \subseteq N$, set $i := |S \cup T| + 1$ and let π be an ordering of the players such that the first $i-1$ players are $\pi(\{1, \ldots, i-1\}) = S \cup T$. Let r denote the root of the π-QOBDD for $(N, \mathcal{W})$. Intuitively, by the choice of the ordering all the players in S and T have made their choice at level i in the QOBDD. There are usually many choices for π. We set $u := \mathsf{node}(r, i, S)$ and $v := \mathsf{node}(r, i, T)$. The QOBDD with root r is flat and therefore it holds $u \subseteq v$ or $v \subseteq u$. W.l.o.g. we assume $u \subseteq v$. Let $U \subseteq N \setminus (S \cup T)$ and assume that $S \cup U \in \mathcal{W}$. Because $\mathsf{set}(r) = \mathcal{W}$ it follows $S \cup U \in \mathsf{set}(r)$. By the choice of π we get $U \subseteq \pi(\{i, \ldots, n\})$. Therefore, by Theorem 3.3 it holds $U \in \mathsf{set}(u)$. Because $u \subseteq v$, it also follows $U \in \mathsf{set}(v)$. Again by Theorem 3.3 it now holds $T \cup U \in \mathsf{set}(r)$. Consequently, $T \cup U \in \mathcal{W}$ and therefore, $S \preceq_L T$. □

It is an open question, if there is a complete simple game, whose QOBDD representation is not flat for every ordering of the players. Hosaka et al. (1994), Takenaga, Nouzoe, and Yajima (1997) and Hosaka, Takenaga, Kaneda, and Yajima (1997) have studied the variable ordering problem for QOBDDs representing WVGs and threshold functions, respectively. It seems reasonable to put at least some of their results into the context of flat QOBDDs which, however, is out of the scope of this thesis.

5.3. Weighted Voting Games

Most structural properties of QOBDDs representing WVGs have already been discussed in Section 4.1, because they were used in earlier sections. What remains is the size of a QOBDD representing a WVG. In the previous section we have seen that if Lapidot's desirability relation on coalitions is complete on 2^N, then the size of the QOBDD is bounded by $\mathcal{O}(2^{n/2})$ and this is the case for WVGs. In this section we will answer the question whether this bound can be improved for WVGs.

Throughout this section let $[Q; w_1, \ldots, w_n]$ be a weighted representation of the simple game $(N, \mathcal{W})$ with players $N = \{1, \ldots, n\}$ and quota $Q \geq 1$. The QOBDD with root r represents $(N, \mathcal{W})$.

We start by showing a rather rough, but well-known upper bound for the size of a QOBDD representing a WVG.

Proposition 5.9. *It holds $\mathsf{width}(r) \leq Q+1$ and $\mathsf{size}(r) \leq n(Q+1) \in \mathcal{O}(nQ)$ for $Q \geq 1$.*

Proof. Due to the commitment to integer weighted representations it holds $\Delta(v) \geq 1$ for each node. By applying Lemma 4.6 and 4.7 from Section 4.1 we obtain for each $i \in N$:

$$|\mathsf{V}_i| - 2 \leq \frac{\max\limits_{v \in \mathsf{V}_i} l(v) - \min\limits_{v \in \mathsf{V}_i} u(v)}{\min_{v \in \mathsf{V}_i} \Delta(v)} \leq \frac{(Q-1) - 0}{1} = Q - 1\,.$$

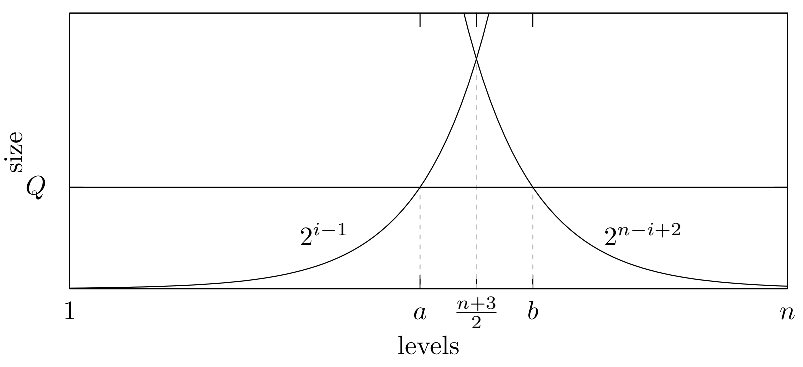


Figure 5.2.: Idea for the upper bound of the size of QOBDDs representing WVGs.

From this it follows $\mathsf{width}(r) \leq Q+1$. The size of *any* QOBDD is bounded by the product of its width and its height, where the height is n here. Therefore, we get for the size:

$$\mathsf{size}(r) \leq \sum_{i \in N} |\mathsf{V}_i| \leq \sum_{i \in N} \mathsf{width}(r) = n \cdot \mathsf{width}(r) \leq n(Q+1)\,. \qquad \square$$

By using Lemma 5.4 from the previous section, we can improve this upper bound. In comparison to the upper bound in Proposition 5.9, it states that the influence of the quota overrides that of the number of players, when Q grows. We need the following lemma.

Lemma 5.10. *It holds:*

$$\mathsf{size}(r) \leq \begin{cases} 2Q(n - 2\log_2 Q + 7) & \text{if } Q \leq 2^{(n+1)/2} \\ 12Q & \text{otherwise} \end{cases}.$$

Proof. Because r represents a WVG, regardless of the ordering of the players, with Lemma 5.4 (and induction) it follows

$$|\mathsf{V}_i| \leq \min\{2^{i-1}, 2^{n-i+2}\}\,.$$

Additionally, by Proposition 5.9 for every level $i \in N$ it holds $|\mathsf{V}_i| \leq Q+1$. Hence:

$$\mathsf{size}(r) = \sum_{i=1}^{n} |\mathsf{V}_i| \leq \sum_{i=1}^{n} \min\{2^{i-1}, Q+1, 2^{n-i+2}\}\,. \tag{5.1}$$

Set $k := \log_2 Q$ and set $a := k+1$ and $b := n-k+2$. The idea for the remaining proof is illustrated in Figure 5.2. We have $2^{x-1} = Q$ for $x = a$ and $2^{n-x+2} = Q$ for $x = b$. Both functions are equal for $x = (n+3)/2$, that is, $2^{x-1} = 2^{n-x+2}$. The levels are divided into three blocks, where the block in the middle is empty if $a \geq b$. This is the reason for two cases in the upper bound of $\mathsf{size}(r)$ which correspond to $a < b$ and $a \geq b$. The former inequality can be rewritten to:

$$a < b \iff k+1 < n-k+2 \iff \log_2 Q < (n+1)/2 \iff Q \leq 2^{(n+1)/2}\,.$$

For the following, keep in mind that $-\lfloor x \rfloor = \lceil -x \rceil$ for any $x \in \mathbb{R}$. Starting with (5.1) we have:

$$\sum_{i=1}^{n} \min\{2^{i-1}, 2^k + 1, 2^{n-i+2}\} \leq \sum_{i=1}^{\lceil a \rceil} 2^i + \max\{0, b-a\}(2^k+1) + \sum_{i=\lfloor b \rfloor}^{n} 2^{n-i+2} \,. \qquad (5.2)$$

For the two sums on the right-hand side of (5.2) it holds:

$$\sum_{i=1}^{\lceil a \rceil} 2^i + \sum_{i=\lfloor b \rfloor}^{n} 2^{n-i+2} \leq 2^{\lceil a \rceil + 1} + 2^{n - \lfloor b \rfloor + 3} \leq 2^{\lceil k \rceil + 2} + 2^{\lceil k \rceil + 1} \leq 3 \cdot 2^{k+2} = 12Q \,. \qquad (5.3)$$

Therefore, if $b \leq a$ then $\mathsf{size}(r) \leq 12Q$. Intuitively, this means, that Q is above the intersection of 2^{i-1} and 2^{n-i+2} in Figure 5.2. If, however, $b > a$, then it is

$$\max\{0, b-a\}(2^k+1) = (b-a)(2^k+1) = (n-2k+1)(2^k+1) \leq 2^{k+1}(n-2k+1)$$

which in turn can be used in conjunction with our previous result for the sums in (5.3) to conclude with:

$$\mathsf{size}(r) \leq 6 \cdot 2^{k+1} + 2^{k+1}(n-2k+1) = 2Q(n - 2\log_2 Q + 7) \,. \qquad \square$$

The proof of our main result is rather straightforward now.

Theorem 5.11. *It holds* $\mathsf{size}(r) \in \mathcal{O}(\max\{n - \log Q, 1\}Q)$.

Proof. We have to show that there is a constant $C > 0$ and $M \in \mathbb{N}$ such that for each WVG satisfying $n, Q \geq M$ it holds $\mathsf{size}(r) \leq C|\max\{Q, (n - \log_2 Q)Q\}|$. Because Q is assumed to be at least 1, the maximum is always positive. We set $C := 16$ and $M := 3$ and consider two cases. First, we assume $Q > 2^{(n+1)/2}$. Then from the previous lemma we know $\mathsf{size}(r) \leq 12Q$ and consequently, $\mathsf{size}(r) \leq CQ$. Second, we assume $Q \leq 2^{(n+1)/2}$. Let $k = \log_2 Q$. Because $n \geq 2$, $n-k$ is always positive and from $k \leq (n+1)/2$ we get $n - k \geq (n-1)/2 \geq 1$ for $n \geq 3$. By the result from the previous lemma we get:

$$\mathsf{size}(r) \leq 2Q(n-2k+7) < 2Q(n-k+7) \leq 2Q \cdot 8(n-k) \leq CQ(n-k) \,. \qquad \square$$

In the following sections, we will consider weighted voting games with additional restrictions. For these classes of WVGs, we will prove lower upper bounds for the size of their QOBDDs. We begin with so-called homogeneous simple games in the next section.

5.4. Homogeneous Simple Games

In this section we study the size of QOBDDs representing homogeneous simple games, a subclass of weighted voting games. We will show that the QOBDD for a directed and homogeneous n-person simple game has size $\mathcal{O}(n^2)$, but the size can be exponential in n, if the game is not directed.

Definition 5.3. A weighted representation $[Q; w_1, \ldots, w_n]$ is called *homogeneous*, if for each minimal winning coalition $S \in \mathcal{W}_{\min}$ it holds $w(S) = Q$. The simple game $(N, \mathcal{W})$ is called *homogeneous*, if it has a homogeneous weighted representation. □

As an example in the real world, consider the UN Security Council together with its homogeneous representation

$$[39; 7, 7, 7, 7, 7, \underbrace{1, \ldots, 1}_{\text{10-times}}] \,.$$

Each minimal winning coalition in this game has weight $5 \cdot 7$ for the five veto players plus weight $4 \cdot 1$ for four additional players, so that in the end, its weight is 39.

Homogeneous simple games have been mentioned as early as in the famous book *"Theory of Games and Economic Behavior"* by Morgenstern and von Neumann (1944). They have been studied extensively in the 1980's by a group of researchers in Bielefeld, Germany, which has published several papers on structural properties of homogeneous simple games, see Rosenmüller (1984), Ostmann (1987) and Rosenmüller (1987). A characterization of homogeneous simple games by so-called incidence vectors has been presented by Sudhölter (1989).

In the first part of this section we show that the QOBDD for a directed and homogeneous simple game has size $\mathcal{O}(n^2)$. We will make use of the following auxiliary statements.

Lemma 5.12. *If the weighted representation $g := [Q; w_1, \ldots, w_n]$ with $w_1 \geq \cdots \geq w_n$ is homogeneous, then both $h := [Q - w_1; w_2 \ldots, w_n]$ and $h' := [Q; w_2, \ldots, w_n]$ are homogeneous weighted representations.*

Proof. A coalition S is minimal winning in g if and only if

$$w(S) \geq Q > w(S) - \min_{k \in S} w_k \,.$$

Let S be minimal winning in h. We have to verify $w(S) = Q - w_1$. By the minimality of S in h it follows $w(S) \geq Q - w_1 > w(S) - \min_{k \in S} w_k$ and hence, $w(S) + w_1 \geq Q$ and $Q > w(S) + w_1 - \min_{k \in S} w_k$. Because $w_1 \geq \min_{k \in S} w_k$ we get

$$w(S \cup \{1\}) \geq Q > w(S \cup \{1\}) - \min_{k \in S \cup \{1\}} w_k \,.$$

Therefore, $S \cup \{1\}$ is minimal in g. By the homogeneity of g is follows $w(S \cup \{1\}) = Q$ and thus, $w(S) = Q - w_1$. The remaining case for h' can be shown analogously. □

The next statements brings QOBDDs into play.

Lemma 5.13. *If the* QOBDD *node v with label i represents the weighted voting game $[Q; w_i, \ldots, w_n]$, then* $\mathsf{then}(v)$ *represents $[Q - w_i; w_{i+1}, \ldots, w_n]$ and* $\mathsf{else}(v)$ *represents $[Q; w_{i+1}, \ldots, w_n]$.*

Proof. For a subset S of $\{i+1,\dots,n\}$ by the definition of $\mathsf{set}(v)$ it holds:

$$\begin{aligned} S \in \mathsf{set}(\mathsf{then}(v)) &\iff S \cup \{i\} \in \mathsf{set}(v) \\ &\iff w(S \cup \{i\}) \geq Q \\ &\iff w(S) + w_i \geq Q \\ &\iff w(S) \geq Q - w_i \,. \end{aligned}$$

The remaining case for $S \in \mathsf{set}(\mathsf{else}(v))$ can be shown similarly. □

The first important statement can be shown by induction on the structure of the QOBDD and the previous two lemmas. A proof is therefore omitted.

Lemma 5.14. *If the* QOBDD *with root r represents a simple game with the homogeneous weighted representation $[Q; w_1,\dots,w_n]$ and $w_1 \geq \dots \geq w_n$, then for each $i \in N$ and each $v \in \mathsf{V}_i(r)$ the node v represents a simple game having a homogeneous weighted representation with weights $w_i,\dots,w_n$.* □

The second important statement corresponds to Lemma 1.2 in Rosenmüller (1984). It states that given a set of n weights, there are at most n different quotas for which the weights and the quota constitute a homogeneous weighted representation. Its proof is straightforward and therefore omitted.

Lemma 5.15. *If $[Q; w_1,\dots,w_n]$ is a homogeneous weighted representation such that $w_1 \geq \dots \geq w_n$ and $w_n < Q \leq w_1 + \dots + w_n$, then there is $i_0 \in \{1,\dots,n\}$ with $Q = w(\{1,\dots,i_0\})$.*

The restrictions $w_n < Q$ and $Q \leq w_1 + \ldots + w_n$ guarantee that at least one coalition is losing and winning, respectively. The lemma does not cover the cases where all and, respectively, no coalitions are winning. In the former case, $[0; w_1,\dots,w_n]$ is a homogeneous weighted representation, whereas in the latter case $[w(N)+1; w_1,\dots,w_n]$ is such a representation. These cases correspond to the QOBDD nodes $\mathbb{I}_i$ and $\mathbb{O}_i$, if the remaining weights are $w_i,\dots,w_n$.

To get the idea of the following theorem, we take a look at any level i of the QOBDD for a directed and homogeneous simple game. From Lemma 5.14 we known that each inner node on level i represents a simple game having a homogeneous weighted representation with weights $w_i,\dots,w_n$. Furthermore, from Lemma 5.15 and the extreme cases $\mathbb{I}_i, \mathbb{O}_i$, we know that the number of homogeneous weighted representations with weights $w_1,\dots,w_n$ is at most $n-i+3$. This observation is used in the proof of the following result.

Theorem 5.16. *Let r be the* QOBDD *for a directed and homogeneous simple game. For each $i \in \{1,\dots,n\}$ it holds $\mathit{width}_i(r) \leq n-i+3$ and therefore it follows:*

$$\mathit{size}(r) \leq \frac{n^2+5n}{2} \in \mathcal{O}(n^2)\,.$$

Proof. Let $[Q; w_1, \dots, w_n]$ be the natural representation of $(N, \mathcal{W})$ and for $i \in N$ let $v \in \mathsf{V}_i$ be a node with label i. Because the game is directed we have $w_1 \geq \cdots \geq w_n$. By Lemma 5.14, the pair $(\{i, \dots, n\}, \mathsf{set}(v))$ is a directed and homogeneous simple game having a homogeneous weighted representation with weights $w_i, \dots, w_n$. If $v \notin \{\mathbb{O}_i, \mathbb{I}_i\}$, then by Lemma 5.15 there is a player $k_0 \in \{i, \dots, n\}$ such that v has the homogeneous weighted representation $[w(\{i, \dots, k_0\}); w_i, \dots, w_n]$. Hence, there are at most $n - i + 1$ possible quotas for the nodes in $\mathsf{V}_i \setminus \{\mathbb{O}_i, \mathbb{I}_i\}$ and therefore, it follows $|\mathsf{V}_i| \leq n - i + 3$. Based on that, for the size of the QOBDD it holds:

$$\mathsf{size}(r) = \sum_{i=1}^{n} |\mathsf{V}_i| \leq \sum_{i=1}^{n} (n - i + 3) = \frac{n^2 + 5n}{2}. \qquad \square$$

Hosaka, Takenaga, Kaneda, and Yajima (1997) have shown, that if the ordering of the players for a QOBDD representing a WVG is reversed, then the size of the result increases by at most $n - 1$ additional inner nodes. Therefore, there are at least two orderings for which the QOBDD for a homogeneous simple game has size $\mathcal{O}(n^2)$.

In the remainder of this section we show that there is a homogeneous simple game and an ordering of the players, such that the corresponding QOBDD has size $\Omega(2^{n/2})$ and thus, the restriction to directed simple games in the previous analysis is crucial.

For the proof, we use a slightly altered version of the weighted voting game EXP_n in Hosaka, Takenaga, and Yajima (1994) that we call $\text{EXP}_n + 1$. In order to end up with a homogeneous game, we increase the quota of EXP_n by 1. We do only consider the case of an even number of players to keep things easy.

Definition 5.4. For an even number of players $n \geq 2$, the WVG $\text{EXP}_n + 1$ has weights $w_k := w_{n-k+1} := 2^{k-1}$ for $k = 1, \dots, n/2$ and quota $Q := 1 + \sum_{i=1}^{n} w_i/2 = 2^{n/2}$. $\square$

For instance, $\text{EXP}_n + 1$ for $n = 8$ is $[16; 1, 2, 4, 8, 8, 4, 2, 1]$. In comparison, the original EXP_n for $n = 8$ has the same weights but quota 15. By rearranging the players it can easily be seen, that the weighted representation of $\text{EXP}_n + 1$ is homogeneous. Therefore, by reordering the players and by Theorem 5.16 we can conclude:

Proposition 5.17. *For $\pi : N \to N$ fulfilling $w_{\pi(i)} \geq w_{\pi(i+1)}$, $1 \leq i \leq n - 1$, the π-QOBDD for $\text{EXP}_n + 1$ has size $\mathcal{O}(n^2)$.* $\square$

However, if the players remain as they are, the QOBDD representation of $\text{EXP}_n + 1$ has an exponential size in n.

Proposition 5.18. *The π-QOBDD for $\text{EXP}_n + 1$ with $\pi(i) = i$ has size $\Omega(2^{n/2})$.*

Proof. This result has been shown by Hosaka, Takenaga, and Yajima (1994) for EXP_n. We therefore just sketch the proof. For a quota $x \in \{0, 1, \dots, 2^{n/2} - 1\}$ consider the WVG $g(x) = [x; w_{n/2+1}, \dots, w_n]$, that contains only the second half of the players. For each such x there is no x', such that $g(x)$ and $g(x')$ have the same winning coalitions, so that the number of games for the possible quotas is $2^{n/2}$. On the other hand, each such game $g(x)$ appears as an inner node in the QOBDD with root r representing $\text{EXP}_n + 1$, because there is a subset of the first half of the players $S \subseteq \{1, \dots, n/2\}$ with $w(S) = x$. Thus, $|\mathsf{V}_{n/2+1}(r)| = 2^{n/2}$. $\square$

We will use $\textsc{Exp}_n + 1$ in the next section on $\textsc{Wvg}$s with sequential weights again.

5.5. Sequential Weights

In this section we study weighted voting games with sequential weights. This class of simple games is not so interesting because it appears very often in practice, but because it has been posed as an open question in Chakravarty, Goel, and Sastry (2000) and in Aziz and Paterson (2008) if, for instance, the Banzhaf power index for a player (see Section 6.7) can be computed in time polynomial in the number of players n, or if this problem is NP-hard already. We will answer this question and show that this problem can be solved in polynomial time. We will also show that there is a $\textsc{Wvg}$ with sequential weights, whose $\textsc{Qobdd}$ representation has size $\Omega(2^{n/2})$ for at least one ordering of the players. Hence, as for homogeneous simple games, for $\textsc{Wvg}$s with sequential weights the ordering of the players matters.

$a|b$ In the following for $a, b \in \mathbb{N}$, we write $a|b$ if and only if a divides b.

Definition 5.5. A list of weights $w_1, \ldots, w_n$ is called *sequential* if $w_i|w_{i+1}$ for each $i \in \{1 \ldots, n-1\}$. In other words, w_2 is a multiple of w_1, w_3 is a multiple of w_2 and so on. □

For instance, the $\textsc{Wvg}$ $[20; 1, 2, 4, 8, 16]$ has sequential weights, because 1 divides 2, 2 divides 4 and so on. The players of the $\textsc{Wvg}$ $\textsc{Exp}_n + 1$ from the previous sections can be reordered such that the game has sequential weights. For a weighted representation with sequential weights the players are always ordered by non-decreasing weights. We first show our positive result.

Theorem 5.19. *Let $[Q; w_1, \ldots, w_n]$ be a $\textsc{Wvg}$ with sequential weights represented by the $\textsc{Qobdd}$ with root r. Then it holds $|\mathsf{V}_i(r)| \le i + 1$ and hence, $\mathsf{size}(r) \in \mathcal{O}(n^2)$.*

Proof. We show that the gap for almost all nodes on level $i \in N$ can be divided by the weight w_i:

$$\forall i \in N : \forall v \in \mathsf{V}_i \setminus \{\mathbb{I}_i, \mathbb{O}_i\} : w_i|\Delta(v)\,. \tag{5.4}$$

Let $i \in N$ and $v \in \mathsf{V}_i \setminus \{\mathbb{I}_i, \mathbb{O}_i\}$. The gap $\Delta(v)$ of node v is defined by $\Delta(v) = u(v) - l(v)$, and for $u(v)$ and $l(v)$ by definition it holds $u(v) = w(S)$, $l(v) = w(R)$ for some coalitions $R, S \subseteq \{i, \ldots, n\}$, respectively. Because we have a $\textsc{Wvg}$, it holds $l(v) < u(v)$. By $w_i|w_{i+1}, \ldots, w_{n-1}|w_n$ it follows $w_i|(w(S) - w(R))$ and hence, we get $w_i|\Delta(v)$. The nodes $\mathbb{I}_i, \mathbb{O}_i$ are excluded, because they have an infinite gap.

With Lemma 4.6 and because we have sequential weights it follows:

$$\max_{v \in \mathsf{V}_i} l(v) - \min_{v \in \mathsf{V}_i} u(v) \le (Q-1) - (Q - \sum_{k=1}^{i-1} w_k) \le \sum_{k=1}^{i-1} w_k \le (i-1)w_{i-1}\,. \tag{5.5}$$

By (5.4) we obtain $w_i \le \Delta(v)$ for each node $v \in \mathsf{V}_i$. If $\mathsf{V}_i \subseteq \{\mathbb{I}_i, \mathbb{O}_i\}$, then it trivially holds $|\mathsf{V}_i| \le i + 1$. Otherwise $\min_{v \in \mathsf{V}_i} \Delta(v) \in \mathbb{N}$, because at least one node has a finite gap. With Lemma 4.7, (5.5) and $w_{i-1} \leq w_i$ we get:

$$|\mathsf{V}_i| \leq \frac{\max\limits_{v \in \mathsf{V}_i} l(v) - \min\limits_{v \in \mathsf{V}_i} u(v)}{\min\limits_{v \in \mathsf{V}_i} \Delta(v)} + 2 \leq \frac{(i-1)w_{i-1}}{w_i} + 2 \leq i + 1\,.$$

The size of the QOBDD is a direct consequence now:

$$\mathsf{size}(r) = \sum_{i=1}^{n} |\mathsf{V}_i| \leq \sum_{i=1}^{n} (i+1) \in \mathcal{O}(n^2)\,. \qquad \square$$

As mentioned in the previous section, reversing the ordering of the players only slightly affects the size of the QOBDD for a weighted voting game. In concrete, at most one additional node on each level is needed. Therefore, we get the following result:

Corollary 5.20. *If $w_1, \dots, w_n$ are sequential weights, then for the* QOBDD *with root r that represents $[Q; w_n, \dots, w_1]$ it holds $|\mathsf{V}_i| \leq i + 2$ and hence, $\mathsf{size}(r) \in \mathcal{O}(n^2)$.* $\square$

The WVG $\text{EXP}_n + 1$ from Def. 5.4 has sequential weights, if the players are ordered such that $w_i \leq w_{i+1}$ for $1 \leq i < n$. In the previous section we have also seen, that its QOBDD has exponential size in n, if the weights are not reordered. Hence, we can state without further proof:

Corollary 5.21. *There is a* WVG *with sequential weights, namely* $\text{EXP}_n + 1$*, whose* QOBDD *has size $\Omega(2^{n/2})$ for at least one ordering of the players.* $\square$

This result is likely the reason why both Chakravarty et al. (2000) and Aziz and Paterson (2008) have not been able to prove, that the Banzhaf power index for WVGs with sequential weights can be computed in polynomial time in n. But this is the case, because, as we will see in Section 6.7, that problem has time complexity linear in the size of the QOBDD.

5.6. Conclusions

In this chapter we have studied the structure and size of QOBDDs for specific classes of simple games. Namely, we have discussed complete simple games, weighted voting games, homogeneous simple games and WVGs with sequential weights. We have also defined and analyzed flat QOBDDs, which correspond to the very intuitive concept, that the nodes on each level of a QOBDD are totally ordered w.r.t. set inclusion. The upper bound $\mathcal{O}(2^{n/2})$ for the size of QOBDDs representing WVGs presented in Hosaka, Takenaga, and Yajima (1994) does already hold for flat QOBDDs, but being flat is a much weaker condition. In the context of simple games, many non-weighted real world simple games have flat QOBDD representations. We have used Lapidot's desirability relation on coalitions to establish a connection between flat QOBDDs and simple games. The exact meaning of flat QOBDDs in the context of simple games remains open though. It is also unknown, how to find an ordering of the players, for which the QOBDD representation is flat in general. It would be interesting to see if this problem, given a vector-weighted representation, is NP-hard.

For QOBDDs representing weighted voting games we have been able to improve the upper bound for the QOBDD size to $\mathcal{O}(\max\{(n-\log Q),1\}Q)$, where Q is the quota and n is the number of players. This result is interesting, because by using the algorithm to build the QOBDD for a WVG in Section 4.2 we immediately obtain an algorithm with (deterministic) running time

$$\mathcal{O}(\min\{n2^{n/2}, \max\{(n-\log Q),1\}Q\log Q\})$$

to solve the 0-1 knapsack problem. See Behle (2008) for details on the connection between the different problems. In the context of simple games, as we will see in Section 6.7, this finding improves some complexity results for computing power indices for WVGs.

For homogeneous simple games and for weighted voting games with sequential weights, we have been able to establish an upper bound of $\mathcal{O}(n^2)$ for fixed orderings of the players and weights, respectively, even though for other orderings the size might be exponential in n. We have used the weighted representation of $\text{EXP}_n + 1$ to this end, which is homogeneous and has sequential weights.

Some classes of weighted voting games, like WVGs with k-unbalanced weights, have been omitted. The interested reader is referred to Chakravarty, Goel, and Sastry (2000) and Aziz and Paterson (2008) for an overview of candidates for future research.

Part III.

Solving Problems on Simple Games

6. Algorithms for Simple Games

In Chapter 4 we have seen how the QOBDD for a simple game can be built and in Chapter 5 we have considered the size of the QOBDDs for some classes of simple games, like weighted voting games. The aim of this chapter is to develop methods and algorithms to solve problems on simple games represented by QOBDDs.

Essential parts of this chapter have been published in Berghammer and Bolus (2010), Bolus (2011b) and Berghammer and Bolus (2012). The presentation is different here though. In Section 6.1 we start by introducing the idea of a manipulator, which can be used to manipulate a traversal of a QOBDD. As we will see, the use of manipulators requires the adaption of some ideas for QOBDDs. To the authors knowledge, the idea of manipulators is novel. The reason for this might be, that the idea cannot easily be applied to ROBDDs, but most practitioners use that kind of decision diagrams.

Section 6.2 presents a thorough discussion of counting algorithms for QOBDDs that have been used in Bolus (2011b) to compute power indices. Power indices are discussed later in Section 6.7. One of the main results of Section 6.2 is the following. Let $(N, \mathcal{W})$ be a simple game with players $N = \{1, \dots, n\}$ that is represented by the QOBDD with root r. For a player $i \in N$ we consider the cardinality $c_i := |\{S \in \mathcal{W} \mid i \in S\}|$. We will show, that *all* the values $c_1, \dots, c_n$ can be obtained with just $\mathcal{O}(\mathsf{size}(r))$ arithmetic operations.

The remaining sections cover fundamental problems like the computation of the desirability relation on individuals in Section 6.3, and the minimal winning (resp. maximal losing) coalitions in Section 6.5. We will make use of some of these results in Chapter 7. In Section 6.9 we draw conclusions and present running times of our algorithms for some real world simple games which frequently appear in the literature.

6.1. Manipulators

In this section we present a novel approach to improve the performance when using operations which cause only local changes to the QOBDD structure.

Developing a QOBDD package is a complex problem and involves many fundamental design decisions. One such design decision usually is, that once a QOBDD has been created, it is immutable and represents the same Boolean function (resp. subset of 2^N for labels N) during its lifetime. This, however, has a very negative effect on memory usage because even trivial changes in the represented set $\mathcal{A} \subseteq 2^N$ can require a completely new QOBDD and both QOBDDs do not share a single inner node. This is illustrated in Figure 6.1. By switching only the edges for the node v, the resulting QOBDD for the set $\mathcal{A}'$ cannot reuse any inner node. The memory usage therefore doubles. Memory, however, is limited and cannot easily be traded for computation time. So, saving memory can be considered more important than saving computation time.

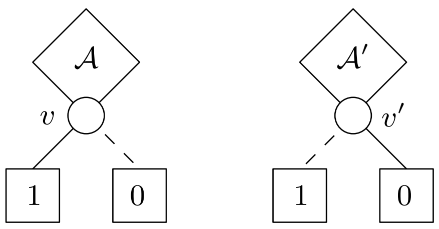


Figure 6.1.: Even though the QOBDDs for the sets $\mathcal{A}$ and $\mathcal{A}'$ structurally differ only in one node, they do not share a single inner node.

Our approach is easy, but powerful enough to be useful in practice. Its fundamental idea is to *manipulate* decisions in a QOBDD. For instance, a manipulation could mean that for every inner node with label $i \in N$ we take the 0-edge instead of the 1-edge and vice versa. Another example for a manipulator could be the redirection of the 1-edge to another inner node, say, $\mathbb{O}_{i+1}$.

Definition 6.1. A pair $\Phi = (\Phi_T, \Phi_E)$ of functions Φ_T, Φ_E that for each $i \in N$ maps QOBDD nodes of label i to QOBDD nodes of label $i+1$ is called a *manipulator*. □

Φ_T, Φ_E

id

Intuitively, a manipulator can be considered as an abstraction of the functions then and else for a node. Therefore, whenever Φ is a manipulator we refer to Φ_T ("then") as its first component and to Φ_E ("else") as its second component. The most trivial manipulator is the identity id that is defined by:

$$\mathsf{id} := (\mathsf{then}, \mathsf{else}) \ . \tag{6.1}$$

$\Phi \circ \Psi$

The composite $\Phi \circ \Psi$ of two manipulators Φ, Ψ, which is a manipulator again, is defined component-wise by:

$$\Phi \circ \Psi := (\Phi_T \circ \Psi_T, \Phi_E \circ \Psi_E) \, . \tag{6.2}$$

$\Phi(S)$

For a set of QOBDD nodes S, as a convenience, we define the successors w.r.t. Φ as $\Phi(S) := \Phi_T(S) \cup \Phi_E(S)$.

The set represented by a QOBDD changes when a manipulator is used:

$\mathsf{set}_\Phi(v)$

Definition 6.2. Let Φ be a manipulator and let v be a (maybe terminal) QOBDD node. We define the *set represented by v w.r.t. Φ*, denoted by $\mathsf{set}_\Phi(v)$, as:

$$\mathsf{set}_\Phi(v) = \begin{cases} \emptyset & \text{if } v = \mathbb{O} \\ \{\emptyset\} & \text{if } v = \mathbb{I} \\ \{S + \mathsf{var}(v) \mid S \in \mathsf{set}_\Phi(\Phi_T(v))\} \cup \mathsf{set}_\Phi(\Phi_E(v)) & \text{otherwise} \end{cases} \ .$$

To stay consistent with our previous definition of set we redefine set as $\mathsf{set}_{\mathsf{id}}$. □

Before we introduce additional manipulators, we have to reconsider some fundamentals for QOBDDs in the context of manipulators.

Lemma 6.1 (Shannon decomposition)**.** *Let Φ be a manipulator. For an inner* QOBDD *node v with label i and a set $S \subseteq \{i, \dots, n\}$ it holds $S \in \mathsf{set}_\Phi(v)$ if and only if*

$$\big(i \in S \wedge S - i \in \mathsf{set}_\Phi(\Phi_T(v))\big) \vee \big(i \notin S \wedge S \in \mathsf{set}_\Phi(\Phi_E(v))\big) . \tag{6.3}$$

Proof. The equivalence follows straight from the definition of set_Φ in Def. 6.2 and $\mathsf{set}_\Phi(v) \subseteq 2^{\{\mathsf{var}(v),\dots,n\}}$ for any QOBDD node v. $\square$

We will usually omit the parenthesis and instead use that $\wedge$ has precedence over $\vee$.

Because for the prerequisites of the previous lemma, $S \in \mathsf{set}_\Phi(\Phi_E(v))$ implies $\mathsf{var}(v) \notin S$, in some situations we will omit the $i \notin S$ part in (6.3) and instead use:

$$S \in \mathsf{set}_\Phi(v) \iff i \in S \wedge S - i \in \mathsf{set}_\Phi(\Phi_T(v)) \vee S \in \mathsf{set}_\Phi(\Phi_E(v)) . \tag{6.4}$$

One of our main intentions is to use the binary synthesis in conjunction with manipulators. Similar to the original algorithm apply from Section 3.4 we use Algorithm 4 for this purpose. In the postcondition of algorithm apply_2, we use $S \in \mathsf{set}_\Phi(u)$ and $S \in \mathsf{set}_\Psi(v)$ as predicates on a meta level.

Algorithm 4 $\mathsf{apply}_2(u, \Phi, v, \Psi, \otimes)$

Require: u, v are QOBDD nodes with label $i = \mathsf{var}(u)$, Φ, Ψ are manipulators and $\otimes : \mathbb{B}^2 \to \mathbb{B}$ is a function.

Ensure: $\forall S \in 2^{\{i,\dots,n\}} : (S \in \mathsf{set}(\mathsf{apply}_2(u, \Phi, v, \Psi, \otimes)) \iff S \in \mathsf{set}_\Phi(u) \otimes S \in \mathsf{set}_\Psi(v))$.

```
1: if $(u = \mathbb{I}) \otimes (v = \mathbb{I})$ then return $\mathbb{I}$
2: else if $u \in \{\mathbb{I}, \mathbb{O}\}$ then return $\mathbb{O}$
3: else if $e \neq \bot$ then return $e$ where $e = lookup(T, (\otimes, (u, \Phi), (v, \Psi)))$
4: else $w \leftarrow \mathsf{ite}(i, \mathsf{apply}_2(\Phi_T(u), \Phi, \Psi_T(v), \Psi, \otimes), \mathsf{apply}_2(\Phi_E(u), \Phi, \Psi_E(v), \Psi, \otimes))$
       $insert(T, (\otimes, (u, \Phi), (v, \Psi)), w))$
       return $w$
```

The correctness can similarly be shown to that of the original algorithm apply. The proof of the following theorem is therefore omitted.

Theorem 6.2. *If u and v are* QOBDD *nodes with label i, Φ and Ψ are manipulators and $\otimes : \mathbb{B}^2 \to \mathbb{B}$ is a function, then for each $S \in 2^{\{i,\dots,n\}}$ it holds:*

$$S \in \mathsf{set}(\mathit{apply}_2(u, \Phi, v, \Psi, \otimes)) \iff S \in \mathsf{set}_\Phi(u) \otimes S \in \mathsf{set}_\Psi(v) . \qquad \square$$

Notice that it is crucial for the computed table to contain the nodes as well as the manipulators in the keys. This is, because the same node can have a different meaning for different manipulators.

The running time of $\mathsf{apply}_2(u, \Phi, v, \Psi, \otimes)$ is more subtle. The idea for the proof of the running time $\mathcal{O}(\sum_{i=\mathsf{var}(u)}^{n} \mathsf{width}_i(u) \cdot \mathsf{width}_i(v))$ of the algorithm apply without manipulators is, that the size of the computed table is bounded by

$$|\bigcup_{i=\mathsf{var}(u)}^{n} \mathsf{V}_i(u) \times \mathsf{V}_i(v)| .$$

But when we use manipulators, this does not hold anymore, because the manipulators may map to inner nodes not in $\mathsf{V}(u)$ or $\mathsf{V}(v)$. The nodes that are reachable from the node u with label 1, without a manipulator, are $\{u\} = \mathsf{V}_1(u)$, $\mathsf{V}_2(u)$, $\mathsf{V}_3(u)$ and so on. When a manipulator Φ is used, the sets of reachable inner nodes are

$$\{u\},\ \Phi(\{u\}),\ \Phi^2(\{u\}),\ \Phi^3(\{u\}), \dots, \Phi^{n-1}(\{u\})$$

$\mathsf{V}_i(u, \Phi)$

$\mathsf{width}_i(u, \Phi)$

$\mathsf{width}(u, \Phi)$

$\mathsf{size}(u, \Phi)$

where Φ^k applies Φ exactly k-times. These are the nodes on the levels of the "manipulated" QOBDD. Therefore, similar to $\mathsf{V}_i(u)$, for a manipulator Φ we define $\mathsf{V}_i(u, \Phi)$ as $\Phi^{i-1}(\{u\})$ and $\mathsf{V}(u, \Phi)$ as the union of $\mathsf{V}_i(u, \Phi)$ for all $i = 1, \dots, n$. Analogous to the definition for QOBDDs without manipulators, we define $\mathsf{width}_i(u, \Phi) := |\mathsf{V}_i(u, \Phi)|$ and $\mathsf{width}(u, \Phi) := \max_{i \in N} \mathsf{width}_i(u, \Phi)$. The *size of u w.r.t. Φ* is $\mathsf{size}(u, \Phi) := |\mathsf{V}(u, \Phi)|$.

Theorem 6.3. *Disregarding the time necessary to evaluate the manipulators, the algorithm $\mathsf{apply}_2(u, \Phi, v, \Psi, \otimes)$ has expected running time*

$$\mathcal{O}\Big(\sum_{k=\mathsf{var}(u)}^{n} \mathsf{width}_i(u, \Phi) \cdot \mathsf{width}_i(v, \Psi) \Big)$$

which in turn, is bounded from above by $\mathcal{O}((n - \mathsf{var}(u) + 1) \cdot \mathsf{width}(u, \Phi) \cdot \mathsf{width}(v, \Psi))$.

Proof. The size $|T|$ of the computed table is an upper bound for the number of recursive calls. Each recursive call costs expected time $\mathcal{O}(1)$ due to the use of a hash table for T. The size of T is bounded by

$$\sum_{k=\mathsf{var}(u)}^{n} \mathsf{width}_i(u, \Phi) \cdot \mathsf{width}_i(v, \Psi),$$

because in the worst case on each level $i \in \{\mathsf{var}(u), \dots, n\}$ each element in

$$\{\otimes\} \times (\mathsf{V}_i(u, \Phi) \times \{\Phi\}) \times (\mathsf{V}_i(v, \Psi) \times \{\Psi\})$$

appears in an entry in T. □

We define shorthands for some special cases for the function $\otimes$:

$$\begin{aligned}
\mathsf{and}(u, \Phi, v, \Psi) &:= \mathsf{apply}_2(u, \Phi, v, \Psi, \wedge) \\
\mathsf{or}(u, \Phi, v, \Psi) &:= \mathsf{apply}_2(u, \Phi, v, \Psi, \vee) \\
\mathsf{minus}(u, \Phi, v, \Psi) &:= \mathsf{apply}_2(u, \Phi, v, \Psi, \not\Rightarrow) .
\end{aligned}$$

By Theorem 6.2, these definitions have straightforward set-theoretic interpretations:

$$\begin{aligned}
\mathsf{set}(\mathsf{and}(u, \Phi, v, \Psi)) &= \mathsf{set}_\Phi(u) \cap \mathsf{set}_\Psi(v) \\
\mathsf{set}(\mathsf{or}(u, \Phi, v, \Psi)) &= \mathsf{set}_\Phi(u) \cup \mathsf{set}_\Psi(v) \\
\mathsf{set}(\mathsf{minus}(u, \Phi, v, \Psi)) &= \mathsf{set}_\Phi(u) \setminus \mathsf{set}_\Psi(v) .
\end{aligned}$$

The algorithm apply_2 can also be used to compare the sets represented by two QOBDD nodes u, v with label i. It holds:

$$\mathsf{set}_\Phi(u) \subseteq \mathsf{set}_\Psi(v) \iff \mathsf{minus}(u, \Phi, v, \Psi) = \mathbb{O}_i\,.$$

Because it is sufficient to know if the result is the node $\mathbb{O}_i$ it would be superfluous to create any temporary nodes (Brace et al. 1990). Therefore, we use a variant of apply_2 called forall, that coincides with apply_2 with the exception that true or false is returned instead of a QOBDD node. The recursive calls in Line 4 in Algorithm 4

$$w \leftarrow \mathsf{ite}(i, \mathsf{apply}_2(\Phi_T(u), \Phi, \Psi_T(v), \Psi, \otimes), \mathsf{apply}_2(\Phi_E(u), \Phi, \Psi_E(v), \Psi, \otimes))$$

are replaced by

$$w \leftarrow \mathsf{forall}(\Phi_T(u), \Phi, \Psi_T(v), \Psi, \otimes) \wedge \mathsf{forall}(\Phi_E(u), \Phi, \Psi_E(v), \Psi, \otimes))\,. \tag{6.5}$$

A proof for the correctness is omitted again. We only state the corresponding result:

Theorem 6.4. *For* QOBDD *nodes u, v with label i, manipulators Φ, Ψ and a function $\otimes : \mathbb{B}^2 \to \mathbb{B}$ it holds $\mathsf{forall}(u, \Phi, v, \Psi, \otimes) = \mathsf{true}$ if and only if*

$$\forall S \in 2^{\{i,\dots,n\}} : S \in \mathsf{set}_\Phi(u) \otimes S \in \mathsf{set}_\Psi(v)\,.$$

The running time is the same as for apply_2 in Theorem 6.3. □

Further variations are possible, e.g., existential quantification by using disjunction instead of conjunction in (6.5). We are especially interested in the subset and the equality relation and therefore, we define:

$$\mathsf{subseteq}(u, \Phi, v, \Psi) := \mathsf{forall}(u, \Phi, v, \Psi, \Rightarrow) \tag{6.6}$$

$$\mathsf{equal}(u, \Phi, v, \Psi) := \mathsf{forall}(u, \Phi, v, \Psi, \Leftrightarrow) \tag{6.7}$$

We then have $\mathsf{subseteq}(u, \Phi, v, \Psi) = \mathsf{true}$ if and only if $\mathsf{set}_\Phi(u) \subseteq \mathsf{set}_\Psi(v)$ and we have $\mathsf{equal}(u, \Phi, v, \Psi) = \mathsf{true}$ if and only if $\mathsf{set}_\Phi(u) = \mathsf{set}_\Psi(v)$.

It is well-known that the set-equality test $\mathsf{set}(u) = \mathsf{set}(v)$ for QOBDDs without manipulators can be implemented in deterministic running time $\mathcal{O}(\min\{\mathsf{size}(u), \mathsf{size}(v)\})$[1]. We use a weaker result which does hold for our approach using forall.

Corollary 6.5. *Disregarding the time necessary to evaluate the manipulators, the algorithm $\mathsf{equal}(u, \Phi, v, \Psi, \otimes)$ has expected running time $\mathcal{O}(\min\{\mathsf{size}(u, \Phi), \mathsf{size}(v, \Psi)\})$.* □

We now present the manipulators that we will use in the remainder of the thesis. With id we have already seen a manipulator at the beginning of this section. This manipulator is special, because it is the only manipulator that directly refers to the 1-edge $\mathsf{then}(v)$ and the 0-edge $\mathsf{else}(v)$ of a node v. All other manipulators will be defined relative to another manipulator Φ. For instance, the manipulator compls that interchanges the compls

[1]This fact uses that QOBDDs are canonical for a fixed ordering of the variables and in case the sets coincide, the structure of the QOBDDs is isomorphic. This special case can therefore be implemented without a computed table.

edges of each node is defined as:

$$\mathsf{compls} \circ \Phi := (\Phi_E, \Phi_T) \ . \tag{6.8}$$

As a consequence, the composition compls ∘ compls ∘ id would rule out the effect of each compls. As a simplification later, we usually omit the composition with id, so that, for example, compls actually refers to compls ∘ id.

Lemma 6.6. *Let Φ be a manipulator and let v be a* QOBDD *node with label i. For $S \subseteq \{i, \dots, n\}$ it holds:*

$$S \in \mathsf{set}_{\mathit{compls}\circ\Phi}(v) \iff \{i, \dots, n\} \setminus S \in \mathsf{set}_{\Phi}(v) \, .$$

Proof. By Ψ we denote compls. For $k \leq n+1$ we denote the set $\{k, \dots, n\}$ by D_k. The proof is by induction on the ordering of the labels, i.e., we start with label $n+1$ and the sinks. The induction base is trivially correct. For the induction step let v be a QOBDD node with label i and assume the statement is true for nodes with label $i+1$. For $S \subseteq D_i$ we have:

$$\begin{aligned}
& S \in \mathsf{set}_{\Psi\circ\Phi}(v) \\
\iff & i \in S \wedge S - i \in \mathsf{set}_{\Psi\circ\Phi}((\Psi \circ \Phi)_T(v)) \vee \\
& i \notin S \wedge S \in \mathsf{set}_{\Psi\circ\Phi}((\Psi \circ \Phi)_E(v)) && \text{(Eq. 6.3)} \\
\iff & i \in S \wedge S - i \in \mathsf{set}_{\Psi\circ\Phi}(\Phi_E(v)) \vee \\
& i \notin S \wedge S \in \mathsf{set}_{\Psi\circ\Phi}(\Phi_T(v)) && \text{(Def. compls)} \\
\iff & i \in S \wedge D_{i+1} \setminus (S - i) \in \mathsf{set}_{\Phi}(\Phi_E(v)) \vee \\
& i \notin S \wedge D_{i+1} \setminus S \in \mathsf{set}_{\Phi}(\Phi_T(v)) && \text{(Ind. Hyp.)} \\
\iff & i \notin D_i \setminus S \wedge D_i \setminus S \in \mathsf{set}_{\Phi}(\Phi_E(v)) \vee i \in D_i \setminus S \wedge (D_i \setminus S) - i \in \mathsf{set}_{\Phi}(\Phi_T(v)) && (*) \\
\iff & D_i \setminus S \in \mathsf{set}_{\Phi}(v) \, . && \text{(Eq. 6.3)}
\end{aligned}$$

The equivalence of the left-hand sides of the disjunctions in the step marked with (*) can be seen as follows. It is $i \in S$ if and only if $i \notin D_i \setminus S$. By having $i \in S$ we also have

$$D_{i+1} \setminus (S - i) = (D_{i+1} + i) \setminus ((S - i) + i) \overset{i \in S}{=} (D_{i+1} + i) \setminus S = D_i \setminus S \, .$$

The missing equivalences of the right-hand sides of the disjunctions in (*) can be seen analogously. □

without$_i$ Let $i \in N$ be a label. The next manipulator without$_i$ makes a decision based on the label of the node passed to it and it may map to a node that is not in the QOBDD, namely, $\mathbb{O}_{i+1}$. Similar to compls, without$_i$ is defined relative to a manipulator Φ:

$$\mathsf{without}_i \circ \Phi := \left(v \mapsto \begin{cases} \mathbb{O}_{i+1} & \text{if } \mathsf{var}(v) = i \\ \Phi_T(v) & \text{otherwise} \end{cases}, \Phi_E \right) \, . \tag{6.9}$$

For any node v with $\mathsf{var}(v) \neq i$, the manipulator behaves like Φ. Otherwise, as illustrated in Figure 6.2, if $\mathsf{var}(v) = i$ then instead of $\Phi_T(v)$ the node $\mathbb{O}_{i+1}$ is returned. In the context of simple games this means, that every coalition containing player i becomes losing.

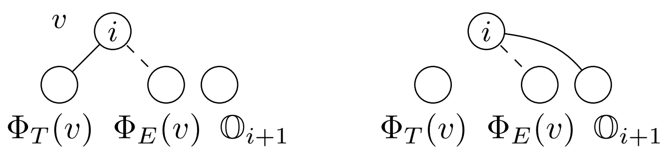


Figure 6.2.: Effect of Φ (left) and $\mathsf{without}_i \circ \Phi$ (right) for an inner node v with label i.

Lemma 6.7. *Let Φ be a manipulator, let $k \in N$ be a label and let v be a* QOBDD *node with label i. For $S \subseteq \{i, \dots, n\}$ it holds:*

$$S \in \mathsf{set}_{\mathsf{without}_k \circ \Phi}(v) \iff k \notin S \land S \in \mathsf{set}_\Phi(v)\,.$$

Proof. By Ψ we denote $\mathsf{without}_k$. The proof is again by induction on the ordering of the labels. Let v be a QOBDD node with label i. We have to distinguish the cases $i > k$, $i = k$ and $i < k$. First, if $i > k$ then it is rather easy to see that $S \in \mathsf{set}_{\Psi\circ\Phi}(v)$ if and only if $S \in \mathsf{set}_\Phi(v)$. This case includes the induction base $v \in \{\mathbb{O}, \mathbb{I}\}$. Second, if $k = i$ then for $S \subseteq \{i, \dots, n\}$ we have:

$$\begin{aligned}
& S \in \mathsf{set}_{\Psi\circ\Phi}(v) \\
\iff & i \in S \land S - i \in \mathsf{set}_{\Psi\circ\Phi}((\Psi \circ \Phi)_T(v)) \lor \\
& S \in \mathsf{set}_{\Psi\circ\Phi}((\Psi \circ \Phi)_E(v)) && \text{(Eq. 6.4)} \\
\iff & i \in S \land S - i \in \mathsf{set}_{\Psi\circ\Phi}(\mathbb{O}_{k+1}) \lor S \in \mathsf{set}_{\Psi\circ\Phi}(\Phi_E(v)) && \text{(Def. } \mathsf{without}_k,\ i = k) \\
\iff & \mathsf{false} \lor S \in \mathsf{set}_{\Psi\circ\Phi}(\Phi_E(v)) && \text{(Def. } \mathbb{O}_{k+1}) \\
\iff & S \in \mathsf{set}_\Phi(\Phi_E(v)) && \text{(Case } \mathsf{var}(\Phi_E(v)) > k) \\
\iff & k \notin S \land S \in \mathsf{set}_\Phi(v)\,. && \text{(Eq. 6.4)}
\end{aligned}$$

Finally, for $i < k$ we assume the statement holds for nodes with label $i + 1$. We have:

$$\begin{aligned}
& S \in \mathsf{set}_{\Psi\circ\Phi}(v) \\
\iff & i \in S \land S - i \in \mathsf{set}_{\Psi\circ\Phi}((\Psi \circ \Phi)_T(v)) \lor \\
& S \in \mathsf{set}_{\Psi\circ\Phi}((\Psi \circ \Phi)_E(v)) && \text{(Eq. 6.4)} \\
\iff & i \in S \land S - i \in \mathsf{set}_{\Psi\circ\Phi}(\Phi_T(v)) \lor \\
& S \in \mathsf{set}_{\Psi\circ\Phi}(\Phi_E(v)) && \text{(Def. } \mathsf{without}_k,\ i < k) \\
\iff & i \in S \land (k \notin S - i \land S - i \in \mathsf{set}_\Phi(\Phi_T(v))) \lor \\
& (k \notin S \land S \in \mathsf{set}_\Phi(\Phi_E(v))) && \text{(Ind. Hyp.)} \\
\iff & k \notin S \land \big(i \in S \land S - i \in \mathsf{set}_\Phi(\Phi_T(v)) \lor S \in \mathsf{set}_\Phi(\Phi_E(v))\big) && \text{(Rearrange)} \\
\iff & k \notin S \land S \in \mathsf{set}_\Phi(v)\,. && \text{(Eq. 6.4)}
\end{aligned}$$

This completes the proof. □

Let $i \in N$ be a label. For a QOBDD node v with label $\mathsf{var}(v) \leq i$, the last two manipulators remove_i and add_i are used two remove label i from (resp. add label i to) remove_i add_i

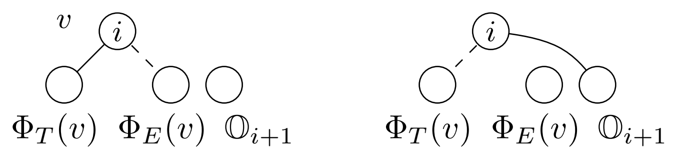


Figure 6.3.: Effect of Φ (left) and $\mathsf{remove}_i \circ \Phi$ (right) for an inner node v with label i.

all sets in $\mathsf{set}(v)$ that contain i (resp. do not contain i). For a manipulator Φ they are defined relative to Φ by:

$$\mathsf{remove}_i \circ \Phi := \left(v \mapsto \begin{cases} \mathbb{O}_{i+1} & \text{if } \mathsf{var}(v) = i \\ \Phi_T(v) & \text{otherwise} \end{cases}, v \mapsto \begin{cases} \Phi_T(v) & \text{if } \mathsf{var}(v) = i \\ \Phi_E(v) & \text{otherwise} \end{cases} \right) \tag{6.10}$$

$$\mathsf{add}_i \circ \Phi := \left(v \mapsto \begin{cases} \Phi_E(v) & \text{if } \mathsf{var}(v) = i \\ \Phi_T(v) & \text{otherwise} \end{cases}, v \mapsto \begin{cases} \mathbb{O}_{i+1} & \text{if } \mathsf{var}(v) = i \\ \Phi_E(v) & \text{otherwise} \end{cases} \right) \tag{6.11}$$

Because both manipulators are very similar, we do only discuss remove_i. For an inner node v, the manipulator behaves like Φ if $\mathsf{var}(v) \neq i$. For $\mathsf{var}(v) = i$, its behavior is illustrated in Figure 6.3. In this case, the 1-edge is "redirected" to the node $\mathbb{O}_{i+1}$ and the 0-edge is "redirected" to $\Phi_T(v)$.

Lemma 6.8. *Let Φ be a manipulator, let $k \in N$ be a label and let v be a* QOBDD *node with label i. For $S \subseteq \{i, \dots, n\}$ it holds*

$$\begin{aligned} S \in \mathsf{set}_{\mathsf{remove}_k \circ \Phi}(v) &\iff k \notin S \wedge S + k \in \mathsf{set}_\Phi(v) \\ S \in \mathsf{set}_{\mathsf{add}_k \circ \Phi}(v) &\iff k \in S \wedge S - k \in \mathsf{set}_\Phi(v) \end{aligned}$$

if $i \leq k$ and otherwise,

$$S \in \mathsf{set}_{\mathsf{remove}_k \circ \Phi}(v) \iff S \in \mathsf{set}_{\mathsf{add}_k \circ \Phi}(v) \iff S \in \mathsf{set}_\Phi(v)\,.$$

Proof. By Ψ we denote remove_k. The proof is again by induction on the ordering of the labels. Let v be a QOBDD node with label i. We have to distinguish the cases $i > k$, $i = k$ and $i < k$. First, if $i > k$ then it is rather easy to see that $S \in \mathsf{set}_{\Psi \circ \Phi}(v)$ if and only if $S \in \mathsf{set}_\Phi(v)$. This case includes the induction base $v \in \{\mathbb{O}, \mathbb{I}\}$. Second, if $k = i$ then we have for $S \subseteq \{i, \dots, n\}$:

$$\begin{aligned} & S \in \mathsf{set}_{\Psi\circ\Phi}(v) \\ \iff & i \in S \wedge S - i \in \mathsf{set}_{\Psi\circ\Phi}((\Psi \circ \Phi)_T(v)) \vee \\ & S \in \mathsf{set}_{\Psi\circ\Phi}((\Psi \circ \Phi)_E(v)) && \text{(Eq. 6.4)} \\ \iff & i \in S \wedge S - i \in \mathsf{set}_{\Psi\circ\Phi}(\mathbb{O}_{k+1}) \vee \\ & S \in \mathsf{set}_{\Psi\circ\Phi}(\Phi_T(v)) && (\text{Def. } \mathsf{remove}_k,\ i = k) \\ \iff & \mathsf{false} \vee S \in \mathsf{set}_{\Psi\circ\Phi}(\Phi_T(v)) && (\text{Def. } \mathbb{O}_{k+1}) \\ \iff & S \in \mathsf{set}_\Phi(\Phi_T(v)) && (\text{Case } \mathsf{var}(\Phi_T(v)) > k) \\ \iff & k \notin S \wedge T + k \in \mathsf{set}_\Phi(v)\,. \end{aligned}$$

Finally, for $i < k$ we assume the statement holds for nodes with label $i+1$. We have:

$$\begin{aligned}
& S \in \mathsf{set}_{\Psi}(v) \\
\iff & i \in S \wedge S - i \in \mathsf{set}_{\Psi\circ\Phi}((\Psi \circ \Phi)_T(v)) \vee \\
& S \in \mathsf{set}_{\Psi\circ\Phi}((\Psi \circ \Phi)_E(v)) && \text{(Eq. 6.4)} \\
\iff & i \in S \wedge S - i \in \mathsf{set}_{\Psi\circ\Phi}(\Phi_T(v)) \vee \\
& S \in \mathsf{set}_{\Psi\circ\Phi}(\Phi_E(v)) && \text{(Def. } \mathsf{remove}_k,\ i < k) \\
\iff & i \in S \wedge (k \notin S \wedge (S - i) + k \in \mathsf{set}_{\Phi}(\Phi_T(v))) \vee \\
& (k \notin S \wedge S + k \in \mathsf{set}_{\Phi}(\Phi_E(v))) && \text{(Ind. Hyp.)} \\
\iff & k \notin S \wedge \big(i \in S \wedge (S + k) - i \in \mathsf{set}_{\Phi}(\Phi_T(v)) \vee S + k \in \mathsf{set}_{\Phi}(\Phi_E(v))\big) && \text{(Rearrange)} \\
\iff & k \notin S \wedge S + k \in \mathsf{set}_{\Phi}(v)\,. && \text{(Eq. 6.4)}
\end{aligned}$$

The statement for add_k can be shown analogously. □

For the running time later, we have to discuss the potential blow-up caused by the manipulators. Fortunately, it is very moderate.

Proposition 6.9. *For a* QOBDD *node v with label $i \leq n$ and for $k \in N$ it holds*

$$\mathit{width}_i(v, \mathit{compls} \circ \Phi) = \mathit{width}_i(v, \Phi)$$

$$\left.\begin{array}{r} \mathit{width}_i(v, \mathit{without}_k \circ \Phi) \\ \mathit{width}_i(v, \mathit{remove}_k \circ \Phi) \\ \mathit{width}_i(v, \mathit{add}_k \circ \Phi) \end{array}\right\} \leq \begin{cases} \mathit{width}_i(v, \Phi) & \text{if } i \leq k \\ |\mathsf{V}_i(v, \Phi) \setminus \{\mathbb{O}_i\}| + 1 & \text{otherwise} \end{cases}$$

and therefore it follows

$$\mathit{size}(v, \mathit{compls} \circ \Phi) = \mathit{size}(v, \Phi)$$

$$\left.\begin{array}{r} \mathit{size}(v, \mathit{without}_k \circ \Phi) \\ \mathit{size}(v, \mathit{remove}_k \circ \Phi) \\ \mathit{size}(v, \mathit{add}_k \circ \Phi) \end{array}\right\} \leq \mathit{size}(v, \Phi) + n - k\,.$$

Proof. We first discuss the blow-up on level i. The manipulator compls does not cause any new inner node. The same holds for the remaining manipulators if $\mathsf{var}(v) \leq k$, because they imitate Φ's behavior. If $i > k$, then the node $\mathbb{O}_i$ has to be taken into account. The blow up is therefore at most 1.

The equality for $\mathsf{size}(v, \mathsf{compls} \circ \Phi)$ follows from:

$$\mathsf{size}(v, \mathsf{compls} \circ \Phi) = \sum_{j=i}^{n} \mathsf{width}_j(v, \mathsf{compls} \circ \Phi) = \sum_{j=i}^{n} \mathsf{width}_j(v, \Phi) = \mathsf{size}(v, \Phi)\,.$$

Let Ψ denote any of the manipulators $\mathsf{without}_k$, remove_k or add_k. If $i > k$, then some of the sums below are 0. The upper bound for $\mathsf{size}(v, \Psi \circ \Phi)$ follows from:

$$\begin{aligned}\mathsf{size}(v, \Psi \circ \Phi) &= \sum_{j=i}^{k} \mathsf{width}_j(v, \Psi \circ \Phi) + \sum_{j=k+1}^{n} \mathsf{width}_j(v, \Psi \circ \Phi) \\ &\leq \sum_{j=i}^{k} \mathsf{width}_j(v, \Phi) + \sum_{j=k+1}^{n} (\mathsf{width}_j(v, \Phi) + 1) \\ &= n - k + \sum_{j=i}^{n} \mathsf{width}_j(v, \Phi) \\ &= \mathsf{size}(v, \Phi) + n - k\,. \qquad \square\end{aligned}$$

Experiments and the Effect of the compls Manipulator

As for the binary synthesis of QOBDDs without manipulators, the formal upper bound in the case with manipulators does not coincide with what we experience in practice. In this section, we therefore present some experiments that we have obtained by our implementation and that shed light on what we can expect in practice, when we use manipulators. To keep things simple, we do only consider the case of the manipulator compls and the very specific case

$$\mathsf{forall}(r, \mathsf{compls}, r, \mathsf{id}, \otimes) \tag{6.12}$$

for a QOBDD with root r and any $\otimes$ that can be computed in constant time. This case is used later in Section 6.4 to decide if a simple game is proper and strong, respectively. The main result of this section is, that if r represents a WVG, then this case has expected running time $\mathcal{O}(\mathsf{size}(r))$ instead of $\mathcal{O}(n \cdot \mathsf{width}(r)^2)$. With respect to the nearly trivial effect of compls, this does not look very surprising. However, even if we relax the requirement to r slightly and "only" assume a flat QOBDD, the arguments used in the proof do not hold anymore.

Because we are only interested in the effect of the manipulator compls, we use forall with a function $\otimes$ that always maps to true.

$$\mathsf{reachable}(u, \Phi, v, \Psi) := \mathsf{forall}(u, \Phi, v, \Psi, (u, v) \mapsto \mathsf{true}) \tag{6.13}$$

This has two benefits. First, no matter which binary Boolean function $\otimes$ we use later, the size of the computed table of $\mathsf{reachable}(u, \Phi, v, \Psi)$ is an upper bound for the size of the computed table used in $\mathsf{forall}(r, \mathsf{compls}, r, \mathsf{id}, \otimes)$ and second, the computed tables for $\mathsf{reachable}(u, \Phi, v, \Psi)$ and $\mathsf{reachable}(v, \Psi, u, \Phi)$ have the same size. Both variants are used in our application later.

In the following, the computed table that is used for $\mathsf{reachable}(r, \mathsf{compls}, r, \mathsf{id})$ is denoted by T and $|T|$ is its size. Figure 6.4 shows the average ratio between the size of the computed table T and the size of r for vector-weighted voting games with weights between 0 and 10^6, for 30 players and different numbers of rules (100 samples per point).

From the figure we can see that the quota rate, as well as the number of rules, have a big impact on the ratio and hence, on the running time. At a quota rate of 50% for WVGs the average ratio is nearly 1. This is because, most games are decisive (strong and proper) for these parameters. We also notice, that for just one rule (resp. a WVG), the average ratio is never above 2. This observation motivates Theorem 6.11 below.

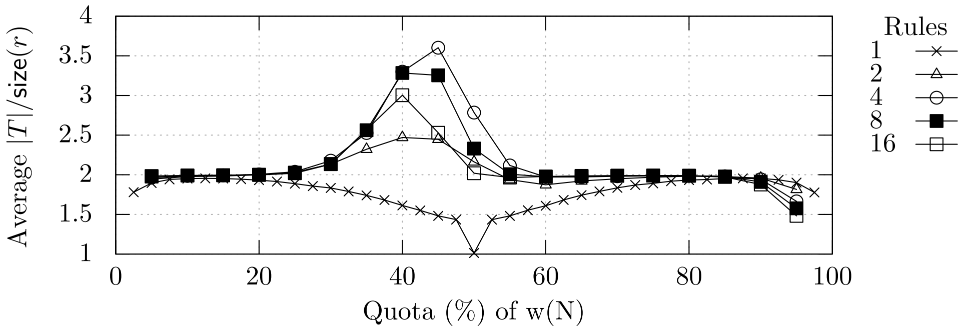


Figure 6.4.: Average ratio of the size of the computed table T to the size of the input QOBDD with root r for $\mathsf{reachable}(r, \mathsf{compls}, r, \mathsf{id})$ and 30 players.

Figure 6.5 shows another perspective. Here we have a fixed rate for the quota at 45% and a varying number of players. Instead of what we would expect from the upper bound in Theorem 6.4, the actual size of the computed table and hence, the running time grows much slower when the number of players increases.

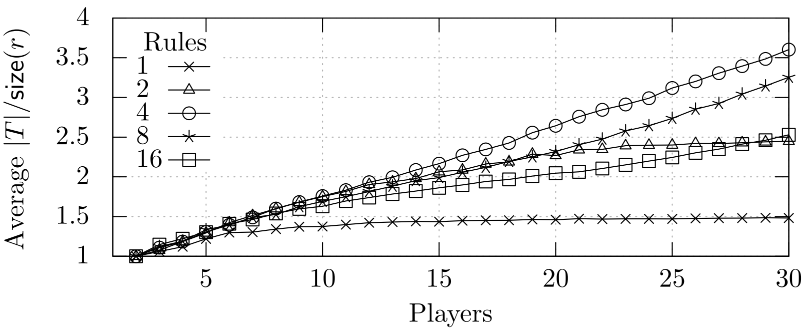


Figure 6.5.: Average ratio of the size of the computed table T to the size of the input QOBDD with root r for $\mathsf{reachable}(r, \mathsf{compls}, r, \mathsf{id})$ and a quota rate of 45%.

To ease the proof of the upcoming Theorem 6.11, we use the following statement that can easily be verified.

Proposition 6.10. *Assume that $(N, \mathsf{set}(r))$ is a* WVG *and that $[Q; w_1, \ldots, w_n]$ is a weighted representation of the game. For $i \in N$ let $u, v \in \mathsf{V}_i(r)$ be two different nodes. Let $R, S \subseteq \{1, \ldots, i-1\}$ be sets such that $r \overset{R}{\to} u$ and $r \overset{S}{\to} v$. Then it holds:*

$$u \subset v \iff w(R) < w(S)\,. \qquad \square$$

After these observations, we now prove our main result for the manipulator compls.

Theorem 6.11. *If $(N, \mathsf{set}(r))$ is a weighted simple game, then the size of the computed table T that is used in $\textit{reachable}(r, \textit{compls}, r, \textit{id})$ has size at most $2 \cdot \textit{size}(r)$.*

Proof. To keep things simple, we assume that the entries in T are pairs of nodes. This is appropriate, because we consider two fixed manipulators. Because compls does only map to nodes in $\mathsf{V}(r)$, it holds $T \subseteq \mathsf{V}(r) \times \mathsf{V}(r)$. For $i \in N$ by T_i we denote the subset of T whose nodes all have label i, that is, $T_i = T \cap (\mathsf{V}_i(r) \times \mathsf{V}_i(r))$. The sets $T_1, \ldots, T_n$ are disjoint and $T = T_1 \cup \cdots \cup T_n$.

Because the simple game $(N, \mathsf{set}(r))$ is weighted, there is a weighted representation $[Q; w_1, \ldots, w_n]$ for it, that we use in the remainder of the proof.

For all $i \in N$ we will show $|T_i| \leq 2|\mathsf{V}_i(r)|$, what then implies

$$|T| = \sum_{i \in N} |T_i| \leq \sum_{i \in N} 2|\mathsf{V}_i| = 2 \sum_{i \in N} |\mathsf{V}_i| = 2 \cdot \mathsf{size}(r)\,.$$

Let $i \in N$ be a player. The set $\{1, \ldots, i-1\}$ is denoted by I. First of all, if $(x, y) \in T_i$ then it is rather easy to see that also $(y, x) \in T_i$. Second, if $(x, x), (y, y) \in T_i$ then $x = y$. To see this, assume to the contrary that $x \neq y$. The QOBDD r is flat and therefore w.l.o.g. we can assume $x \supset y$. Because $(x, x), (y, y) \in T_i$, there are coalitions $R, S \subseteq I$ with $r \overset{R}{\to} x$, $r \overset{I \setminus R}{\to} x$ and $r \overset{S}{\to} y$, $r \overset{I \setminus S}{\to} y$. Hence, because both $w(R)$ and $w(I \setminus R)$ are greater than $w(S)$ and $w(I \setminus S)$, it follows:

$$w(I) = w(S) + w(I \setminus S) < w(R) + w(I \setminus R) = w(I)\,.$$

This a contradiction and therefore it holds $x = y$.

For the remainder of the proof we define an undirected graph $H = (\mathsf{V}_i, E)$ with edge set $E = \{\{x, y\} \mid (x, y) \in T_i, x \neq y\}$. Then we have $|T_i| \leq 2|E| + 1$. We will show that H is bipartite and without cycles what implies $|E| \leq |\mathsf{V}_i| - 1$ and finally,

$$|T_i| \leq 2|E| + 1 \leq 2|\mathsf{V}_i| - 1\,.$$

To see that H is bipartite, we need some notation. Let x be a node in V_i. We define the set $F(x)$ as $\{w(R) \mid \exists R \subseteq I : r \overset{R}{\to} x\}$. It can easily be seen that $F(x) \neq \emptyset$, because there is always a subset R of I with $r \overset{R}{\to} x$. Using $F(x)$, for x we denote the maximum and, respectively, minimum value in $F(x)$ by $\overline{f}(x) := \max F(x)$ and $\underline{f}(x) := \min F(x)$. Obviously, it holds $\underline{f}(x) \leq \overline{f}(x)$. Let $y \in \mathsf{V}_i$ be another node. It can rather easily be seen that

$$\underline{f}(x) > \overline{f}(y) \iff x \supset y\,. \tag{6.14}$$

It is a direct consequence that the sets $F(x)$ and $F(y)$ are disjoint, because being flat implies either $x \supset y$ or $x \subset y$.

We will use $\overline{f}$, $\underline{f}$ and the value $m := w(I)/2$ to define the bipartition of the graph H. The set V_i is obviously partitioned by:

$$P := \{v \in \mathsf{V}_i \mid \overline{f}(v) \geq m\}$$
$$Q := \{v \in \mathsf{V}_i \mid \overline{f}(v) < m\}\,.$$

To see that the graph is bipartite, let $\{x, y\} \in E$ be an edge. Because x, y are different nodes in V_i and the QOBDD is flat, w.l.o.g. we can assume $x \supset y$. We show that $x \in P$ and $y \in Q$. By the definition of the edge set E, there is a set $R \subseteq I$ with $r \overset{R}{\to} x$ and $r \overset{I \setminus R}{\to} y$. With Proposition 6.10 it follows $w(R) > w(I \setminus R)$. Because we have a WVG, the weights of $w(R)$ and $w(I \setminus R)$ sum up to $w(I)$. From this observation we obtain $w(R) > m > w(I \setminus R)$. The fact $w(R) > m$ can be seen by

$$w(R) > w(R)/2 + w(I \setminus R)/2 = w(I)/2 = m$$

and the remaining inequality can be shown analogously. From $w(R) > m$ it follows $\overline{f}(x) > m$ and therefore, $x \in P$. Beside that, from $w(I \setminus R) < m$ it follows $\underline{f}(y) < m$. In order to verify $y \in Q$, we have to show that also $\overline{f}(y) < m$. To see this, we assume to the contrary, that $\overline{f}(y) \geq m$. Using $\underline{f}(y) < m$ and $\{y, y\} \notin E$ the following two inequalities can be shown:

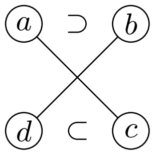
$$\begin{aligned} \underline{f}(y) &> m - (\overline{f}(y) - m)\,, \\ \overline{f}(y) &< m + (m - \underline{f}(y))\,. \end{aligned}$$

Because these inequalities are contradictory, it follows $\overline{f}(y) < m$. We show the first inequality for illustration. Let S be a subset of I with $w(S) = \overline{f}(y)$. Then S is a path from r to y. Because $\{y, y\} \notin E$, the complement $I \setminus S$ is not a path to y, that is, $\mathsf{node}(r, i, I \setminus S) \neq y$. Let z denote that node. It can easily be seen that $w(S) > m$ and hence, $w(I \setminus S) < m$. Therefore, using Proposition 6.10 we get $y \supset z$. With (6.14) it follows $\underline{f}(y) > \overline{f}(z)$ and we finally obtain:

$$\underline{f}(y) > \overline{f}(z) > w(I \setminus S) = w(I) - w(S) = 2m - w(S) = m - (\overline{f}(y) - m)\,.$$

To see that H is free of cycles, we show that a particular subgraph does not appear in H. Afterwards, we will show that a cycle would imply such a subgraph and hence, it will follow that H is free of cycles.

Let $a, b \in P$ and $c, d \in Q$ be nodes with $b \supset c$. By the choice of the nodes it follows $a \supset b \supset c \supset d$. We will prove $\{a, c\} \notin E$ or $\{b, d\} \notin E$.



Assume to the contrary that $\{a, c\}, \{b, d\} \in E$. Then there are $R, S \subseteq I$ such that $r \overset{R}{\to} a$, $r \overset{I \setminus R}{\to} c$ and $r \overset{S}{\to} b$, $r \overset{I \setminus S}{\to} d$. Because $a \supset b \supset c \supset d$ it holds

$$w(R) > w(S) > w(I \setminus R) > w(I \setminus S)\,.$$

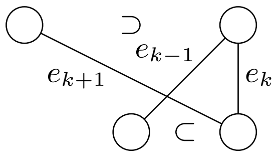
However, this implies $w(R) + w(I \setminus R) > w(S) + w(I \setminus S)$ which is a contradiction, because both sides are equal to $w(I)$.

For convenience, we consider an edge e as a pair whose first (resp. second) component is the edge's node in P (resp. Q). We define the following partial order on the edges E:

$$(u, v) \preceq (u', v') :\Longleftrightarrow u \supseteq u' \wedge v \subseteq v' . \tag{6.15}$$

For different edges e, e' with a common node (as in the path below), one of the subset relations in (6.15) is equality and the other is strict, because r is flat.

Assume to the contrary, that there is a cycle with edges $e_1, \dots, e_l$ of length l without multiple occurrences of an edge. Because the graph is bipartite, it holds $l \geq 4$. W.l.o.g. we assume that the edges are arranged such that $e_1 \preceq e_2 \preceq \dots \preceq e_k$ and k is maximal. If $k < l$ then it trivially holds $e_k \not\preceq e_{k+1}$. If, however, $k = l$ then we set $e_{l+1} := e_1$. By the antisymmetry of $\preceq$ and $l \geq 4$ it then follows $e_k \not\preceq e_{k+1} = e_1$. We also have $k > 1$, because otherwise it would hold $e_2 \preceq e_1$ due to $l \geq 4$ and the remark for (6.15) above. In conclusion, we have $e_{k-1} \preceq e_k \not\preceq e_{k+1}$ as follows:



This, however, is a contradiction to our forbidden subgraph above. □

6.2. Counting

In our considerations, a QOBDD represents a subset $\mathcal{A}$ of 2^N where $N = \{1, \dots, n\}$ is a set of players. For the root r of a QOBDD we will assume $\mathsf{var}(r) = 1$ in this section unless stated otherwise. When it comes to questions about the cardinality of $\mathcal{A}$, we know that for any QOBDD node v, the cardinality of $\mathsf{set}(v)$ can be computed recursively by the following formula:

$$|\mathsf{set}(v)| = \begin{cases} 0 & \text{if } v = \mathbb{O} \\ 1 & \text{if } v = \mathbb{I} \\ |\mathsf{set}(\mathsf{then}(v))| + |\mathsf{set}(\mathsf{else}(v))| & \text{otherwise} \end{cases} \tag{6.16}$$

It can easily be used to derive an algorithm that requires $\mathcal{O}(\mathsf{size}(v))$ arithmetic operations, if a computed table (e.g., a hash table) is used to avoid redundant computations. This result is well-known. Much more interesting is the computation of the cardinality $|\{S \in \mathcal{A} \mid i \in S\}|$ for each $i \in N$. Together with $|\mathcal{A}|$, these $n + 1$ values are known as Chow parameters (Chow 1961).

$\mathsf{chow}_{\mathcal{A}}(i)$

Definition 6.3. Let $\mathcal{A}$ be a subset of 2^N. The *Chow parameter for player $i \in N$*, denoted by $\mathsf{chow}_{\mathcal{A}}(i)$, or just $\mathsf{chow}(i)$ if there is no risk of confusion, is defined by:

$$\mathsf{chow}_{\mathcal{A}}(i) := |\{S \in \mathcal{A} \mid i \in S\}| . \qquad \square$$

Chow parameters can, for instance, be used to identify non-equally desirable players in simple games (see Section 6.3) and they are used in the identification of non-symmetric players in general Boolean functions (Möller, Mohnke, and Weber 1993). Furthermore, Chow parameters have the remarkable property that they characterize WvGs and threshold functions (Chow 1961), respectively, which in our context can be formulated like this. Let $(N, \mathcal{W}_1)$ and $(N, \mathcal{W}_2)$ be simple games. If $|\mathcal{W}_1| = |\mathcal{W}_2|$ as well as $\mathsf{chow}_{\mathcal{W}_1}(i) = \mathsf{chow}_{\mathcal{W}_2}(i)$ for each player $i \in N$ and if at least one of the games is a WvG, then $\mathcal{W}_1 = \mathcal{W}_2$. Consequently, (derivatives of) Chow parameters are frequently used in approaches to find weighted representations of a simple game, for instance, Palaniswamy, Goparaju, and Tragoudas (2010) use the so-called *modified Chow parameters*. As we will see in Section 6.7, Chow parameters are used in the Holler-Packel, Deegan-Packel and Shift power indices.

In a naive approach, the Chow parameters are computed one by one. However, as a main result of this section we will show that, among other values, the Chow parameters of all players can be computed by means of QOBDDs with just $\mathcal{O}(\mathsf{size}(r))$ arithmetic operations.

For the previously mentioned Deegan-Packel power index it is not sufficient to have the Chow parameters of the players. Instead of $|\{S \in \mathcal{A} \mid i \in S\}|$ we need the number of sets that contain a given number of players. Clearly speaking, for $i \in N$ and $k \in \{0, \dots, n\}$ we need the cardinality

$$|\{S \in \mathcal{A} \mid i \in S \wedge |S| = k\}| \,. \tag{6.17}$$

The ability to compute these numbers for all combinations of i and k without multiple traversals of the QOBDD representation for $\mathcal{A}$ requires some generalizations and also, an additional factor of n in the running time.

Throughout this section, we use a variant of the *Cartesian product*, denoted by $\times$, which for disjoint sets X, Y and $\mathcal{A} \subseteq 2^X, \mathcal{B} \subseteq 2^Y$ is defined by $\mathcal{A} \times \mathcal{B}$

$$\mathcal{A} \times \mathcal{B} := \{R \cup S \mid R \in \mathcal{A},\, S \in \mathcal{B}\}\,, \tag{6.18}$$

so that we have sets in the result instead of pairs.

The following statement is a prototype lemma that is used multiple times in this section. For $i \in N$ we say that a predicate $P : 2^N \to \mathbb{B}$ does *only depend on* $\{i,\dots,n\}$ if

$$\forall S \in 2^N : P(S) = P(S \cap \{i, \dots, n\})\,.$$

In the remainder of this section we will choose P such that, for instance, $P(S)$ is satisfied if and only if $i \in S$. This will be the case for the Chow parameter of player i.

Lemma 6.12. *For a* QOBDD *with root r and a player $i \in N$ let $P : 2^N \to \mathbb{B}$ be a predicate that only depends on $\{i, \dots, n\}$. Then it holds:*

$$\{S \in \mathsf{set}(r) \mid P(S)\} = \bigcup_{v \in \mathsf{V}_i} \mathsf{paths}(r, v) \times \{S \in \mathsf{set}(v) \mid P(S)\}\,.$$

Proof. With $D := \{1, \dots, i-1\}$ it holds:

$$\begin{aligned}
&\{S \in \mathsf{set}(r) \mid P(S)\} \\
=&\{R \cup S \mid R \subseteq D,\, S \subseteq N \setminus D,\, R \cup S \in \mathsf{set}(r),\, P(S)\} && \text{(Choice of } P\text{)} \\
=&\bigcup_{v \in \mathsf{V}_i} \{R \cup S \mid R \subseteq D,\, r \overset{R}{\to} v,\, S \in \mathsf{set}(v),\, P(S)\} && \text{(Thm. 3.3)} \\
=&\bigcup_{v \in \mathsf{V}_i} \{R \subseteq D \mid r \overset{R}{\to} v\} \times \{S \in \mathsf{set}(v) \mid P(S)\} \\
=&\bigcup_{v \in \mathsf{V}_i} \mathsf{paths}(r, v) \times \{S \in \mathsf{set}(v) \mid P(S)\}\,. && \text{(Def. } \mathsf{paths}\text{)}
\end{aligned}$$

This completes the proof. □

By choosing P as a predicate that is always satisfied (and hence, does not depend on anything), we can easily verify the following equation for any QOBDD root r:

$$\begin{aligned}
|\mathsf{set}(r)| &= |\bigcup_{v \in \mathsf{V}_i} \mathsf{paths}(r, v) \times \mathsf{set}(v)| && \text{(Lem. 6.12)} \\
&= \sum_{v \in \mathsf{V}_i} |\mathsf{paths}(r, v) \times \mathsf{set}(v)| && \text{(Disjoint)} \\
&= \sum_{v \in \mathsf{V}_i} |\mathsf{paths}(r, v)| \cdot |\mathsf{set}(v)|\,. && \text{(Def. } \times\text{)}
\end{aligned}$$

Our next step is to define a generalized version of the cardinality $|\mathcal{A}|$ of $\mathcal{A} \subseteq 2^N$ which takes the size of the sets in $\mathcal{A}$ into account as in (6.17). Notice that in this section, subscripts of elements in a vector usually start with 0 instead of 1, because the subscript has the meaning of the size of a set and a set can be empty.

maxsize($\mathcal{A}$)
$\|\mathcal{A}\|$

Definition 6.4. Let $\mathcal{A}$ be a non-empty subset of 2^N. By maxsize($\mathcal{A}$) we denote the size of the largest set in $\mathcal{A}$, that is, $\max_{S \in \mathcal{A}} |S|$. We set $m :=$ maxsize($\mathcal{A}$). By $\|\mathcal{A}\|$ we denote the vector $(v_0, \dots, v_m) \in \mathbb{N}_0^{m+1}$ such that for each $j \in \{0, \dots, m\}$ it holds:

$$v_j = |\{S \in \mathcal{A} \mid j = |S|\}|\,.$$

For the empty set we define $\|\emptyset\|$ as the vector (0). □

For $a, b \in \mathbb{N}$ and vectors $u \in \mathbb{N}_0^a$ and $v \in \mathbb{N}_0^b$ the component-wise addition of u and v usually requires vectors of same size, that is, $a = b$ in our case. In the remainder of this section, we will be more liberal. If $a \neq b$ then the missing components in u or v are assumed to be 0. The result of $u + v$ has $\max\{a, b\}$ components. For instance, $(1) + (2, 3, 4) + (5, 6)$ would be calculated as

$$(1, 0, 0) + (2, 3, 4) + (5, 6, 0) = (8, 9, 4)\,.$$

For disjoint sets $\mathcal{A}, \mathcal{B} \subseteq 2^N$ it is easy to see, that similar to $|\mathcal{A} \cup \mathcal{B}| = |\mathcal{A}| + |\mathcal{B}|$, it holds $\|\mathcal{A} \cup \mathcal{B}\| = \|\mathcal{A}\| + \|\mathcal{B}\|$. By induction, this result can easily be extended to a finite number of disjoint sets $\mathcal{A}_1, \ldots, \mathcal{A}_m$:

$$\|\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_m\| = \|\mathcal{A}_1\| + \cdots + \|\mathcal{A}_m\| .$$

Let $i \in N$ be a player and let $\mathcal{A} \subseteq 2^{\{1,\ldots,i-1\}}$ and $\mathcal{B} \subseteq 2^{\{i,\ldots,n\}}$ be (finite) sets. We have $|\mathcal{A} \times \mathcal{B}| = |\mathcal{A}| \cdot |\mathcal{B}|$. This, however, does not work with $\|\mathcal{A} \times \mathcal{B}\|$ anymore, because the ordinary multiplication is insufficient and we do not have an appropriate alternative for vectors, yet.

Any set $T \in \mathcal{A} \times \mathcal{B}$ with j elements can be partitioned into $T = R \cup S$ such that $R \in \mathcal{A}$, $S \in \mathcal{B}$ and $|R| + |S| = j$. Therefore, by the definition of $\| \cdot \|$ it is:

$$\|\mathcal{A} \times \mathcal{B}\|_j = \sum_{k=0}^{j} \|\mathcal{A}\|_k \cdot \|\mathcal{B}\|_{j-k} . \tag{6.19}$$

Based on this observation in the following we define a vector multiplication $\odot$ such that $\|\mathcal{A} \times \mathcal{B}\| = \|\mathcal{A}\| \odot \|\mathcal{B}\|$.

Definition 6.5. For vectors $\vec{u} = (u_0, \ldots, u_a) \in \mathbb{N}_0^{a+1}$ and $\vec{v} = (v_0, \ldots, v_b) \in \mathbb{N}_0^{b+1}$ we define the vector $\vec{u} \odot \vec{v} \in \mathbb{N}_0^{a+b+1}$ for $j \in \{0, \ldots, a+b\}$ by $\vec{u} \odot \vec{v}$

$$(\vec{u} \odot \vec{v})_j := \sum \{u_x \cdot v_y \mid 0 \leq x \leq a,\, 0 \leq y \leq b,\, x + y = j\} ,$$

where $\sum X$ sums up all the elements in a set X. □

For $\vec{u} \in \mathbb{N}_0^{a+1}$ and $\vec{v} \in \mathbb{N}_0^{b+1}$ the vector $\vec{u} \odot \vec{v}$ can be computed with $(a+1)(b+1)$ arithmetic operations using two **for**-loops. For future reference we state:

Lemma 6.13. *For a player $i \in N$ and sets $\mathcal{A} \subseteq 2^{\{1,\ldots,i-1\}}$ and $\mathcal{B} \subseteq 2^{\{i,\ldots,n\}}$ it holds $\|\mathcal{A} \times \mathcal{B}\| = \|\mathcal{A}\| \odot \|\mathcal{B}\|$.* □

By using this result we can now prove our second prototype lemma which we will use multiple times in the remainder of this section. An explanation is given below.

Lemma 6.14. *Let r be the root of a* QOBDD, *let $\mathcal{A}$ be a subset of 2^N and let $i \in N$ be a player. By D we denote the set $\{1, \ldots, i-1\}$. If f and g are functions with domain $\mathsf{V}_i(r)$ such that for each $v \in \mathsf{V}_i(r)$ it holds $f(v) \subseteq 2^D$, $g(v) \subseteq 2^{N \setminus D}$ and $\mathcal{A}$ is the disjoint union*

$$\mathcal{A} = \bigcup_{v \in \mathsf{V}_i} f(v) \times g(v) , \tag{6.20}$$

then it holds

$$|\mathcal{A}| = \sum_{v \in \mathsf{V}_i} |f(v)| \cdot |g(v)| \qquad \textit{and} \qquad \|\mathcal{A}\| = \sum_{v \in \mathsf{V}_i} \|f(v)\| \odot \|g(v)\| .$$

Proof. As an example, you can think of $\mathsf{paths}(r,v)$ as $f(v)$ and $\mathsf{set}(v)$ as $g(v)$. Let the pair $m, \circ$ be either $|\cdot|, \cdot$ or $\|\cdot\|, \odot$. In general it holds:

$$\begin{aligned} m(\mathcal{A}) &= m(\bigcup_{v\in \mathsf{V}_i} f(v)\times g(v)) && \text{(Eq. 6.20)}\\ &= \sum_{v\in \mathsf{V}_i} m(f(v)\times g(v)) && \text{(Disjoint union)}\\ &= \sum_{v\in \mathsf{V}_i} m(f(v))\circ m(g(v))\,. && \text{(Lem. 6.13 for } m=\odot) \end{aligned}$$

This completes the proof. □

The lemma frees us from tedious work later at the cost of an rather abstract statement. However, things become clearer by considering a concrete example. Assume $\mathsf{set}(r) = \mathcal{W}$ for the QOBDD. The first question is, why $\mathcal{A}$ can be different from $\mathsf{set}(r)$. At this point, remember the Chow parameters from the beginning of this section. For player $i \in N$ the set $\{S \in \mathcal{W} \mid i \in S\}$ is used for $\mathsf{chow}_{\mathcal{W}}(i)$ which is different from $\mathcal{W}$. So assume $\mathcal{A}$ is that set. The second question is the role of the functions f and g. These are very much related to paths and set, respectively. In the case of $\mathsf{chow}_{\mathcal{W}}(i)$, f is chosen as $v \mapsto \mathsf{paths}(r,v)$ and g is chosen as $v \mapsto \{S \in \mathsf{set}(v) \mid i \in S\}$.

As another example, we reconsider a result that we have already seen before for $|\cdot|$ and the scalar multiplication. Using Lemma 6.14, this time, we can also use $\|\cdot\|$ and $\odot$.

Corollary 6.15. *For any* QOBDD *with root r and any $i \in N$ it holds:*

$$\begin{aligned} |\mathsf{set}(r)| &= \sum_{v\in \mathsf{V}_i} |\mathsf{paths}(r,v)|\cdot|\mathsf{set}(v)|\,,\\ \|\mathsf{set}(r)\| &= \sum_{v\in \mathsf{V}_i} \|\mathsf{paths}(r,v)\|\odot\|\mathsf{set}(v)\|\,. \end{aligned}$$

Proof. Let P be a predicate on 2^N that is always satisfied. Then by Lemma 6.12 for $\mathsf{set}(r)$, P and any $i \in N$ we have:

$$\mathsf{set}(r) = \bigcup_{v\in \mathsf{V}_i} \mathsf{paths}(r,v)\times \mathsf{set}(v)\,.$$

We apply Lemma 6.14 with $\mathsf{set}(r)$ as $\mathcal{A}$, $\mathsf{paths}(r,v)$ as $f(v)$ and $\mathsf{set}(v)$ as $g(v)$. The prerequisites of the lemma are then trivially fulfilled. Hence, we directly obtain the claimed equations. □

The component-wise definition of $\odot$ is sometimes laborious as in Lemma 6.16 below, where we need the result of $(0,1)\odot\vec{u}$ for some vector $\vec{u}$. In these situations it is more convenient to use the following *right-shift* operation instead. Let $\vec{u} = (u_0,\dots,u_a) \in \mathbb{N}_0^{a+1}$ be a vector and let $k \in \mathbb{N}_0$ be a scalar. The vector $\vec{u} \gg k \in \mathbb{N}_0^{a+k+1}$ is defined for

$\vec{u} \gg k$

$j \in \{0, \ldots, a+k\}$ by:

$$(\vec{u} \gg k)_j = \begin{cases} 0 & \text{if } j < k \\ u_{j-k} & \text{otherwise} \end{cases} . \tag{6.21}$$

Hence, $\vec{u} \gg k$ is equal to $(0, \ldots, 0, u_0, \ldots, u_a)$ with k zeros at the front. The vector can be computed in $\mathcal{O}(k + a + 1)$ steps. We use the right-shift as follows:

Lemma 6.16. *For $i \in N$ and $\mathcal{A} \subseteq 2^{N \setminus \{i\}}$ we have the equations*

$$|\{i\} \times \mathcal{A}| = |\mathcal{A}| \qquad \text{and} \qquad \|\{i\} \times \mathcal{A}\| = \|\mathcal{A}\| \gg 1 .$$

Proof. The case of $|\cdot|$ is trivial. The remaining case holds due to:

$$\|\{i\} \times \mathcal{A}\| = \|\{1\}\| \odot \|\mathcal{A}\| = (0,1) \odot \|\mathcal{A}\| = \|\mathcal{A}\| \gg 1 . \qquad \square$$

We can now state our first main result regarding the Chow parameters for the players. An example for the statement is provided at the end of this section.

Theorem 6.17. *For any* QOBDD *with root r and each $i \in N$ it holds:*

$$\mathsf{chow}_{\mathsf{set}(r)}(i) = \sum_{v \in \mathsf{V}_i} |\mathsf{paths}(r,v)| \cdot |\mathsf{set}(\mathsf{then}(v))| ,$$
$$\|\{S \in \mathsf{set}(r) \mid i \in S\}\| = \sum_{v \in \mathsf{V}_i} \|\mathsf{paths}(r,v)\| \odot (\|\mathsf{set}(\mathsf{then}(v))\| \gg 1) .$$

Proof. For a coalition $S \subseteq N$ and a player $i \in N$ we define the predicate $P : 2^N \to \mathbb{B}$ to be satisfied for S if and only if $i \in S$. By using P and Lemma 6.12 the coalitions in $\mathsf{set}(r)$ containing player i can be rewritten to:

$$\{S \in \mathsf{set}(r) \mid i \in S\} = \bigcup_{v \in \mathsf{V}_i} \mathsf{paths}(r,v) \times \{S \in \mathsf{set}(v) \mid i \in S\} . \tag{6.22}$$

By the properties of $\mathsf{set}(v)$, for the right-hand side of $\times$ in (6.22) it holds for any $v \in \mathsf{V}_i$:

$$\{S \in \mathsf{set}(v) \mid i \in S\} = \{S \cup \{i\} \mid S \in \mathsf{set}(\mathsf{then}(v))\} = \{i\} \times \mathsf{set}(\mathsf{then}(v)) .$$

We can now apply Lemma 6.14 with $\mathsf{paths}(r,v)$ as $f(v)$ and $\{i\} \times \mathsf{set}(\mathsf{then}(v))$ as $g(v)$. Thereby we obtain the equations

$$|\{S \in \mathsf{set}(r) \mid i \in S\}| = \sum_{v \in \mathsf{V}_i} |f(v)| \cdot |\{i\} \times \mathsf{set}(\mathsf{then}(v))|$$

and

$$\|\{S \in \mathsf{set}(r) \mid i \in S\}\| = \sum_{v \in \mathsf{V}_i} \|f(v)\| \odot \|\{i\} \times \mathsf{set}(\mathsf{then}(v))\| .$$

Now, the claimed equations follow from Lemma 6.16. $\square$

The importance of this result lies in the fact, that if the values for $|\mathsf{paths}(r,v)|$ and $|\mathsf{set}(v)|$ were known in advance for every node in $\mathsf{V}(r)$, then $\mathcal{O}(|\mathsf{V}_i(r)|)$ operations of type $+,\cdot$ or $+,\odot,\gg$ would be sufficient to compute $\mathsf{chow}_{\mathsf{set}(r)}(i)$ and $\|\{S \in \mathsf{set}(r) \mid i \in S\}\|$, respectively. We will see later in this section, how these values can be obtained.

Besides the Chow parameters we are also interested in counting the so-called swings for a player. Swings are fundamental for the Banzhaf and Shapley-Shubik power indices in Section 6.7.

$\mathsf{swings}_{\mathcal{A}}(i)$

Definition 6.6. Let $\mathcal{A} \subseteq 2^N$ be an up-set. For a coalition $S \in \mathcal{A}$ we say *player $i \in N$ is critical for S*, if $i \in S$ and $S - i \notin \mathcal{A}$. In that case, we call S *a swing for player i*. We denote the set of all swings for players i by $\mathsf{swings}_{\mathcal{A}}(i)$ and $\mathsf{swings}(i)$, respectively, if there is not risk of confusion. □

Usually, the swings are used in the context of simple games and the set $\mathcal{A}$ equals $\mathcal{W}$ as in the following example.

Example 6.1. Consider players A, B, C, D and winning coalitions AB, AC, BCD, ABC, ACD and $ABCD$. For instance, because neither player A, nor B is winning alone, both players are critical for the coalition AB. In comparison, no player is critical for $ABCD$. In a minimal winning coalition all players are critical. □

Similar to Theorem 6.17 for the Chow parameters, we can show the following result for the swings:

Theorem 6.18. *For a* QOBDD *with root r such that* $\mathsf{set}(r)$ *is an up-set and for each level $i \in N$ it holds:*

$$|\mathsf{swings}_{\mathsf{set}(r)}(i)| = \sum_{v \in \mathsf{V}_i} |\mathsf{paths}(r,v)| \cdot (|\mathsf{set}(\mathsf{then}(v))| - |\mathsf{set}(\mathsf{else}(v))|) \ ,$$

$$\|\mathsf{swings}_{\mathsf{set}(r)}(i)\| = \sum_{v \in \mathsf{V}_i} \|\mathsf{paths}(r,v)\| \odot ((\|\mathsf{set}(\mathsf{then}(v))\| - \|\mathsf{set}(\mathsf{else}(v))\|) \gg 1) \ .$$

Proof. Similar to the proof of Theorem 6.17, the prototype lemmas are used. For a coalition $S \subseteq N$ we define the predicate $P : 2^N \to \mathbb{B}$ to be satisfied for S if and only if $i \in S$ and $S - i \notin \mathsf{set}(r)$. By using P and Lemma 6.12, the swings of player i can be rewritten to:

$$\mathsf{swings}(i) = \bigcup_{v \in \mathsf{V}_i} \mathsf{paths}(r,v) \times \{S \in \mathsf{set}(v) \mid i \in S,\ S - i \notin \mathsf{set}(v)\} \,. \tag{6.23}$$

By the properties of $\mathsf{set}(v)$, for the right-hand side of $\times$ in (6.23) we have for any $v \in \mathsf{V}_i$:

$$\{S \in \mathsf{set}(v) \mid i \in S,\ S - i \notin \mathsf{set}(v)\} = \{i\} \times (\mathsf{set}(\mathsf{then}(v)) \setminus \mathsf{set}(\mathsf{else}(v))) \ .$$

We can now apply Lemma 6.14 with $f(v)$ as $\mathsf{paths}(r,v)$ and $g(v)$ as $\{i\} \times \mathsf{set}(\mathsf{then}(v))$, so that with $D(v) := \mathsf{set}(\mathsf{then}(v)) \setminus \mathsf{set}(\mathsf{else}(v))$ we obtain the equations

$$|\mathsf{swings}(i)| = \sum_{v \in \mathsf{V}_i} |f(v)| \cdot |\{i\} \times D(v)|$$

and

$$\|\mathsf{swings}(i)\| = \sum_{v \in \mathsf{V}_i} \|f(v)\| \odot \|\{i\} \times D(v)\| \,.$$

Because the set $\mathsf{set}(r)$ is an up-set, it holds $\mathsf{then}(v) \supset \mathsf{else}(v)$ for any node $v \in \mathsf{V}$. As a consequence, for $m \in \{|\cdot|, \|\cdot\|\}$ we get:

$$m(D(v)) = m(\mathsf{set}(\mathsf{then}(v))) - m(\mathsf{set}(\mathsf{else}(v)))\,.$$

The claimed equations finally follow from Lemma 6.16. □

What remains is to compute the values $|\mathsf{set}(v)|$, $|\mathsf{paths}(v)|$ and $\|\mathsf{set}(v)\|$, $\|\mathsf{paths}(r,v)\|$, respectively, for any inner node $v \in \mathsf{V}(r) \cup \{\mathbb{O}, \mathbb{I}\}$ and a QOBDD with root r. In the following we shift towards a more algorithmic perspective.

We use integers variables c_v and p_v to store the values $|\mathsf{set}(v)|$ and $|\mathsf{paths}(r,v)|$, respectively, for a node $v \in \mathsf{V}(r) \cup \{\mathbb{O}, \mathbb{I}\}$. These values are initially 0 and can be obtained as follows. We know that $|\mathsf{set}(\mathbb{I})| = 1$ and $|\mathsf{set}(\mathbb{O})| = 0$, and therefore we set $c_{\mathbb{I}} \leftarrow 1$ and $c_{\mathbb{O}} \leftarrow 0$. The remaining values for c_v can be computed by a depth-first traversal of the QOBDD with root r and by using the formula in (6.16) from the beginning of the section.

The empty set is a path from r to r. Therefore we set $p_r \leftarrow 1$. For the remaining values p_v we use the following short algorithm:

```
for $i = 1$ to $n$ do
    for $v \in \mathsf{V}_i(r)$ do
        $p_{\mathsf{then}(v)} \leftarrow p_{\mathsf{then}(v)} + p_v$
        $p_{\mathsf{else}(v)} \leftarrow p_{\mathsf{else}(v)} + p_v$
```

The computation of all the values c_v and p_v, $v \in \mathsf{V}(r) \cup \{\mathbb{O}, \mathbb{I}\}$, requires $\mathcal{O}(\mathsf{size}(r))$ steps and arithmetic operations of type $+$, respectively.

Computing the values $\|\mathsf{set}(v)\|$ and $\|\mathsf{paths}(r)\|$ for each node in $\mathsf{V}(r) \cup \{\mathbb{O}, \mathbb{I}\}$ is more expensive, because we have vectors in this case. We use integer vectors (e.g., arrays of variables) C_v and P_v to store the values $\|\mathsf{set}(v)\|$ and $\|\mathsf{paths}(r,v)\|$, respectively, for a node $v \in \mathsf{V}(r) \cup \{\mathbb{O}, \mathbb{I}\}$. These values are initialized with the vector (0). We know that $\mathsf{set}(\mathbb{I}) = \{\emptyset\}$ and $\mathsf{set}(\mathbb{O}) = \emptyset$, so we set $C_{\mathbb{I}} \leftarrow (1)$ and $C_{\mathbb{O}} \leftarrow (0)$. As in the case of $|\cdot|$, we use a depth-first traversal of the QOBDD with root r to compute the remaining values of C_v. We make use of the fact, that for an inner node v we have

$$\|\mathsf{set}(v)\| = (\|\mathsf{set}(\mathsf{then}(v))\| \gg 1) + \|\mathsf{set}(\mathsf{else}(v))\|\,. \tag{6.24}$$

If S is in $\mathsf{set}(\mathsf{then}(v))$ then the set $S+i$ is in $\mathsf{set}(v)$ and it holds $|S+i| = |S|+1$. This is the reason for the right-shift in (6.24). Each appearing vector has length at most $n+1$.

Because $r \xrightarrow{\emptyset} r$, there is a path without an 1-edge to r and hence, we set $P_r \leftarrow (1)$. Similar to the case of $|\cdot|$, for the remaining values P_v we use the following short algorithm:

```
for $i = 1$ to $n$ do
    for $v \in \mathsf{V}_i(r)$ do
        $P_{\mathsf{then}(v)} \leftarrow P_{\mathsf{then}(v)} + (P_v \gg 1)$
        $P_{\mathsf{else}(v)} \leftarrow P_{\mathsf{else}(v)} + P_v$
```

The computation of all the values C_v and P_v, $v \in \mathsf{V}(r) \cup \{\mathbb{O}, \mathbb{I}\}$, requires $\mathcal{O}(n \cdot \mathsf{size}(r))$ steps and arithmetic operations of type $+$, respectively. We state these results in the following theorem for reference.

Theorem 6.19. *For a* QOBDD *with root r, the values $|\mathsf{set}(v)|$ and $|\mathsf{paths}(r, v)|$ for* all *nodes $v \in \mathsf{V}(r) \cup \{\mathbb{O}, \mathbb{I}\}$ can be computed using $\mathcal{O}(\mathsf{size}(r))$ steps and arithmetic operations of type $+$, respectively. The values $\|\mathsf{set}(v)\|$ and $\|\mathsf{paths}(r, v)\|$ for* all *nodes $v \in \mathsf{V}(r) \cup \{\mathbb{O}, \mathbb{I}\}$ can be computed using $\mathcal{O}(n \cdot \mathsf{size}(r))$ steps and arithmetic operations of type $+$, respectively.* □

We consider an example to illustrate the idea.

Example 6.2. We consider the QOBDD in Figure 6.6 which represents the 3-person simple game $(N, \mathcal{W})$ with 4 winning coalitions $\mathcal{W} = \{AB, AC, BC, ABC\}$. The values in Figure 6.6 have been computed as described above. For instance, the value $\|\mathsf{set}(v_{2,1})\|$ can be computed using (6.24):

$$\begin{aligned}\|\mathsf{set}(v_{2,1})\| &= (\|\mathsf{set}(v_{3,1})\| \gg 1) + \|\mathsf{set}(v_{3,2})\| \\ &= ((1,1) \gg 1) + (0,1) = (0,1,1) + (0,1) = (0,2,1)\,.\end{aligned}$$

We use Theorem 6.17 to compute the values $x_i := \|\{S \in \mathcal{W} \mid i \in S\}\|$ for each player $i \in \{A, B, C\}$:

$$\begin{aligned}x_A &= \sum_{v \in \mathsf{V}_A} \|\mathsf{paths}(r, v)\| \odot (\|\mathsf{set}(\mathsf{then}(v)))\| \gg 1) = \|\mathsf{paths}(r, r)\| \odot (\|\mathsf{set}(v_{2,1})\| \gg 1) \\ &= (1) \odot ((0,2,1) \gg 1) = (1) \odot (0,0,2,1) = (0,0,2,1)\,. \\ x_B &= (0,1) \odot ((1,1) \gg 1) + (1) \odot ((0,1) \gg 1) = (0,0,1,1) + (0,0,1) = (0,0,2,1)\,. \\ x_C &= (0,0,1) \odot ((1) \gg 1) + (0,2) \odot ((1) \gg 1) + (1) \odot ((0) \gg 1) \\ &= (0,0,0,1) + (0,0,2) + (0) = (0,0,2,1)\,.\end{aligned}$$

It is not a coincident that all the values are equal. All players are equally desirable. The game $(N, \mathcal{W})$ has the weighted representation $[2; 1, 1, 1]$. □

6.3. Desirability Relation on Individuals

In this section, we apply our approach of using QOBDDs to represent simple games to the desirability relation on individuals $\preceq_I$ and the relation $\approx_I$ for equally desirable players. Having these relations at hand is a necessity to answer more complex questions. For

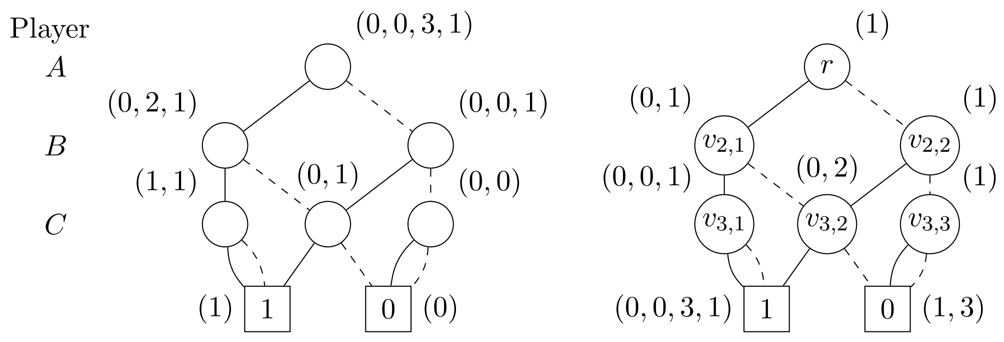


Figure 6.6.: Values for $\|\mathsf{set}(v)\|$ (left) and $\|\mathsf{paths}(r, v)\|$ (right) for each node $v \in \mathsf{V}(r)$.

instance, the desirability relation $\preceq_I$ is used to decide if a simple game is complete and it is necessary to obtain the set of the shift-minimal winning and shift-maximal losing coalitions in Section 6.6. Throughout this section, let $(N, \mathcal{W})$ be a simple game with players $N = \{1, \dots, n\}$ that is represented by the QOBDD with root r.

In the following discussion we consider the relations $\preceq_I$ and $\approx_I$ as oracles which can be *queried* for a pair i, j of players. The query will be answered in various ways. The fallback solution is to use an algorithm that operates on the QOBDD that represents $(N, \mathcal{W})$. There is one algorithm for each relation. Using these algorithms is rather expensive, so that one of the main intentions in this section is to reduce the number of cases when the algorithms have to be used, and therefore to reduce the costs to solve a given problem. The problems we are concerned with are the identification of the equivalence classes of $\approx_I$, the decision if a simple game is complete and directed, respectively, and the sorting problem for $\preceq_I$. As in Daskalakis, Karp, Mossel, Riesenfeld, and Verbin (2011) by *sorting problem* we refer to the disclosure of all the relationships between the elements in N.

We start with some implications of the Chow parameters of the players for the relations $\preceq_I$ and $\approx_I$, which can easily be verified. As we have seen in Section 6.2, the Chow parameters of all players for $\mathcal{W}$ can be computed with $\mathcal{O}(\mathsf{size}(r))$ arithmetic operations.

Lemma 6.20. *For players $i, j \in N$ it holds:*

1. *If $i \preceq_I j$ then $\mathsf{chow}(i) \leq \mathsf{chow}(j)$.*
2. *If $i \prec_I j$ then $\mathsf{chow}(i) < \mathsf{chow}(j)$.*
3. *If $i \approx_I j$ then $\mathsf{chow}(i) = \mathsf{chow}(j)$.* □

Furthermore, if we know in advance that $(N, \mathcal{W})$ is complete, e.g., because the QOBDD is built from a weighted representation, then $\preceq_I$ is entirely determined by the Chow parameters. This result is well-known in the context of Boolean functions and threshold functions; see Sheng (1969), Theorem 4.3.2.

Proposition 6.21. *If $(N, \mathcal{W})$ is complete, then for $i, j \in N$ it holds $i \preceq_I j$ if and only if $\mathsf{chow}_{\mathcal{W}}(i) \leq \mathsf{chow}_{\mathcal{W}}(j)$.*

Proof. The direction "$\Longrightarrow$" is obvious from Lemma 6.20. For the direction "$\Longleftarrow$" suppose $\mathsf{chow}(i) \leq \mathsf{chow}(j)$ and assume to the contrary that $i \not\preceq_I j$. Because the game is complete, $i \not\preceq_I j$ is equivalent to $i \succ_I j$ which implies the contradiction $\mathsf{chow}(i) > \mathsf{chow}(j)$. □

In general, if a query $i \preceq_I j$ or $i \approx_I j$ cannot be answered by Lemma 6.20 or Proposition 6.21, then we use the QOBDD for $(N, \mathcal{W})$ to answer it.

Theorem 6.22. *For $i, j \in N$ it holds $i \preceq_I j$ if and only if*

$$\mathit{subseteq}(r, \mathit{without}_j \circ \mathit{remove}_i, r, \mathit{without}_i \circ \mathit{remove}_j) = \mathit{true}$$

and $i \approx_I j$ if and only if

$$\mathit{equal}(r, \mathit{without}_j \circ \mathit{remove}_i, r, \mathit{without}_i \circ \mathit{remove}_j) = \mathit{true}\,.$$

We can decide in expected time $\mathcal{O}(\sum_{k=1}^{n} \mathit{width}_k(r)^2)$ and thus, $\mathcal{O}(n \cdot \mathit{width}(r)^2)$ if $i \preceq_I j$, and we can decide in expected time $\mathcal{O}(\mathit{size}(r))$ if $i \approx_I j$.

Proof. For the first equivalence we use the definition of $i \preceq_I j$, which is

$$i \preceq_I j \iff \forall S \in 2^N : (i \notin S \wedge j \notin S \wedge S + i \in \mathcal{W} \Rightarrow S + j \in \mathcal{W})\ ,$$

and we use the general equivalence $\varphi \wedge \psi \implies \rho$ if and only if $\varphi \wedge \psi \implies \varphi \wedge \rho$ for formulas φ, ψ, ρ. By using the manipulators from Lemma 6.7 and 6.8, respectively, for $S \in 2^N$ we obtain:

$$\begin{aligned}
&(i \notin S \wedge j \notin S \wedge S + i \in \mathcal{W}) \Rightarrow S + j \in \mathcal{W} \\
\iff &(i \notin S \wedge j \notin S \wedge S + i \in \mathsf{set}(r)) \Rightarrow (i \notin S \wedge j \notin S \wedge S + j \in \mathsf{set}(r)) \\
\iff &(j \notin S \wedge S \in \mathsf{set}_{\mathsf{remove}_i}(r)) \Rightarrow (i \notin S \wedge S \in \mathsf{set}_{\mathsf{remove}_j}(r)) \\
\iff &S \in \mathsf{set}_{\mathsf{without}_j \circ \mathsf{remove}_i})(r) \Rightarrow S \in \mathsf{set}_{\mathsf{without}_i \circ \mathsf{remove}_j}(r)\,.
\end{aligned}$$

The missing step is the application of $\mathsf{subseteq}$ from Eq. (6.6) on page 73. By using Theorem 6.4, the expected running time for the first case is

$$\mathcal{O}(\sum_{k=1}^{n} \mathsf{width}_k(r, \mathsf{without}_j \circ \mathsf{remove}_i) \cdot \mathsf{width}_k(r, \mathsf{without}_i \circ \mathsf{remove}_j))\,.$$

For $k \in \{1, \ldots, n\}$ both widths have upper bound $\mathsf{width}_k(r) + 1$ by Proposition 6.9. The remaining step follows from $(\mathsf{width}_k(r) + 1)^2 \in \mathcal{O}(\mathsf{width}_k(r)^2)$.

Similar arguments can be used for the case of $i \approx_I j$ with equal instead of $\mathsf{subseteq}$. By Proposition 6.9 it holds $\mathsf{size}(r, \Phi) \leq \mathsf{size}(r) + (n - k)$ for any manipulator under consideration. Therefore, by using Corollary 6.5 and $\mathsf{size}(r) \geq n$ we obtain

$$\mathcal{O}(\mathsf{size}(r, \Phi)) \leq \mathcal{O}(\mathsf{size}(r) + (n - k)) \leq \mathcal{O}(\mathsf{size}(r))$$

as the expected running time to decide if $i \approx_I j$. □

After these preparations we can now discuss the problems in the domain of simple games. We first decide if the simple game $(N, \mathcal{W})$ is complete. To answer that question requires at most $n-1$ queries to $\preceq_I$ that use the fallback solution. To see this, obtain from the Chow parameters (for $\mathcal{W}$) an ordering π of N such that

$$\mathsf{chow}(\pi(1)) \geq \cdots \geq \mathsf{chow}(\pi(n)) .$$

By Proposition 6.21 the game is complete if $\pi(i) \succeq_I \pi(i+1)$ for $i = 1, \ldots, n-1$. This can be decided with at most $n-1$ queries to $\preceq_I$. The game is directed, if π *can* be chosen as the identity function on N, that is, $\pi(i) = i$ for $i = 1, \ldots, n$.

Proposition 6.23. *If the Chow parameters of the players are known, then we can decide in expected time $\mathcal{O}(n^2 \cdot \mathsf{width}(r)^2)$ if $(N, \mathcal{W})$ is complete.*

Proof. The running time is dominated by the $n-1$ queries each of which requires expected time $\mathcal{O}(n \cdot \mathsf{width}(r)^2)$. □

To identify the equivalence classes of $\approx_I$, that is, the t types $N_1, \ldots, N_t$, at most $\mathcal{O}(tn)$ queries to $\approx_I$ are necessary that use the fallback solution. Due to *3.* of Lemma 6.20 we can use the Chow parameters again. Let C be the equivalence relation on N, such that $(i,j) \in C$ if and only if $\mathsf{chow}(i) = \mathsf{chow}(j)$, and for $i \in N$ let $[i]_C$ denote the equivalence class of i w.r.t. C. It is rather easy to see that the quotient set $N/{\approx_I}$ is a refinement of the partition N/C.

To find out the equivalence classes of $\approx_I$ in $[i]_C$ for player $i \in N$ at most

$$|[i]_C/{\approx_I}| \cdot |[i]_C|$$

queries to $\approx_I$ are necessary. Hence to identify the types $N_1, \ldots, N_t$ requires

$$\sum_{X \in N/C} |X/{\approx_I}| \cdot |X| \leq \sum_{X \in N/C} t|X| \leq tn$$

queries to $\approx_I$. In the worst-case, $t = n$ and all players have the same Chow parameters.

Proposition 6.24. *If the Chow parameters of the players are known, then the equivalence classes $N_1, \ldots, N_t$ of $\approx_I$ can be identified in expected time $\mathcal{O}(tn \cdot \mathsf{size}(r))$.* □

To solve the sorting problem for $\preceq_I$, we first compute the equivalence classes (resp. types) $N_1, \ldots, N_t$ of $\approx_I$. Using these we define a relation $\trianglelefteq_I$ on the types such that

$$N_p \trianglelefteq_I N_q :\Longleftrightarrow \exists i \in N_p, j \in N_q : i \preceq_I j$$

where $p, q = 1, \ldots, t$. The relation $\trianglelefteq_I$ is a partial order and we can use the algorithm by Daskalakis, Karp, Mossel, Riesenfeld, and Verbin (2011) which requires $\mathcal{O}(tw \log w \log(t/w))$ queries of kind $\trianglelefteq_I$ and thus, of kind $\preceq_I$, where w is the width[2] of $\trianglelefteq_I$. In practice some of the query results are implied by the Chow parameters using Lemma 6.20.

[2]The *width of a partial order* is the size of the maximum antichain, that is, the maximum number of incomparable elements.

Proposition 6.25. *If the Chow parameters of the players are known, then the sorting problem for $\preceq_I$ can be solved in expected time*

$$\mathcal{O}(tn \cdot \mathsf{size}(r) + tw \log w \log(t/w) n \cdot \mathsf{width}(r)^2)$$

where w is the width of the induced partial order of $\preceq_I$ on the equivalence classes of $\approx_I$.

Proof. It takes expected time $\mathcal{O}(tn \cdot \mathsf{size}(r))$ to find the types $N_1, \ldots, N_t$. The complexity of the algorithm Poset-Mergesort in Daskalakis et al. (2011) is (without queries) $\mathcal{O}(tw^2 \log w \log(t/w))$, if the width of the partial order is not known in advance. The number of queries is bounded by $\mathcal{O}(tw \log w \log(n)/w)$ in this case and each query takes expected time $\mathcal{O}(n \cdot \mathsf{width}(r)^2)$. The expected running time for the sorting algorithm is therefore dominated by the expected time for all queries. □

6.4. Blocking Coalitions, the Dual and Properties of Simple Games

In this section, we are concerned with some important properties of simple games, and on this occasion we address the so-called dual of a simple game, which is closely related to the properties, that we will discuss.

The problem to decide if a weighted voting game $[Q; w_1, \ldots, w_n]$ is proper, is co-NP-complete (Freixas, Molinero, Olsen, and Serna 2012). Therefore, we cannot expect to find a polynomial time algorithm for Wvgs unless NP=co-NP. Pseudo-polynomial algorithms exist though. We will arrive at a similar result by using the Qobdd for a Wvg.

We begin with the notions of a blocking coalition and the dual of a simple game as they are used by Taylor and Zwicker (1999). Throughout this section let $(N, \mathcal{W})$ be a simple game represented by the Qobdd with root r.

$\mathcal{W}^d$

Definition 6.7. A coalition $S \in 2^N$ is called *blocking* if $N \setminus S \notin \mathcal{W}$. The set of all blocking coalitions of $(N, \mathcal{W})$, which is denoted by $\mathcal{W}^d$, is an up-set. The simple game $(N, \mathcal{W}^d)$ is said to be the *dual* of $(N, \mathcal{W})$. □

The intuitive idea behind a blocking coalition is, that one has a collection of players that can prevent any collection of the remaining players from being a winning coalition. For example, in the Wvg $[4; 2, 1, 1, 1]$ the coalition AB is blocking because the weight of the remaining players is just 2. But AB is losing as well.

Because $\mathsf{set}(r) = \mathcal{W}$, the set represented by the Qobdd with root $\overline{r}$ are the losing coalitions $\mathcal{L}$ of $(N, \mathcal{W})$. By Def. 6.7 and the manipulator compls in Lemma 6.6 for every coalition $S \in 2^N$ it holds:

$$S \in \mathcal{W}^d \iff N \setminus S \notin \mathcal{W} \iff N \setminus S \in \mathsf{set}(\overline{r}) \iff S \in \mathsf{set}_{\mathsf{compls}}(\overline{r})\,. \tag{6.25}$$

By using the blocking coalitions and the dual of $(N, \mathcal{W})$, now it is straightforward to decide if $(N, \mathcal{W})$ is proper and strong, respectively.

Proposition 6.26 (Taylor and Zwicker (1999), Prop. 1.3.7)**.** *We have that $(N,\mathcal{W})$ is proper if and only if $\mathcal{W} \subseteq \mathcal{W}^d$ and $(N,\mathcal{W})$ is strong if and only if $\mathcal{W}^d \subseteq \mathcal{W}$.* □

Testing set inclusion is easily realizable using QOBDDs by the algorithm subseteq from Section 6.1. By Proposition 6.26 and (6.25) we have

$$(N,\mathcal{W}) \text{ proper} \iff \mathcal{W} \subseteq \mathcal{W}^d \iff \mathcal{W} \subseteq \mathsf{set}_{\mathsf{compls}}(\overline{r}) \iff \mathsf{subseteq}(r, \mathsf{id}, \overline{r}, \mathsf{compls})$$

and analogously

$$(N,\mathcal{W}) \text{ strong} \iff \mathcal{W}^d \subseteq \mathcal{W} \iff \mathsf{set}_{\mathsf{compls}}(\overline{r}) \subseteq \mathcal{W} \iff \mathsf{subseteq}(\overline{r}, \mathsf{compls}, r, \mathsf{id})\,.$$

For the running time we get the following result:

Theorem 6.27. *We can decide in expected time $\mathcal{O}(n \cdot \mathsf{width}(r)^2)$ if $(N,\mathcal{W})$ is proper and strong, respectively. Furthermore, if $(N,\mathcal{W})$ is weighted, then we can decide in expected time $\mathcal{O}(\mathsf{size}(r))$ if the game is proper and strong, respectively.*

Proof. We have $\mathsf{size}(r) = \mathsf{size}(\overline{r})$ and by Proposition 6.9 it holds $\mathsf{size}(r) = \mathsf{size}(r, \mathsf{compls})$. The expected running time for the general case follows from the expected running time of the algorithm apply_2 in Theorem 6.3 which is used by subseteq. The result for the case of a weighted simple game is a direct consequence of Theorem 6.11. □

Finally, the following well-known dependencies are sometimes useful in practice:

Proposition 6.28. *For the simple game $g = (N,\mathcal{W})$ it holds:*

1. *If g is decisive, then $|\mathcal{W}| = 2^{n-1}$.*
2. *If $|\mathcal{W}| = 2^{n-1}$ and g is not decisive, then g is neither proper nor strong.*
3. *If g is a weighted voting game, then $|\mathcal{W}| = 2^{n-1}$ if and only if g is decisive.*

Proof. It holds $|\mathcal{W}| + |\mathcal{L}| = 2^n$. The function $f : 2^N \to 2^N$, $S \mapsto N \setminus S$ is a bijection. It can easily be seen that if g is proper (resp. strong), then it holds $f(\mathcal{W}) \subseteq \mathcal{L}$ (resp. $f(\mathcal{L}) \subseteq \mathcal{W}$). For the first statement, if g is both proper and strong, then it follows $|\mathcal{W}| = |\mathcal{L}|$ and hence, $|\mathcal{W}| = 2^n/2$.

To see the second statement, we suppose that $|\mathcal{W}| = 2^{n-1}$ and that g is not decisive. Assume to the contrary that g is proper but not strong. The opposite case can be shown similarly. Because g is proper is holds $f(\mathcal{W}) \subseteq \mathcal{L}$. From $|\mathcal{W}| = 2^{n-1}$ it follows $|\mathcal{W}| = |\mathcal{L}|$. Therefore, by the injectivity of f we have $f(\mathcal{W}) = \mathcal{L}$. However, f is a bijection and because g is not strong, it holds $f(\mathcal{L}) \neq \mathcal{W}$. This contradicts the fact $f(\mathcal{W}) = \mathcal{L}$.

The last claim is a consequence of the fact, that a WVG is proper, strong or both. □

6.5. Minimal Winning and Maximal Losing Coalitions

The minimal winning coalitions $\mathcal{W}_{\min}$ and the maximal losing coalitions $\mathcal{L}_{\max}$ are fundamental for the computation of some power indices in Section 6.7 and they are necessary to compute the set of shift-minimal winning and shift-maximal losing coalitions in Section 6.6. In this section we present algorithms to compute the QOBDDs for $\mathcal{W}_{\min}$ and $\mathcal{L}_{\max}$, respectively. In the special case of a directed simple game we will show, that this can be done in time linear in the size of the QOBDD for $\mathcal{W}$.

Algorithm 5 *MinWin*(i, v)

```
Require: $v$ is a QOBDD node with label $i$ and $\mathsf{set}(v)$ is an up-set.
Ensure: Returns node $v'$ such that $\min \mathsf{set}(v) = \mathsf{set}(v')$.
1: if $v \in \{\mathbb{I}, \mathbb{O}\}$ then return $v$
2: else if $\mathsf{then}(v) = \mathsf{else}(v)$ then return $\mathsf{ite}(i, \mathbb{O}_{i+1}, \mathit{MinWin}(i+1, \mathsf{else}(v)))$
3: else return $\mathsf{ite}(i, \mathsf{minus}(\mathit{MinWin}(i+1, \mathsf{then}(v)), e), e)$
       where $e = \mathit{MinWin}(i+1, \mathsf{else}(v))$
```

Throughout this section let $(N, \mathcal{W})$ be a simple game with players $N = \{1, \ldots, n\}$ represented by the QOBDD with root r. We discuss the computation of the QOBDD for the set $\mathcal{W}_{\min}$ from r first. We use Algorithm 5 to this end. The explicit use of a computed table in *MinWin* has been omitted to ease the presentation.

Proposition 6.29. *It holds* $\min \mathsf{set}(r) = \mathsf{set}(\mathit{MinWin}(1, r))$.

Proof. The proof is by induction on the structure of the QOBDD. For the induction base assume v is a sink. If $v = \mathbb{O}$ then $\min \mathsf{set}(v) = \emptyset = \mathsf{set}(v)$. Otherwise if $v = \mathbb{I}$ then $\min \mathsf{set}(v) = \{\emptyset\} = \mathsf{set}(v)$.

For the induction step, let v be an inner node with label $i \in N$ and assume the statement holds for $\mathsf{then}(v)$ and $\mathsf{else}(v)$. If $\mathsf{then}(v) = \mathsf{else}(v)$ then:

$$\begin{aligned}
\mathsf{set}(\mathit{MinWin}(i, v)) &= \mathsf{set}(\mathsf{ite}(i, \mathbb{O}_{i+1}, \mathit{MinWin}(i+1, \mathsf{else}(v)))) && \text{(Alg.)}\\
&= \mathsf{set}(\mathit{MinWin}(i+1, \mathsf{else}(v))) && \text{(Def. } \mathsf{set}\text{)}\\
&= \min \mathsf{set}(\mathsf{else}(v)) && \text{(Ind. Hyp.)}\\
&= \min \mathsf{set}(v)\,. && (\mathsf{then}(v) = \mathsf{else}(v))
\end{aligned}$$

Otherwise, v is not redundant. Let e denote the node returned by $\mathit{MinWin}(i+1, \mathsf{else}(v))$ and set $\mathcal{A} := \min \mathsf{set}(\mathsf{else}(v))$. By our induction hypothesis we have $\mathsf{set}(e) = \mathcal{A}$ and we can conclude with:

$$\begin{aligned}
&\mathsf{set}(\mathit{MinWin}(i, v))\\
&\quad= \mathsf{set}(\mathsf{ite}(i, \mathsf{minus}(\mathit{MinWin}(i+1, \mathsf{then}(v)), e), e)) && \text{(Alg.)}\\
&\quad= \{S + i \mid S \in \mathsf{set}(\mathsf{minus}(\mathit{MinWin}(i+1, \mathsf{then}(v), e)))\} \cup \mathsf{set}(e) && \text{(Def. } \mathsf{set}\text{)}\\
&\quad= \{S + i \mid S \in \mathsf{set}(\mathit{MinWin}(i+1, \mathsf{then}(v))) \setminus \mathsf{set}(e)\} \cup \mathsf{set}(e) && \text{(Alg. } \mathsf{minus}\text{)}\\
&\quad= \{S + i \mid S \in \min \mathsf{set}(\mathsf{then}(v)) \setminus \mathcal{A}\} \cup \mathcal{A} && \text{(Ind. Hyp.)}\\
&\quad= \min \mathsf{set}(v)\,.
\end{aligned}$$

The last equality is due to Lemma 5.1 on page 54. □

The idea to obtain the QOBDD for $\mathcal{L}_{\max}$ from the QOBDD for $\mathcal{L}$ is similar. By recognizing that $\mathcal{L} = \mathsf{set}(\overline{r})$, however, we can also compute the QOBDD for $\mathcal{L}_{\max}$ directly from r. This fact has been incorporated in Algorithm 6 by twisting the sinks.

Algorithm 6 *MaxLosing*(i, v)

Require: v is a QOBDD node with label i and $\mathsf{set}(v)$ is an up-set.
Ensure: Returns node v' such that $\max \mathsf{set}(\overline{v}) = \mathsf{set}(v')$.

```
if $v = \mathbb{I}$ then return $\mathbb{O}$
else if $v = \mathbb{O}$ then return $\mathbb{I}$
else if $\mathsf{then}(v) = \mathsf{else}(v)$ then return $\mathsf{ite}(i, \mathit{MaxLosing}(i+1, \mathsf{then}(v)), \mathbb{O}_{i+1})$
else return $\mathsf{ite}(i, t, \mathsf{minus}(\mathit{MaxLosing}(i+1, \mathsf{else}(v)), t))$
    where $t = \mathit{MaxLosing}(i+1, \mathsf{then}(v))$
```

Because the structure of the algorithm is very similar to that of *MinWin*, a proof of the following statement is omitted.

Proposition 6.30. *It holds* $\max \mathsf{set}(\overline{r}) = \mathsf{set}(\mathit{MaxLosing}(1, r))$. □

The running time of the algorithms *MinWin* and *MaxLosing* suffers from the recursive application of the algorithm minus in line 3. Fortunately, for the important class of directed simple games we can prove, that slightly modified versions of the algorithms have expected running time $\mathcal{O}(\mathsf{size}(r))$. Most real world simple games are complete and therefore have an ordering of the players, such that the simple game is directed.

In the following we assume, that $(N, \mathcal{W})$ is directed. Let $v \in \mathsf{V}(r)$ be an inner node. Because $\mathsf{set}(r)$ is an up-set, also v represents an up-set. Additionally, because $(N, \mathsf{set}(r))$ is a directed simple game, also v represents a directed simple game with players $\{\mathsf{var}(v), \dots, n\}$ and winning coalitions $\mathsf{set}(v)$. By Lemma 5.2 on page 54 it follows:

$$\min \mathsf{set}(\mathsf{then}(v)) \setminus \min \mathsf{set}(\mathsf{else}(v)) = \min \mathsf{set}(\mathsf{then}(v)) \,.$$

Therefore in Algorithm 5, the expression

$$\mathsf{minus}(\mathit{MinWin}(i+1, \mathsf{then}(v)), \mathit{MinWin}(i+1, \mathsf{else}(v)))$$

always evaluates to $\mathit{MinWin}(i+1, \mathsf{then}(v))$ and thus, it can be substituted by the latter. The resulting algorithm has running time linear in the size of the input QOBDD. Similar arguments can be used to show that *MaxLosing* can be altered to run in linear time for directed simple games. The structural changes between the QOBDDs for $\mathcal{W}$, $\mathcal{W}_{\min}$ and $\mathcal{L}_{\max}$ are illustrated in Figure 6.7.

Theorem 6.31. *For a directed simple game $(N, \mathcal{W})$ represented by a* QOBDD *with root r, the* QOBDD*s for $\mathcal{W}_{\min}$ and $\mathcal{L}_{\max}$, respectively, can be computed in time $\mathcal{O}(\mathsf{size}(r))$ and either* QOBDD *has size at most $\mathsf{size}(r) + n$.*

Proof. The running time is clear from the previous discussion. The upper bound for the size follows because at most n inner nodes are created for $\mathbb{O}_1, \dots, \mathbb{O}_n$. □

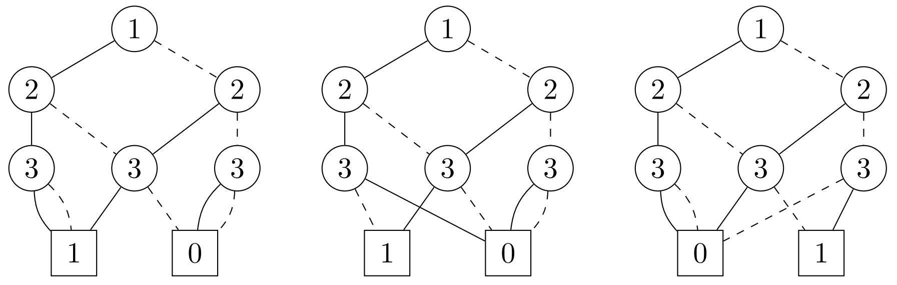


Figure 6.7.: Difference between the QOBDDs for $\mathcal{W}$ (left), $\mathcal{W}_{\min}$ (center) and $\mathcal{L}_{\max}$ (right) for the (directed) weighted voting game $[2; 1, 1, 1]$.

6.6. Shift-minimal Winning and Shift-maximal Losing Coalitions

In this section we show how to obtain the QOBDD for the shift-minimal winning coalitions $\mathcal{W}_{\text{shift}}$ from the QOBDD for minimal winning coalitions $\mathcal{W}_{\min}$ and, analogously, how to obtain the QOBDD for the shift-maximal losing coalitions $\mathcal{L}_{\text{shift}}$ from the QOBDD for the maximal losing coalitions $\mathcal{L}_{\max}$. These coalitions and their models play an important role in the characterization of complete simple games (Carreras and Freixas 1996), they can be used to find weighted representations of simple games using (integer) linear programming (Freixas, Molinero, Olsen, and Serna 2012) and, recently, they have been used to define a new power index (Alonso-Meijide and Freixas 2010).

Throughout this section let $(N, \mathcal{W})$ be a simple game with players $N = \{1, \ldots, n\}$ represented by the QOBDD with root r and let the set of the minimal winning coalitions $\mathcal{W}_{\min}$ be represented by a QOBDD with root $r_{\min}$.

In the following, we will focus on the shift-minimal winning coalitions. The case of $\mathcal{L}_{\text{shift}}$ can be treated similarly. We develop the algorithm successively, starting at the formal definition of a shift-minimal winning coalition in Def. 2.6 on page 14. A coalition $S \in 2^N$ is in $\mathcal{W}_{\text{shift}}$ if and only if $S \in \mathcal{W}_{\min}$ and

$$\forall i, j \in N : (j \prec_I i \wedge i \in S \wedge j \notin S \implies (S - i) + j \notin \mathcal{W})\,. \tag{6.26}$$

By negation we obtain $S \in 2^N$ is not in $\mathcal{W}_{\text{shift}}$ if $S \notin \mathcal{W}_{\min}$ or (6.26) is false for S. The negation of (6.26) is:

$$\exists i, j \in N : j \prec_I i \wedge i \in S \wedge j \notin S \wedge (S - i) + j \in \mathcal{W}\,. \tag{6.27}$$

$\Phi_{i,j}(S)$ We define $\Phi_{i,j}(S)$ as a shorthand for the formula $i \in S \wedge j \notin S \wedge (S - i) + j \in \mathcal{W}$, so that (6.27) can be rewritten to

$$\exists i, j \in N : j \prec_I i \wedge \Phi_{i,j}(S)\,. \tag{6.28}$$

It suffices to consider a subset of all pairs i, j of players in N to decide if (6.28) is true Λ_i for a given S. To this end, for player $i \in N$ we define the set Λ_i as the set of players

$j \in N$ such that $j \prec_I i$ and there is no player $k \in N$ with $j \prec_I k \prec_I i$. Intuitively, if $j \in \Lambda_i$ then j is a *direct predecessor* of i w.r.t. $\prec_I$. We use the following equivalence where $N_1, \ldots, N_t$ are the t equivalence classes of $\approx_I$ (types):

Lemma 6.32. *For $S \in 2^N$ it holds:*

$$\exists i, j \in N : j \prec_I i \land \Phi_{i,j}(S) \iff \exists k \in \{1, \ldots, t\} : \exists i \in N_k : \exists j \in \Lambda_i : \Phi_{i,j}(S)\,.$$

Proof. The direction "$\Longleftarrow$" is obvious, because $j \in \Lambda_i$ implies $j \prec_I i$. For the remaining direction, assume there are players $i, j \in N$ such that $j \prec_I i$ and $\Phi_{i,j}(S)$ is satisfied. Because every player belongs to some type, we only have to show that there are $i, j \in N$ with $j \in \Lambda_i$ and $\Phi_{i,j}(S)$. To this end, for players $i, j \in N$ with $j \prec_I i$ we define the *distance between* $i, j \in N$, denoted by $d(i, j)$, as the minimum number of edges between i and j in the directed graph $(N, \prec_I)$. It is $j \in \Lambda_j$ if and only if $d(i, j) = 1$.

Assume to the contrary, that there is no pair i, j such that $j \in \Lambda_i$ and $\Phi_{i,j}(S)$. Let i, j be a pair of players with $\Phi_{i,j}(S)$ and with minimum distance $d(i, j)$. Because $j \notin \Lambda_i$ it follows $d(i, j) > 1$ and hence, there is a player $p \in N$, such that $j \prec_I p \prec_I i$. There are two cases. First, assume $p \in S$. Because $\Phi_{i,j}(S)$ is satisfied, we have $(S - i) + j \in \mathcal{W}$ and together with $p \prec_I i$ this implies $(S - p) + j \in \mathcal{W}$ and $\Phi_{i,p}(S)$. Since $d(p, j) < d(i, j)$ this contradicts the minimality in the choice of i. Second, assume $p \notin S$. Because $\Phi_{i,j}(S)$ it holds $(S - i) + j \in \mathcal{W}$ and together with $j \prec_I p$ this implies $(S - i) + p \in \mathcal{W}$ and $\Phi_{p,j}(S)$. Since $d(i, p) < d(i, j)$ this contradicts the minimality in the choice of j. Consequently, there are players $i, j \in N$ with $\Phi_{i,j}(S)$ and $d(i, j) = 1$ and therefore, $j \in \Lambda_i$. $\square$

By using disjunctions instead of existential quantifications in the right-hand side of Lemma 6.32 we can rewrite (6.28) to:

$$\bigvee_{k=1}^{t} \bigvee_{i \in N_k} \bigvee_{j \in \Lambda_i} i \in S \land j \notin S \land (S - i) + j \in \mathcal{W}\,. \tag{6.29}$$

The manipulators remove and add from Section 6.1 can now be used for the transition to QOBDDs and sets. By using $\mathsf{set}(r) = \mathcal{W}$ and the manipulator remove from Lemma 6.8 we have that (6.29) is equivalent to:

$$\begin{aligned}
&\bigvee_{k=1}^{t} \bigvee_{i \in N_k} \bigvee_{j \in \Lambda_i} i \in S \land j \notin S \land (S - i) + j \in \mathsf{set}(r) \\
\iff &\bigvee_{k=1}^{t} \bigvee_{i \in N_k} \bigvee_{j \in \Lambda_i} i \in S \land S - i \in \mathsf{set}_{\mathsf{remove}_j}(r) \\
\iff &\bigvee_{k=1}^{t} \bigvee_{i \in N_k} i \in S \land S - i \in \bigcup_{j \in \Lambda_i} \mathsf{set}_{\mathsf{remove}_j}(r)\,.
\end{aligned} \tag{6.30}$$

For player $i \in N$ the QOBDD for

$$\bigcup_{j \in \Lambda_i} \mathsf{set}_{\mathsf{remove}_j}(r) \tag{6.31}$$

can be computed successively using the algorithm or as can be seen in Algorithm 7 below. The result QOBDD node is denoted by x. By using this and by Lemma 6.8 for the manipulator add, we rewrite (6.30) to:

$$\bigvee_{k=1}^{t} \bigvee_{i \in N_k} i \in S \wedge S - i \in \mathsf{set}(x) \iff \bigvee_{k=1}^{t} \bigvee_{i \in N_k} S \in \mathsf{set}_{\mathsf{add}_i}(x)$$

Finally, we have that a coalition $S \in 2^N$ is shift-minimal winning if and only if

$$S \in \mathsf{set}(r_{\min}) \wedge \neg \bigvee_{k=1}^{t} \bigvee_{i \in N_k} S \in \mathsf{set}_{\mathsf{add}_i}(x)\,.$$

By using set operations instead of logical operators, this is equivalent to

$$S \in \mathsf{set}(r_{\min}) \setminus \bigcup_{k=1}^{t} \bigcup_{i \in N_k} \mathsf{set}_{\mathsf{add}_i}(x)\,.$$

From this it is rather easy now to obtain the algorithmic formulation in Algorithm 7.

Algorithm 7 *ShiftMinWin*$(r, r_{\min})$

Require: r and $r_{\min}$ such that $\mathsf{set}(r) = \mathcal{W}$ and $\mathsf{set}(r_{\min}) = \mathcal{W}_{\min}$, respectively.
Ensure: Returns QOBDD node v such that $\mathsf{set}(v) = \mathcal{W}_{\text{shift}}$.

```
1: $y \leftarrow \mathbb{O}_1$
2: for $k = 1, \dots, t$ do
3:     $x \leftarrow \mathbb{O}_1$
4:     let $p$ be any player in $N_k$
5:     for $j \in \Lambda_p$ do
6:         $x \leftarrow \mathsf{or}(x, \mathsf{id}, r, \mathsf{remove}_j)$
7:     for $i \in N_k$ do
8:         $y \leftarrow \mathsf{or}(y, \mathsf{id}, x, \mathsf{add}_i)$
9: return $\mathsf{minus}(r_{\min}, y)$
```

We will omit a theoretical analysis of the running time for the algorithm *ShiftMinWin*, because the upper bound that we would obtain by a naive analysis, would be rather distracting from what we experience in practice. We therefore refer to the experiments later. We briefly discuss the number of QOBDD operations instead.

Because the types $N_1, \dots, N_t$ are disjoint in line 7 of Algorithm 7, each player appears once as i. In line 5, a player may appear multiple times or not at all as j. We therefore can bound the number of QOBDD operations from above by

$$1 + n + \sum_{p=1}^{n} \frac{|\Lambda_p|}{|N_{\tau(p)}|} \tag{6.32}$$

where $\tau : N \to \{1,\dots,t\}$ with $\tau(p) = k$ if and only if $p \in N_k$. The algorithm depends mainly on the relation $\prec_I$. Therefore, if many pairs of players are incomparable, then the computation becomes quicker. If all players are incomparable, that is $\prec_I = \emptyset$, then the sum in (6.32) evaluates to 0 and the minus operation in line 9 has expected running time $\mathcal{O}(\mathsf{size}(r_{\min}))$. Many real world simple games are complete, though. If the simple game is complete and $t = n$, then the sum in (6.32) evaluates to $n - 1$.

Algorithm 8 *ShiftMaxLosing*$(s, s_{\max})$

Require: s and $s_{\max}$ such that $\mathsf{set}(s) = \mathcal{L}$ and $\mathsf{set}(s_{\max}) = \mathcal{L}_{\max}$, respectively.
Ensure: Returns QOBDD node v such that $\mathsf{set}(v) = \mathcal{L}_{\text{shift}}$.

```
$y \leftarrow \mathbb{O}_1$
for $k = 1,\dots,t$ do
    $x \leftarrow \mathbb{O}_1$
    let $p$ be any player in $N_k$
    for $j \in \Lambda_p$ do
        $x \leftarrow \mathsf{or}(x, \mathsf{id}, s, \mathsf{add}_j)$
    for $i \in N_k$ do
        $y \leftarrow \mathsf{or}(y, \mathsf{id}, x, \mathsf{remove}_i)$
return $\mathsf{minus}(s_{\max}, y)$
```

Similarly to the computation of the QOBDD for $\mathcal{W}_{\text{shift}}$, one can use Algorithm 8 to compute the QOBDD for the shift-maximal losing coalitions $\mathcal{L}_{\text{shift}}$.

6.7. Power Indices

Measuring the "power" of a player in a simple game is one of the most important topics in practice. However, there is no single notion of power. Various notions have evolved over time. In this thesis we consider so-called *a priori voting power*; see Felsenthal and Machover (2004) and Taylor (1995) for an introduction. In contrast to *actual voting power*, a priori voting power does *only* consider the structure of a simple game. In other words, the players neither have preferences, intentions nor predispositions. *Power indices* are one of the most common approach to a priori power. Roughly speaking, a power index maps each player to a numerical value which corresponds to its power. There is vast literature on this topic from different perspectives. Some of the most commonly used power indices are those of Shapley and Shubik (1954), Banzhaf (1965), Deegan and Packel (1978) and Holler (1982). Further power indices have been developed over time, such as the power index of Johnston (1978) and quite recently, the *shift power index* (Alonso-Meijide and Freixas 2010). Some authors consider additional restrictions on the game, like so-called a priori unions of players. These lead to modified versions of existing power indices; see for instance Alonso-Meijide, Bilbao, Casas-Méndez, and Fernández (2009) and Alonso-Meijide and Fiestras-Janeiro (2002) for an introduction.

In the context of politics, power indices have been used, e.g., to assess the Council of the European Union[3] (Algaba, Bilbao, and Fernández 2007; Kirsch and Langner 2011) and the International Monetary Fund (Leech 1998; Leech 2002). Beside that, power indices have also been used in completely different contexts. Kirstein (2009) studies the distribution of power in the supervisory board of the Porsche Automobile Holding SE after the takeover of the Volkswagen AG in 2008. Bachrach, Rosenschein, and Porat (2008) apply power indices to the so-called network reliability problem to decide which servers in a computer network should be maintained first, and Lucchetti and Radrizzani (2010) use power indices to rank genes, which are potentially responsible for genetic diseases in so-called microarray games with a large number of players (resp. genes).

The most common approach to compute power indices for weighted voting games is to use dynamic programming by so-called generating functions. For an overview see Matsui and Matsui (2000) and Leech (2002). For a weighted voting game $[Q; w_1, \ldots, w_n]$, the problem of computing most power indices is NP-complete, e.g., the Banzhaf and Shapley-Shubik power indices (Matsui and Matsui 2001). Klinz and Woeginger (2005) have presented algorithms, that can compute the Banzhaf and Shapley-Shubik power index for a single player in time $\mathcal{O}(n^2 2^{n/2})$ and $\mathcal{O}(n 2^{n/2})$, respectively, while Uno (2003) has presented pseudo-polynomial time algorithms to compute the Banzhaf and Shapley-Shubik power index for *all* players in time $\mathcal{O}(nQ)$ and $\mathcal{O}(n^2 Q)$, respectively. Approximation algorithms exist for some classes of WVGs. See again Leech (2002) for an overview.

Even though most algorithms work very well for weighted voting games, the case of vector-weighted voting games or simple games in general has been studied less extensively. Algaba et al. (2003) use generating functions to compute the Banzhaf and Shapley-Shubik power index for vector-weighted representations with an application to the Council of the European Union as defined in the Treaty of Nice. In comparison, by means of QOBDDs, we are able to compute power indices for any simple game.

In this section we show how to compute five power indices for the players of a simple game. Namely, we discuss the Banzhaf, Shapley-Shubik, Deegan-Packel, Holler-Packel and the shift power indices mentioned earlier.

Throughout this section, the simple game $(N, \mathcal{W})$ with players $N = \{1, \ldots, n\}$ is represented by a QOBDD with root r, the set $\mathcal{W}_{\min}$ is represented by the QOBDD with root $r_{\min}$ and the set $\mathcal{W}_{\text{shift}}$ is represented by the QOBDD with root r_{shift}. We start by defining the power indices. In the remainder of this section by an *index* we mean a power index.

$\mathsf{bz}(i)$

The *absolute Banzhaf index* $\mathsf{bz}(i)$ for player $i \in N$ is the fraction of coalitions that contain i and for which player i is critical:

$$\mathsf{bz}(i) := \frac{|\{S \in \mathcal{W} \mid i \in S \wedge S - i \notin \mathcal{W}\}|}{2^{n-1}}\,. \tag{6.33}$$

If i is not a member of any minimal winning coalition, then $\mathsf{bz}(i) = 0$. Such a player is called a *dummy player*. The Banzhaf index can be normalized to sum up to 1. This version is called the *normalized Banzhaf index*.

Let π be a permutation of the players N. For a position $p \in N$ the player $\pi(p)$ is called *pivotal for* π if $S := \{\pi(1), \ldots, \pi(p)\} \in \mathcal{W}$, but $S - \pi(p) \notin \mathcal{W}$. If i is critical for

[3]For both, the Treaty of Nice and the Treaty of Lisbon

a coalition $S \in \mathcal{W}$, then there are $(|S|-1)! \cdot (n-|S|)!$ permutations for which player i is pivotal. The *Shapley-Shubik index* $\mathsf{ss}(i)$ of player i is the fraction of permutations for which player i is pivotal: $\mathsf{ss}(i)$

$$\mathsf{ss}(i) := \frac{1}{n!} \sum_{\substack{S \in \mathcal{W},\\ S-i \notin \mathcal{W}}} (|S|-1)! \cdot (n-|S|)! \,. \tag{6.34}$$

The Holler-Packel index[4] and the Deegan-Packel index both use the minimal winning coalitions $\mathcal{W}_{\min}$. Both indices have similar structure. The *normalized Holler-Packel index* $\mathsf{hp}(i)$ for player $i \in N$ is the fraction of minimal winning coalitions containing player i: $\mathsf{hp}(i)$

$$\mathsf{hp}(i) := \frac{|\{S \in \mathcal{W}_{\min} \mid i \in S\}|}{|\mathcal{W}_{\min}|} \,. \tag{6.35}$$

In comparison, the *Deegan-Packel index* $\mathsf{dp}(i)$ for player i sums up player i's shares of the minimal winning coalitions containing i. It is defined by: $\mathsf{dp}(i)$

$$\mathsf{dp}(i) := \frac{1}{|\mathcal{W}_{\min}|} \sum_{\substack{S \in \mathcal{W}_{\min},\\ i \in S}} \frac{1}{|S|} \,. \tag{6.36}$$

The *shift power index* $\mathsf{sh}(i)$ is similar to the Holler-Packel index, but it uses the shift-minimal winning coalitions instead of the minimal winning coalitions: $\mathsf{sh}(i)$

$$\mathsf{sh}(i) := \frac{|\{S \in \mathcal{W}_{\text{shift}} \mid i \in S\}|}{|\mathcal{W}_{\text{shift}}|} \,. \tag{6.37}$$

To compute these indices, we only have to apply our results from Section 6.2. The main work has already been done there. Let $i \in N$ be a player. In case of the Banzhaf index we use (6.33) and the definition of the swings of player i in Def. 6.6 and get:

$$\mathsf{bz}(i) \cdot 2^{n-1} = |\{S \in \mathcal{W} \mid i \in S \wedge S-i \notin \mathcal{W}\}| = |\mathsf{swings}_{\mathcal{W}}(i)| \,.$$

The Shapley-Shubik index does also use the swings of player i, but this time, the size of the coalitions is taken into account. With $\mathcal{S}_i := \mathsf{swings}_{\mathcal{W}}(i)$ and $\|\cdot\|$ as in Def. 6.4 we rewrite (6.34) by:

$$\begin{aligned}
\mathsf{ss}(i) \cdot n! &= \sum_{T \in \mathcal{S}_i} (|T|-1)!(n-|T|)! && \text{(Eq. 6.34)}\\
&= \sum_{k=0}^{n} \sum_{\substack{T \in \mathcal{S}_i,\\ |T|=k}} (|T|-1)!(n-|T|)!\\
&= \sum_{k=1}^{\text{maxsize}(\mathcal{S}_i)} (k-1)!(n-k)! \cdot \|\mathcal{S}_i\|_k
\end{aligned}$$

[4]Also known as *public good index*.

where $\|\mathcal{S}_i\|_k$ is the number of swings of size k. We can safely start at $k = 1$, because a swing S for player i always fulfills $i \in S$. For the Holler-Packel index we have

$$\mathsf{hp}(i) \cdot |\mathcal{W}_{\min}| \overset{(6.35)}{=} |\{S \in \mathcal{W}_{\min} \mid i \in S\}| = \mathsf{chow}_{\mathcal{W}}(i)$$

while, similar to $\mathsf{ss}(i)$, for the Deegan-Packel index we again have to use $\|\cdot\|$ from Def. 6.4. With $\mathcal{C}_i := \{S \in \mathcal{W}_{\min} \mid i \in S\}$ we get:

$$\mathsf{dp}(i) \cdot |\mathcal{W}_{\min}| \overset{(6.36)}{=} \sum_{\substack{S \in \mathcal{W}_{\min},\\ i \in S}} \frac{1}{|S|} = \sum_{k=1}^{n} \sum_{\substack{T \in \mathcal{C}_i,\\ |T|=k}} \frac{1}{k} = \sum_{k=1}^{\text{maxsize}(\mathcal{C}_i)} \frac{\|\mathcal{C}_i\|_k}{k} .$$

Finally, the shift power index is structurally similar to the Holler-Packel index again. We obtain:

$$\mathsf{sh}(i) \cdot |\mathcal{W}_{\text{shift}}| \overset{(6.37)}{=} |\{S \in \mathcal{W}_{\text{shift}} \mid i \in S\}| = \mathsf{chow}_{\mathcal{W}_{\text{shift}}}(i) .$$

The main result in this section is the following.

Theorem 6.33. *The (absolute) Banzhaf, (normalized) Holler-Packel and shift power indices of* all *players can be computed with $\mathcal{O}(\mathit{size}(r))$, $\mathcal{O}(\mathit{size}(r_{\min}))$ and $\mathcal{O}(\mathit{size}(r_{\text{shift}}))$ arithmetic operations, respectively, and the Shapley-Shubik and Deegan-Packel indices of* all *players can be computed with $\mathcal{O}(n \cdot \mathit{size}(r))$ and $\mathcal{O}(n \cdot \mathit{size}(r_{\min}))$ arithmetic operations, respectively.*

Proof. We explain the idea for the Deegan-Packel index. It can similarly be applied to the other indices. The coalitions of interest are in $\mathcal{W}_{\min}$, so our QOBDD has root $r_{\min}$. First, we compute the values $\|\mathsf{paths}(r_{\min}, v)\|$ and $\|\mathsf{set}(v)\|$ for each $v \in \mathsf{V}(r_{\min}) \cup \{\mathbb{O}, \mathbb{I}\}$ using $\mathcal{O}(n \cdot \mathsf{size}(r_{\min}))$ arithmetic operations as shown in Theorem 6.19. The cardinality of $\mathcal{W}_{\min}$ can be calculated by

$$|\mathcal{W}_{\min}| = \sum_{k=0}^{n} \|\mathsf{set}(r_{\min})\|_k .$$

For player $i \in N$, the vector $\|\{S \in \mathcal{W}_{\min} \mid i \in S\}\|$ can be computed using $\mathcal{O}(n|\mathsf{V}_i(r_{\min})|)$ arithmetic operations. Therefore, all the values can be computed with $\mathcal{O}(n \cdot \mathsf{size}(r_{\min}))$ arithmetic operations. □

It should be mentioned again, that the majority of work has already been done in Section 6.2. Here, we have only applied the results from that section and therefore, we have added a convenient level of abstraction between the concept of power indices and the computational problem which is mainly a counting problem.

Because a QOBDD for a WVG has bounded size, for this class of simple games we can state the following result using Theorem 5.5 and Theorem 5.11. The running time does not include the time to build the QOBDD. This would add an additional factor of n and $\log Q$, respectively, to the running time.

Corollary 6.34. *If the* QOBDD *with root r represents a* WVG *with n players and quota $Q \geq 1$ then we can compute the Banzhaf power index for* all *players in time*

$$\mathcal{O}(\min\{2^{n/2}, \max\{n - \log Q, 1\}Q\})$$

and we can compute the Shapley-Shubik power index for all *players in time*

$$\mathcal{O}(n \cdot \min\{2^{n/2}, \max\{n - \log Q, 1\}Q\})$$

under the assumption of constant time arithmetic. □

6.8. Models of Coalitions

In this section we discuss so-called models of sets and present algorithms to count and enumerate them. Models are vectors that abstract from individual elements to equivalence classes of elements. In the context of simple games the equivalence classes are the types $N_1, \ldots, N_t$, the elements are players and the sets are coalitions. The idea is the following. If there is a winning coalition in a simple game with x players of type k then for any other combination of x players of type k there is another winning coalition with these players that does not differ in the remaining players. However, the information about the number of players of each type would suffice to generate all the possible combinations. As an example, for the WVG $[2; 1, 1, 1]$ it is sufficient to know, that a coalition wins, if it contains 2 or 3 players instead of enumerating all the winning coalitions with 2 and 3 players, respectively.

For some problems, it is sufficient to consider a set of models instead of the corresponding set of coalitions. For instance, as we will see in Section 7, the models of the shift-minimal winning and shift-maximal losing coalitions can be used to decide if a simple game has a weighted representation.

To use models instead of coalitions is beneficial only if the number of players n is bigger than the number of types t. If $t = n$ then each model is in a one-to-one relationship to a coalition. Fortunately, many real world simple games have several equally desirable players. For instance, the US Electoral College (2010-2012) has 51 players but only 19 types. The Council of the European Union as defined in the Treaty of Nice has 27 players and 10 types. The US Federal Legislative System has 537 players but just 4 types. In the latter example there are more than $6.77 \cdot 10^{158}$ minimal winning coalitions but there are just three models[5] for them:

$$(50, 218, 1, 1) \qquad (51, 218, 1, 0) \qquad (67, 290, 0, 0)\,.$$

Hence, the number of models can be much smaller than the number of coalitions, so that using models instead of coalitions can make the difference between tractability and intractability in practice.

[5]Order of types: Members of the senate, member of the house of representatives, president, vice-president.

In this section we present algorithms to count and enumerate the models of an arbitrary subset $\mathcal{A}$ of 2^N. In the context of a simple game $(N, \mathcal{W})$, $\mathcal{A}$ usually is one of the sets

$$\mathcal{W}, \mathcal{W}_{\min}, \mathcal{W}_{\text{shift}}, \mathcal{L}, \mathcal{L}_{\max}, \mathcal{L}_{\text{shift}}\,. \tag{6.38}$$

The notion of a model is not limited to simple games. Unfortunately, the desirability relation on individuals has been stated in this context and it would be confusing to apply the notion of desirability to non-players. We, however, remain committed to subsets of 2^N where $N = \{1, \dots, n\}$ for $n \in \mathbb{N}$ but the elements $1, \dots, n$ are not called players anymore.

$i \approx j$

Definition 6.8. For $\mathcal{A} \subseteq 2^N$ we say two elements $i, j \in N$ are *symmetric (in $\mathcal{A}$)* (denoted by $i \approx j$) if

$$\forall S \subseteq N \setminus \{i, j\} : (S + i \in \mathcal{A} \iff S + j \in \mathcal{A})\,. \qquad \square$$

The relation $\approx$ is an equivalence relation. For a simple game $(N, \mathcal{W})$ it can easily be seen, that being symmetric and being equally desirable is the same. The following definitions are also consistent with those from Section 2 for simple games. The equivalence classes of $\approx$ are denoted by $N_1, \dots, N_t$ where t is their number.

It is not necessary to use exactly the equivalence classes of $\approx$ to talk about models. It is perfectly fine to use any refinement of the partition $N_1, \dots, N_t$, for instance, $\{1\}, \dots, \{n\}$.

Definition 6.9. Let the partition $P_1, \dots, P_d$ of N be a refinement of the equivalence classes of $\approx$ and let S be a set in $\mathcal{A}$. The vector $\vec{m} = (m_1, \dots, m_d) \in \mathbb{N}_0^d$ is called a *model of S (w.r.t. $P_1, \dots, P_d$)*, if

$$\forall i \in \{1, \dots, d\} : m_i = |P_i \cap S|\,.$$

$\mathsf{models}_{P_1,\dots,P_d}(\mathcal{A})$
$\mathsf{models}(\mathcal{A})$

The vector $\vec{m}$ is a *model of $\mathcal{A}$ (w.r.t. $P_1, \dots, P_d$)*, if $\vec{m}$ is a model of some element in $\mathcal{A}$ w.r.t. $P_1, \dots, P_d$. The set of models $\mathcal{A}$ w.r.t. $P_1, \dots, P_d$ is denoted by $\mathsf{models}_{P_1,\dots,P_d}(\mathcal{A})$. We write $\mathsf{models}(\mathcal{A})$ if the partition of N is clear from the context. $\square$

Notice that for the case of a simple game $(N, \mathcal{W})$ and in which $\mathcal{A}$ is one of the sets in (6.38), the types of the players in $(N, \mathcal{W})$ are a refinement of the equivalence classes of the relation $\approx$ for $\mathcal{A}$.

In the following, it is crucial that variables for symmetric elements form consecutive levels in the QOBDD. In contrast, the order of the partitions is irrelevant.

Definition 6.10. A partition $P_1, \dots, P_d$ of N is said to be *consecutive*, if for each $k \in \{1, \dots, d\}$ it holds

$$\forall x \in N : (\min P_k \leq x \leq \max P_k \implies x \in P_k)\,. \qquad \square$$

$P_1, \dots, P_d$
$p_1, \dots, p_d$

In the remainder of this section, we assume that $\mathcal{A} \subseteq 2^N$ is represented by the QOBDD with root r, and the partition $P_1, \dots, P_d$ of N is consecutive and a refinement of $N/{\approx}$. For $k \in \{1, \dots, d\}$ we define p_k as $|P_k|$. As a convenience and w.l.o.g. we suppose:

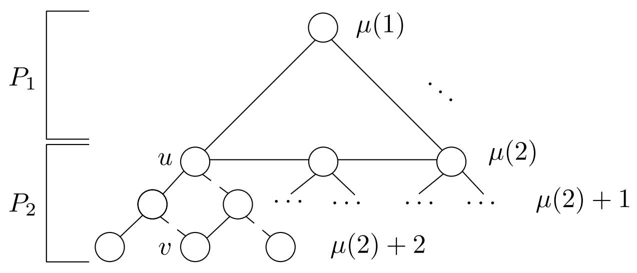


Figure 6.8.: QOBDD structure for a consecutive refinement of $N/{\approx}$.

$$\forall k \in \{1,\dots,d-1\} : \max P_k \leq \min P_{k+1}\,.$$

Hence, we have $P_1 = \{1,\dots,p_1\}$, $P_2 = \{p_1+1,\dots,p_1+p_2\}$ and so on. The element with lowest index in P_k will play a prominent role in the following discussion. Therefore, we define a mapping μ by $\mu(k)$

$$\mu(k) = 1 + \sum_{j=1,\dots,k-1} p_j \tag{6.39}$$

for each $k \in \{1,\dots,d+1\}$. For instance, $\mu(1)=1$, $\mu(2)=1+p_1$, $\mu(3)=1+p_1+p_2$ and so on. Notice that $\mu(d+1)=n+1$ and thus, $\mathsf{V}(r)_{\mu(d+1)} = \{\mathbb{O},\mathbb{I}\}$.

Because the partition $P_1,\dots,P_d$ is consecutive, we have the following property. Consider the node u in Figure 6.8 with label $\mu(2)$. Starting at u the node v is reached by first taking the 1-edge of u and then taking the 0-edge, that is, $v = \mathsf{else}(\mathsf{then}(u))$. Now, because $P_1,\dots,P_d$ is a consecutive refinement of $N/{\approx}$, it also holds $v = \mathsf{then}(\mathsf{else}(v))$. In general, if on a path all variables are symmetric then it does not matter which path we take, as long as the number of 1-edges (and hence, 0-edges) coincides. As a consequence, on level $\mu(k+1)$ there are at most p_k+1 different nodes reachable from u. One for each number of 1-edges. We use this insight to define a successor function for the nodes in $\mathsf{V}_{\mu(k)}$ for any $k \in \{1,\dots,d\}$ as follows.

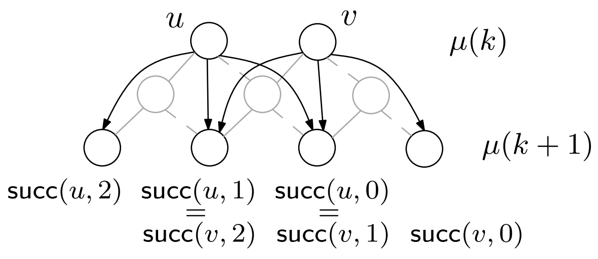
Definition 6.11. We say that succ is a *successor function (for r and $P_1,\dots,P_d$)* if for any $k \in \{1,\dots,d\}$, for any node $u \in \mathsf{V}_{\mu(k)}(r)$ and for any $j \in \{0,\dots,p_k\}$ it holds $\mathsf{succ}(u,j) = v$ if and only if $\mathsf{succ}(v,j)$

(i) $v \in \mathsf{V}_{\mu(k+1)}(r)$ and

(ii) starting at node u the node v is reached by taking j 1-edges and (p_k-j) 0-edges. □

The successor function is unique for a QOBDD and a partition of N, because the successors of each node are unique. It is the key to get rid of individual elements (resp. variables). Figure 6.9 illustrates the idea. We discuss the problem to obtain the successor function later. The inner nodes $\bigcup_{k=1,\dots,d} \mathsf{V}_{\mu(k)}(r)$ together with the terminal nodes $\mathbb{O},\mathbb{I}$ and the successor function can be considered as a *multivalued decision diagram.* See Wegener (2000), Section 9.1 for a formal definition of this type of decision diagrams.

Before we present a method to obtain the successor function, we briefly discuss algorithms to count the number of models and to enumerate the models of $\mathcal{A}$. Thanks to our successor function, this is nearly as easy as counting and enumerating the sets in $\mathcal{A}$.



u
v
$\mu(k)$
$\mu(k+1)$
$\mathsf{succ}(u,2)$
$\mathsf{succ}(u,1) = \mathsf{succ}(v,2)$
$\mathsf{succ}(u,0) = \mathsf{succ}(v,1)$
$\mathsf{succ}(v,0)$

Figure 6.9.: Example for the successor function succ and $u, v \in \mathsf{V}_{\mu(k)}$ with label $\mu(k)$.

Using the successor function succ, for $k \in \{1, \ldots, d+1\}$ and for a QOBDD node $v \in \mathsf{V}_{\mu(k)}(r)$ the number of models can be calculated recursively by the formula:

$$\mathit{CountModels}(k, v) = \begin{cases} 0 & \text{if } v = \mathbb{O} \\ 1 & \text{if } v = \mathbb{I} \\ \sum_{j=0}^{p_k} \mathit{CountModels}(k+1, \mathsf{succ}(v, j)) & \text{otherwise} \end{cases}$$

It is straightforward to turn this into a complete algorithm and it is also rather easy to verify its correctness. The latter is a direct consequence of the definition of the successor function succ. A computed table should be used to store temporary results. Disregarding the costs for insertion and lookup, as well as costs for arithmetic operations, the algorithm can be implemented with running time $\mathcal{O}(\mathsf{size}(r))$.

Algorithm 9 $\mathit{EnumModels}_f(k, v, (m_1, \ldots, m_{k-1}))$

```
Require: $1 \leq k \leq d+1$ and $v \in \mathsf{V}_{\mu(k)}$.
1: if $v = \mathbb{I}$ then {It holds $k = d+1$ in this case}
2:     call $f(m_1, \ldots, m_d)$
3: else if $v \neq \mathbb{O}_{\mathsf{var}(v)}$ then
4:     for $j = 0$ to $p_k$ do
5:         $\mathit{EnumModels}_f(k+1, \mathsf{succ}(v, j), (m_1, \ldots, m_{k-1}, j))$
```

By enumerating the models of $\mathcal{A}$, we denote the process of calling a user supplied unary function f for each model in $\mathsf{models}(\mathcal{A})$. As an example, if f puts its arguments into a list that is initially empty, then in the end, the list contains exactly the elements in $\mathsf{models}(\mathcal{A})$. We use the algorithm that is listed in Algorithm 9. Comments are enclosed in curly braces. The successor function succ and the numbers $p_1, \ldots, p_d$ are assumed to be in a global scope.

Proposition 6.35. *After the call* $\mathit{EnumModels}_f(1, r, ())$*, for each* $\vec{m} \in \mathbb{N}_0^d$ *the function* f *has been called with argument* $\vec{m}$ *exactly if* $\vec{m} \in \mathsf{models}(\mathcal{A})$*. The algorithm has running time* $\mathcal{O}(1 + n|\mathsf{models}(\mathcal{A})|)$*.*

Proof. The correctness follows straight from the definition of succ.

No computed table is used for the algorithm and therefore, nodes are visited multiple times. If $\mathsf{models}(\mathcal{A}) = \emptyset$ then $r = \mathbb{O}_1$ and therefore, constant time is needed. Otherwise,

Algorithm 10 *first*(v, k)

```
Require: $1 \leq k \leq d+1$ and $v \in \mathsf{V}_{\mu(k)}(r)$.
    if $v \notin \{\mathbb{I}, \mathbb{O}\}$ and $v$ has not been visited before then
        $\mathit{next}(v, k, p_k, \mathbf{true}, (v, 0))$
```

for each model, d inner nodes have to be visited starting at the root r. The overall costs for this are $\mathcal{O}(d|\mathsf{models}(\mathcal{A})|)$. Let $\vec{m} = (m_1, \ldots, m_d)$ be a model from $\mathsf{models}(\mathcal{A})$ and let $v_1, \ldots, v_d$ be the inner nodes that are visited for $\vec{m}$. Additionally, let v_{d+1} denote the sink $\mathbb{I}$. For the nodes there are indices $j_1, \ldots, j_d$ such that for $k \in \{1, \ldots, d\}$ it holds $v_{k+1} = \mathsf{succ}(v_k, j_k)$. The calls

$$\mathit{EnumModels}_f(1, v_1, ()), \ldots, \mathit{EnumModels}_f(d, v_d, (m_1, \ldots, m_{d-1}))$$

cause $p_1 + \cdots + p_d$ recursive calls. In the worst case, all except d of these recursive calls lead to the nodes $\mathbb{O}_{\mu(2)}, \ldots, \mathbb{O}_{\mu(d+1)}$ and therefore, have to be accounted to the model $\vec{m}$. Because $p_1 + \cdots + p_d = n$ they can be bounded by n for the model $\vec{m}$. □

Now we discuss how to obtain the successor function succ for the QOBDD with root r and the refinement $P_1, \ldots, P_d$ of $N/\approx$. While it is rather easy to state any algorithm for this problem and to prove its correctness, it is more challenging to find an economic algorithm with respect to time and space usage. Because of the usual trade-off between being easy to understand and begin economic in the sense just mentioned, we go without a formal proof and instead provide a thorough description of the algorithm's ideas. For reasons of simplicity, the algorithm is split into two parts which are listed in Algorithm 10 and 11, respectively. The variables $r, d, p_1, \ldots, p_d$ and the computed table are assumed to be in a global scope. The successor function succ is assumed to be a computed function in the global scope which can, for instance, be implemented by a hash table.

The short algorithm *first* handles the case in which a node v is the first node with a label in P_k, that is, $\mathsf{var}(v) = \mu(k)$ or v is a sink.

Proposition 6.36. *After first(v, k) was called for $k \in \{1, \ldots, d+1\}$ and $v \in \mathsf{V}_{\mu(k)}(r)$ we have that succ is the successor function for v and $P_k, \ldots, P_d$.* □

Intuitively, the successor function for r and $P_1, \ldots, P_d$ has been assigned partially. The correctness follows from the mutual recursion with the algorithm *next*.

Our workhorse *next* uses two main ideas: reuse of previously computed results and avoidance of unnecessary recursive calls. We discuss them in reverse order.

For $k \in \{1, \ldots, d\}$ let v be a node with label $\mu(k)$ as illustrated in Figure 6.10. Then for any $j \in \{0, \ldots, p_k\}$ the successor $\mathsf{succ}(v, j)$ is the node after taking a path with *any* j 1-edges and *any* $(p_k - j)$ 0-edges starting at v. For $j = 0$ there is a unique path. For $j = 1$ we can first take $(p_k - 1)$ 0-edges and then a 1-edge. In general we can first take $(p_k - j)$ 0-edges and then the remaining j 1-edges. This kind of recursion is realized by using the parameter takeZeroEdge in Algorithm 11, which is either true or false. Once false, it never becomes true again until a call to the algorithm *first*.

Algorithm 11 *next*$(v, \text{k}, \text{left}, \text{takeZeroEdge}, (o, s))$

```
if left = 0 then
    $\mathsf{succ}(o, s) \leftarrow v$
    $\mathit{first}(v, k+1)$
else if $\mathit{lookup}(T, v) \neq \bot$ then
    $(o', s') \leftarrow \mathit{lookup}(T, v)$
    if $o \neq o'$ then {All successors of $o'$ are known}
        for $j = 0$ to left do
            $\mathsf{succ}(o, s+j) \leftarrow \mathsf{succ}(o', s'+j)$
    else {takeZeroEdge is false}
        $\mathsf{succ}(o, s+\text{left}) \leftarrow \mathsf{succ}(o', s'+\text{left})$
else
    if takeZeroEdge then
        $\mathit{next}(\mathsf{else}(v), k, \text{left}-1, \textbf{true}, (o, s))$
    $\mathit{next}(\mathsf{then}(v), k, \text{left}-1, \textbf{false}, (o, s+1))$
    $\mathit{insert}(T, v, (o, s))$
```

The algorithm *next* assigns a successor node in line 2 when it reaches it, that is, if no node with a label in P_k is left (left $= 0$). When *next* is called by the algorithm *first* for the node v with label $\mu(k)$, then p_k edges have to be traversed (left is initially p_k) until we arrive at a successor of v. We denote this node v by o (for *origin*). If the argument left, after some calls to *next*, is 0, then the label of the current node v is $\mu(k+1)$. At this point we have to set succ for o and the number of 1-edges we have seen on the path from o to the current node v. We keep track of this number in the argument s (for *seen*), which is increased if a 1-edge is taken in line 14. Because v is now the first node with a label in P_{k+1} we call *first* for v and $k+1$.

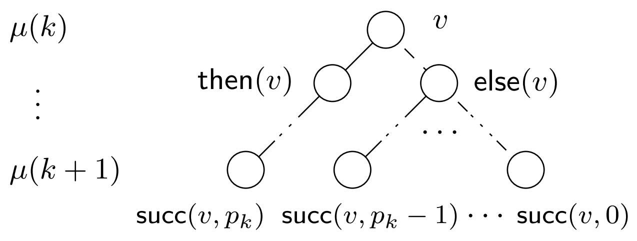


Figure 6.10.: Recursion for algorithm *next*.

For the computed table T in the algorithm *next* a hash table can be used. The idea for the use of the computed table in case $o \neq o'$ is illustrated in Figure 6.11. A dotted edge represents an arbitrary directed path between two nodes and the label next to it indicates the number of 1-edges on the path. As an example, the path from v to the node $\mathsf{succ}(o', s'+j)$ contains j 1-edges.

The idea now is, that when the current node v was visited before with origin o' after taking s' 1-edges and we visit v again with origin o after taking s 1-edges then for any

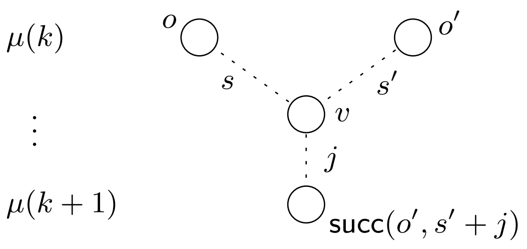


Figure 6.11.: Idea for reusing already computed results in the algorithm *next*.

number of remaining 1-edges $j \in \{0, \dots, \text{left}\}$ to a node on level $\mu(k+1)$ it holds

$$\mathsf{succ}(o, s+j) = \mathsf{succ}(o', s'+j)\,.$$

We therefore can reuse the corresponding successors as performed in lines 7 and 8 of *next*. The case in which we have visited v before, but the origins o and o' coincide is different. By the recursion of *next* all the values

$$\mathsf{succ}(o, 0), \dots, \mathsf{succ}(o, (s-1) + \text{left})$$

are known and this time, takeZeroEdge is false. We therefore have to set $\mathsf{succ}(o, s + \text{left})$ which is, similar to the case of $o \neq o'$, equal to $\mathsf{succ}(o, s' + \text{left})$.

For the costs to compute the successor function of our QOBDD with root r and the partition $P_1, \dots, P_d$ we remark:

Theorem 6.37. *Disregarding the costs for insertion and lookup operations of the computed table, the successor function* succ *for* r *and* $P_1, \dots, P_d$ *can be computed in expected time* $\mathcal{O}(\mathsf{size}(r) \cdot \max_{k=1,\dots,d} p_k)$ *by the initial call* $\mathit{first}(r, 1)$*.*

Proof. The additional factor is due to the assignments in line 7 and 8 in the algorithm *next*. The running time is expected due to the potential use of a hash table for the computed table T. □

Concrete running times are listed in the next section.

6.9. Experiments and Conclusions

In this chapter we have presented QOBDD-based algorithms for some fundamental problems on simple games. To this end, we have introduced two fundamental techniques. By the use of manipulators (Section 6.1) we are able to alter the set (resp. simple game), that is represented by a QOBDD, without modifying the QOBDD itself. This works by an abstraction of the successor functions then and else for the inner QOBDD nodes. We have adopted some of the foundations and algorithms for QOBDDs, like the binary synthesis. Afterwards, we have presented manipulators for some fundamental set-theoretic operations, that we have used in the subsequent sections to solve problems on simple games. Examples include the insertion of a player into every coalition, that is represented by a

QOBDD, and to filter all coalitions that contain a given player. In Section 6.2 we have presented an approach by which we can compute, for instance, the Chow parameters for *all* players and the swings for *every* player w.r.t. a set, that is represented by a QOBDD with root r, by only $\mathcal{O}(\mathsf{size}(r))$ arithmetic operations. Both, the Chow parameters and the swings for the players are necessary to compute some power indices.

In the following sections we have presented algorithms to solve various problems on simple games. Regarding the use of QOBDDs, we can distinguish three different procedures. For the desirability relation on individuals in Section 6.3 and for the QOBDDs for the shift-minimal winning and shift-maximal losing coalitions in Section 6.6, we have not used the structure of the QOBDD directly. Instead, we have just used the meaning of the manipulators, and the binary synthesis. In these cases, our argumentation has been rather formal. We have started with a specification and then, we have transformed it until we have been able to replace logic or set-theoretic parts with equivalent QOBDD constructs. A good example for this is the test to decide if $i \preceq_I j$ for players $i, j \in N$ in Theorem 6.22. On a similar high level of abstraction we have shown how some power indices can be computed by the approach in Section 6.2. Starting with the formula for a particular power index, we just have substituted relevant numbers like the swings for the players. In contrast, for the computation of the QOBDD for the minimal winning coalitions and for the algorithms for the models of coalitions in Section 6.8, we have fallen back on the QOBDD structure. These algorithms are comparably elusive and less elegant. However, by using these algorithm, other problems can be solved on a rather high level of abstraction again. Hence, the results in this chapter can be considered as a toolbox for simple games.

For the manipulator compls we have shown in Theorem 6.11, that if a QOBDD with root r represents a WVG with winning coalitions $\mathcal{W}$ and players N, then

$$\mathsf{reachable}(r, \mathsf{compls}, r, \mathsf{id}) \tag{6.40}$$

has expected running time $\mathcal{O}(\mathsf{size}(r))$. We have used this result in Section 6.4 to show that we can decide in time $\mathcal{O}(\mathsf{size}(r))$ whether $(N, \mathcal{W})$ is proper and strong, respectively. Because we know from Section 5.3, that the size of a QOBDD for a WVG $[Q; w]$ with n players is in

$$\mathcal{O}(\min\{2^{n/2}, \max\{n - \log Q, 1\}Q\}), \tag{6.41}$$

we immediately obtain an algorithm with expected running time (6.41) to decide if $(N, \mathcal{W})$ is proper and strong, respectively, if the QOBDD for $(N, \mathcal{W})$ has already been built. However, we have put considerable effort into the proof for the bound of the running time in (6.40). And because this is a rather simple case with only a single manipulator and a single QOBDD, we may expect that more complex cases are not getting much easier. Therefore, our approach is likely not the best choice to obtain complexity results for problems on simple games.

In general, the formal upper bounds for the running times of our algorithms are rather disappointing, except for a few special cases. One such exception is the linear time algorithm to obtain the QOBDD for $\mathcal{W}_{\min}$ given the QOBDD for $\mathcal{W}$. Apart from these exceptions, the formal upper bounds are far away from the running times that we encounter in practice for real world instances. For example, this is the case for the computation of the QOBDD for $\mathcal{W}_{\text{shift}}$ in Section 6.6. For that reason, we abstained from providing a formal upper bound in this case. However, if we have got a good formal upper bound for the running time, as it is the case for the power indices in Section 6.7, then we can use it together with the knowledge of classes of simple games with QOBDDs of bounded side. The upper bound for the size can be used to obtain a lower upper bound for the running time. As an example, we have already seen in Corollary 6.34 that we can compute the Banzhaf power index of *all* players in (deterministic) running time

$$\mathcal{O}(\min\{2^{n/2}, \max\{n - \log Q, 1\}Q\})$$

if the QOBDD for $\mathcal{W}$ has already been built and under the assumption of constant time arithmetic.

Experiments

Table 6.1 shows the running times (in ms) of the algorithms in this chapter for some real world simple games. The algorithms have been implemented in C++ with a bare implementation of a QOBDD package with shared QOBDD nodes. Our test machine has an AMD Phenom II X4 955 processor and 16 GB of main memory. The columns two and three in Table 6.1 show the number of players and the size of the QOBDD for the simple game, respectively. The remaining columns refer to particular problems and algorithms, respectively.

The fourth column shows the time to build the QOBDD for the winning coalitions $\mathcal{W}$ by the algorithms presented in Chapter 4. The fifth column lists the average running time to compare two players w.r.t. to $\preceq_I$ and $\approx_I$, respectively, if the QOBDD approach in Theorem 6.22 is used. We have chosen a random set of samples here for some examples due to the large number of players. Because all games in Table 6.1, except the US Federal Legislative System, are complete, there is no need to perform any such comparison in practice. For complete simple games, the desirability relation on individuals can be obtained by the Chow parameters of the players as shown in Lemma 6.21.

The sixth column shows the running times to decide whether the simple game is proper and strong, respectively, by the algorithms in Section 6.4. The difference in the running times for the two algorithms are likely due to the fact that most games are proper but not strong and that it is easier to falsify a property.

The columns with titles $r_{\min}$ and r_{shift} show the running times to compute the QOBDDs for the sets $\mathcal{W}_{\min}$ and $\mathcal{W}_{\text{shift}}$, respectively. Because most games are directed, computing the QOBDD for $\mathcal{W}_{\min}$ is a linear time operation by the algorithm from Section 6.5. The only exception is the US Federal Legislative System again. But even for this game the QOBDD has been computed in less than a second. The computation of the QOBDD for $\mathcal{W}_{\text{shift}}$ is comparably expensive though. This does not only hold for time, but also for memory consumption. For the International Monetary Fund (IMF) we have not been able to compute the corresponding QOBDD due to insufficient memory.

The last two columns show the running times to compute the successor function succ from Section 6.8 and to count the number of models of the winning coalitions. For all instances except the IMF we have been able to compute succ in less than 100ms. For the US Federal Legislative System it should be mentioned in this respect, that this game has only four types and therefore, the successor function has a very simple structure.

We have not performed any experiments for the computation of the power indices. This is because these problems can be solved quickly in comparison to most other problems. Moreover, for the counting algorithms and thus, for the computation of the power indices, we have satisfying formal upper bounds for the running time.

Roughly speaking, we can summarize the experiments as follows. For a simple game whose QOBDD has just a few thousand nodes, most problems can be solved in less than a second. This can be considered as fast and the amount of available memory is not an issue. If the QOBDD becomes larger, say some millions of nodes, then we run into problems for some instances. Not only because it can take some time to solve a particular problem, but rather because the memory usage grows too fast. Fortunately, many simple games that appear in practice have rather small QOBDDs. To the author's knowledge, real world simple games with more than a few dozen of players and with more than three rules are rare.

			Running time (in ms)						
Name	n	$\mathsf{size}(r)$	build	$\preceq_I$ / $\approx_I$ (avg.)	Proper/strong?	$r_{\min}$	r_{shift}	succ	$\lvert\mathsf{models}(\mathcal{W})\rvert$
Canadian Constitution (1995)	10	37	0.07	0.01 / 0.01	0.01 / 0.01	0.03	0.32	0.02	0.01
Canadian Constitution (2005)	10	44	0.05	0.01 / 0.01	0.01 / 0.01	0.02	0.35	0.02	0.01
UN Security Council	15	53	0.04	0.01 / 0.01	0.04 / 0.02	0.09	1.3	0.06	0.01
German Bundesrat (2012)	16	162	0.3	0.06 / 0.04	0.01 / 0.01	0.07	2.1	0.04	0.01
Treaty of Nice	27	635	4.09	0.09 / 0.04	0.1 / 0.02	0.22	23	0.2	0.08
Treaty of Lisbon	27	4134	11.65	0.68 / 0.11	0.43 / 0.06	2.1	170	2.1	1.2
US Electoral College (2012-20)	51	4467	3.13	0.63 / 0.16	0.03 / 0.02	2.2	490	1.5	0.78
US Electoral College (2004-08)	51	4558	8.19	0.67 / 0.15	0.03 / 0.02	2.3	620	1.5	0.76
US Federal Legislative System	537	141650	393.54	42 / 40.7	22.07 / 3.4	390	128730	28.96	0.1
IMF (85%, 2009)	186	15712104	24894	14485 / 731	21468 / 226	36807	mem	37821	16649

Table 6.1.: Running times (in ms) for the algorithms in Chapter 6 for some real world simple games.

7. Weighted Representations

In comparison to explicit representations of simple games by their winning coalitions, weighted representations are compact and coherent, and therefore desirable for real world voting systems. However, finding a weighted representation for a simple game (if there is one) is not so easy. In Section 7.1 we will present a new method based on the QOBDD for a simple game and linear programming to decide if a simple game is weighted and to find a weighted representation if so. The method is evaluated in Section 7.2. Beside that, we will employ the property that the QOBDD representation of a weighted simple game is flat, regardless of the ordering of the players, for a heuristic to identify non-weighted simple games in Section 7.3. Based on this property, in Section 7.4 we will show how a witness of not being weighted in form of a 2-trade can be constructed from a non-flat QOBDD. This chapter is essentially based on the article Bolus (2011a).

There is rich literature on the problem to decide whether a Boolean function is a threshold function and a weighted voting game, respectively. For an overview in the area of electrical sciences, see the textbooks of Sheng (1969) and Lewis and Coates (1967). See the textbook of Taylor and Zwicker (1999) for an overview in the area of simple games and weighted voting games. Coates and Lewis (1961) use a so-called tree expansion technique based on backtracking. Hu (1963) uses integer linear programming to decide if a regular positive function (resp. directed simple game) is a threshold function. Freixas and Molinero (2008) use linear programming and the models of the shift-minimal winning and the shift-maximal losing coalitions to decide if a simple game is weighted. Because even the number of models can grow rapidly, new methods are necessary which can exploit the structure of the simple game. Additionally to the exact methods, there is a number of heuristics for the problem. Gowda and Vrudhula (2008) use a so-called tree-based decomposition for the synthesis of a threshold circuit made of threshold functions. Palaniswamy, Goparaju, and Tragoudas (2010) try to choose weights based on the so-called *modified chow parameters* which can be derived from the chow parameters and the number of winning coalitions.

Beyond that, the identification of threshold functions is a basic step in some approaches for threshold logic and network synthesis. Applications can be found in, for instance, Avedillo and Quintana (2004) and Zhang, Gupta, Zhong, and Jha (2005).

7.1. Characterization of Weighted Simple Games and Linear Programming

In this section we present a novel linear programming approach to decide if a QOBDD represents a weighted simple game and to find a real weighted representation if so. In contrast to most other chapters, the focus here is on weighted representations with real and, respectively, rational weights. Some of the ideas in this section are related to the characterization of linearly separable functions by Coates and Lewis (1961). These authors have used the characterization to develop an algorithm to find a real weighted representation of a simple game by means of a tree representation of the game; see Coates, Kirchner, and Lewis (1962) for a shorter version of the algorithm. Some parts of the algorithm have been rediscovered by Smaus (2007). The original algorithm consists of three stages. The first stage is a heuristic that tries to find weights for the players. If at some point the algorithm encounters a contradiction in the weights assigned so far, two other stages revise the current assignment and the algorithm either continues with the first stage or it terminates and reports that the input is not a linearly separable function, that is, it has no real weighted representation.

Coates and Lewis developed their algorithm long before the seminal papers on binary decision diagrams were published by Akers (1978) and Bryant (1986). Otherwise they had likely used QOBDDs instead of their redundant tree representation. The algorithm is quiet complex and it never became very popular. Its exact running time is unknown. Furthermore, the solutions found by the algorithm can be quite disappointing if you are looking for weighted representations with minimal weights for the players or with minimum quota.

Given the minimal winning coalitions $\mathcal{W}_{\min}$ and the maximal losing coalitions $\mathcal{L}_{\max}$ of a simple game $(N, \mathcal{W})$, there is an easy linear programming approach to decide if the simple game is weighted. An introduction to linear programming can, for instance, be found in the textbook by Karloff (1991). The linear program (LP) is:

$$\begin{array}{lrll}
\textbf{Minimize} & f(Q,\vec{w}) & & \\
\textbf{subject to} & \sum_{i\in S} w_i \geq Q & & \text{for all } S \in \mathcal{W}_{\min} \\
 & \sum_{i\in S} w_i \leq Q-1 & & \text{for all } S \in \mathcal{L}_{\max} \\
 & Q, w_1, \ldots, w_n \geq 0 & &
\end{array}$$

The objective function $f(Q,\vec{w})$ has to be linear. For instance, we can choose $f(Q,\vec{w})$ as Q to minimize the quota.

Example 7.1. The inequalities for the 3-person simple game with minimal winning coalitions $\mathcal{W}_{\min} = \{\{1\},\{2,3\}\}$ and maximal losing coalitions $\mathcal{L}_{\max} = \{\{2\},\{3\}\}$ are

$$\begin{array}{rr}
w_1 \geq Q\,, & w_2 \leq Q-1\,, \\
w_2 + w_3 \geq Q\,, & w_3 \leq Q-1
\end{array}$$

and $Q, w_1, w_2, w_3 \geq 0$. The simple game has the weighted representation $[2; 2, 1, 1]$. $\square$

The major drawback of this approach is the number of constraints which, in general, grows exponentially in n. Freixas and Molinero (2008) improve this linear programming approach. First, they use the shift-minimal winning and shift-maximal losing coalitions

	Players	$\mathsf{models}(\mathcal{W}_{\text{shift}})$	$\mathsf{models}(\mathcal{L}_{\text{shift}})$	QOBDD size
Canadian Constitution (1995)	10	1	2	37
Canadian Constitution (2005)	10	3	3	44
US Electoral College (2004-2008)	51	40 966 550	40 972 234	4558
German Bundesrat (2012)	16	12	12	162
Treaty of Nice	27	341	364	635
Treaty of Lisbon	27	20 202	19 927	4134
UN Security Council	15	1	2	53

Table 7.1.: Comparison of the number of models of the shift-minimal winning and shift-maximal losing coalitions and the QOBDD sizes for some real world simple games.

and second, they use models of these coalitions instead of the coalitions. The latter idea uses the fact that each WVG has a representation where all equally desirable players have the same weight. A weighted representation with this property is called type preserving. However, care has to be taken if the weights are restricted to be integers. In this case, some "minimum representations" are no feasible solutions of the corresponding integer linear program anymore; see Freixas and Molinero (2008) for details. For sake of convenience, we assume that the simple game $(N, \mathcal{W})$ is directed and the types are $N_1, \ldots, N_t$ where t is the number of types. Furthermore, we assume that the types $N_1, \ldots, N_t$ are ordered by decreasing desirability. Formally, if $i \in N_x$ and $j \in N_y$ for $x, y \in \{1, \ldots, t\}$ then $i \prec_I j$ if and only if $x > y$. Hence, N_1 contains the most desirable and N_t the least desirable players. The linear program with one weight for each type is:

$$\left.\begin{array}{lll}
\textbf{Minimize} & f(Q, N_1, \ldots, N_t, \vec{w}) & \\
\textbf{subject to} & \displaystyle\sum_{k=1,\ldots,t} m_k \cdot w_k \geq Q & \text{for all } \vec{m} \in \mathsf{models}(\mathcal{W}_{\text{shift}}) \\
& \displaystyle\sum_{k=1,\ldots,t} m_k \cdot w_k \leq Q - 1 & \text{for all } \vec{m} \in \mathsf{models}(\mathcal{L}_{\text{shift}}) \\
& w_i - 1 \geq w_{i+1} & \text{for all } i \in \{1, \ldots, t-1\} \\
& Q, w_1, \ldots, w_t \geq 0 &
\end{array}\right\} \quad (\text{LP}_{\text{models}})$$

To minimize the weights, the objective function has to be chosen as $\sum_{k=1}^{t} |N_k| \cdot w_k$.

The US Federal Legislative System has more than $5.62 \cdot 10^{160}$ winning coalitions but only two models for the shift-minimal winning coalitions, which can be obtained by the methods in Section 6.6 and Section 6.8. Therefore, the second LP can be significantly smaller than the first one. However, as can be seen from Table 7.1, for instance, for the US Electoral College (2004-2008) the number of models of $\mathcal{W}_{\text{shift}}$ is still huge. In comparison, the size of the QOBDD for this game is small. This is the main motivation for our novel approach based on the QOBDD structure here. Moreover, the computation of the models of $\mathcal{W}_{\text{shift}}$ and $\mathcal{L}_{\text{shift}}$ itself is relatively expensive as we have seen in Section 6.9, while the QOBDD for $\mathcal{W}$ is usually available without additional costs. We have used the methods in Chapter 6 to obtain the numbers in Table 7.1.

In the remainder of this section we assume that $N := \{1, \ldots, n\}$ is a set of $n \geq 1$ players and r is the root of a QOBDD for a subset of 2^N. In contrast to our usual assumptions, $\mathsf{set}(r)$ does not have to be an up-set.

The definition of a valid weight function below is central in the following presentation. It makes use of the functions l_w, u_w from Def. 4.1 on page 38.

Definition 7.1. A weight function $w : N \to \mathbb{R}_{\geq 0}$ is called *valid for the* QOBDD *with root r if $l_w(r) < u_w(r)$.* □

We will use the following characterization of being a real weighted representation for a simple game.

Theorem 7.1. *For $Q \in \mathbb{R}$ and a weight function $w : N \to \mathbb{R}_{\geq 0}$ the following statements are equivalent:*

1. *$(N, \mathsf{set}(r))$ is a simple game and $[Q; w]$ is a real weighted representation of it.*
2. *It holds $l_w(r) < Q \leq u_w(r)$ and therefore, w is valid for r.*

Proof. The equivalence follows from the statement of Lemma 4.5, which is:

$$l(r) < Q \leq u(r) \iff \mathsf{set}(r) = \{S \subseteq N \mid w(S) \geq Q\}\,. \tag{7.1}$$

The implication "$1 \implies 2$" holds, because if $(N, \mathsf{set}(r))$ has the real weighted representation $[Q; w]$, then $\mathsf{set}(r) = \{S \subseteq N \mid w(S) \geq Q\}$. By the direction "$\Longleftarrow$" of (7.1) it follows $l(r) < Q \leq u(r)$. The fact that w is valid for r can be seen without further proof, because $l(r) < u(r)$ is implied.

For the implication "$1 \Longleftarrow 2$" we assume that $l(r) < Q \leq u(r)$. By the direction "$\Longrightarrow$" of (7.1) it holds $\mathsf{set}(r) = \{S \subseteq N \mid w(S) \geq Q\}$. Therefore, $(N, \mathsf{set}(r))$ is a simple game with the real weighted representation $[Q; w]$. □

As a direct consequence, we obtain a very simple algorithm to test if a weight function can be used for a real weighted representation for the set represented by a QOBDD. For this to work, the QOBDD does not have to represent a simple game. If $\mathsf{set}(r)$ is not an up-set, then no weight function is valid for r.

Corollary 7.2. *For a weight function $w : N \to \mathbb{R}_{\geq 0}$ we can decide in time $\mathcal{O}(\mathsf{size}(r))$ whether $(N, \mathsf{set}(r))$ is a simple game and if there is a quota Q such that $[Q; w]$ is a real weighted representation of it.* □

Our next step is to develop a linear program based on the QOBDD with root r, that can be used to find a valid weight function for r or which has no feasible solution, if r does no represent a weighted voting game. The idea is to map the structure of the QOBDD and the values $l(v)$, $u(v)$, $v \in \mathsf{V}(r) \cup \{\mathbb{O}, \mathbb{I}\}$, directly into the variables and the constraints of the linear program. By doing this we obtain three types of variables in our LP. We have the weights $\vec{w} = (w_1, \ldots, w_n)$ and two vectors $\vec{x}$ and $\vec{y}$, where for each $v \in \mathsf{V}(r) \cup \{\mathbb{O}, \mathbb{I}\}$ we map $l_w(v)$ to the LP variable x_v and $u_w(v)$ to y_v. We denote a x_v, y_v

feasible solution of such a linear program as triple $(\vec{x}, \vec{y}, \vec{w})$.

In Lemma 4.1 we have seen how the values $l(v)$ and $u(v)$ can be calculated bottom-up for the inner nodes by the equations

$$\begin{aligned} l(v) &= \max\{l(\mathsf{then}(v)) + w(i), l(\mathsf{else}(v))\}\,, \\ u(v) &= \min\{u(\mathsf{then}(v)) + w(i), u(\mathsf{else}(v))\}\,, \end{aligned}$$

where v is an inner node with label i. The values for the sinks are constant with $l(\mathbb{I}) = -\infty$, $u(\mathbb{I}) = 0$, $l(\mathbb{O}) = 0$ and $u(\mathbb{O}) = \infty$. For the constraints of the linear program, we would like to use the aforementioned equations, but we cannot model the minimum and maximum in the LP. However, as we will see in Theorem 7.3 below, a relaxation with inequalities for the constraints is sufficient for our purposes.

The definition of a valid weight function w for r contains the strict inequality $l(r) < u(r)$ which is replaced by $x_r \le y_r - 1$ in the LP formulation. The set of feasible solutions still contains the set of all integer weighted representations, because the difference between the maximum weight of all losing coalitions and the minimum weight of all minimal coalitions is at least 1 for integer weights.

Let v be an inner node. In general, the value x_v can be negative as it is always the case for $x_{\mathbb{I}}$. By the equations above, $l(v)$ is negative only if $l(\mathsf{then}(v))$ and $l(\mathsf{else}(v))$ are negative. Consequently, x_v is negative only if $v = \mathbb{I}_{\mathsf{var}(v)}$ and therefore, only a small number of at most $n + 1$ inner nodes has a negative weight for x_v.

The linear program (LP$_r$) for the QOBDD with root r is given below, where, for instance, $f(\vec{x}, \vec{y}, \vec{w})$ can be chosen as $x_r + 1$ to minimize the quota.

$$\left.\begin{array}{lrll} \textbf{Minimize} & & f(\vec{x}, \vec{y}, \vec{w}) & \\ \textbf{subject to} & & x_r \le y_r - 1 & \\ & x_v \ge & \begin{cases} x_{\mathsf{then}(v)} + w_{\mathsf{var}(v)} \\ x_{\mathsf{else}(v)} \end{cases} & \text{for all } v \in \mathsf{V} \\ & y_v \le & \begin{cases} y_{\mathsf{then}(v)} + w_{\mathsf{var}(v)} \\ y_{\mathsf{else}(v)} \end{cases} & \text{for all } v \in \mathsf{V} \\ & x_{\mathbb{O}}, y_{\mathbb{I}} = & 0 & \\ & x_{\mathbb{I}} \le & -1 & \\ & y_{\mathbb{O}} \ge & 0 & \\ & x_v \le & -1 & \text{for all } v \in \mathsf{V}, v = \mathbb{I}_{\mathsf{var}(v)} \\ & x_v \ge & 0 & \text{for all } v \in \mathsf{V}, v \ne \mathbb{I}_{\mathsf{var}(v)} \\ & y_v \ge & 0 & \text{for all } v \in \mathsf{V} \\ & w_1, \ldots, w_n \ge & 0 & \end{array}\right\} \quad (\text{LP}_r)$$

The following theorem proves the correctness of the linear program. Its essential idea is, that we can obtain a feasible solution of (LP$_r$) from a real weighted representation $[Q; w]$ for $(N, \mathsf{set}(r))$ and vice versa. Because the difference $u_w(r) - l_w(r)$ might be less than 1 for real weighted representations, in such a case we can “spread” the difference by multiplying the weights with a constant $c \in \mathbb{R}_{\geq 1}$. As an example, assume that r represents the WVG $[2; 3/2, 1, 1/2]$. Then $u(r) = 2$ and $l(r) = 3/2$ and the difference is therefore $1/2$. By multiplying the weights with 2, the difference becomes 1.

Theorem 7.3. *Let $\vec{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n_{\geq 0}$ be a vector of weights and let w denote the weight function w with $w(i) := w_i$ for $i \in N$. If w is valid for r, then there are vectors $\vec{x}, \vec{y}$ and a constant $c \in \mathbb{R}_{\geq 1}$ such that $(\vec{x}, \vec{y}, c \cdot \vec{w})$ is a feasible solution of* (LP_r). *In turn, if $(\vec{x}, \vec{y}, \vec{w})$ is a feasible solution of* (LP_r) *then w is valid for r.*

Proof. For the first part we assume that w is valid for r, that is, $l_w(r) < u_w(r)$. We set the constant c to:

$$c := \max\{1, \frac{1}{u_w(r) - l_w(r)}\}\,.$$

It then follows $c \cdot (u_w(r) - l_w(r)) \geq 1$. We use the weight function $c \cdot w$, which is defined by $(c \cdot w)(i) := c \cdot w(i)$ for $i \in N$. By the definition of l and u it holds $l_{c \cdot w}(v) = c \cdot l_w(v)$ and $u_{c \cdot w}(v) = c \cdot u_w(v)$. Therefore, we arrive at $u_{c \cdot w}(r) - l_{c \cdot w}(r) \geq 1$ what is equivalent to $l_{c \cdot w}(r) \leq u_{c \cdot w}(r) - 1$. We obtain a feasible solution for (LP_r) by using $l_{c \cdot w}(v)$ for x_v and $u_{c \cdot w}(v)$ for y_v for all nodes $v \in \mathsf{V}(r)$ and $x_{\mathbb{I}}$ and $y_{\mathbb{O}}$.

For the second part we assume that $(\vec{x}, \vec{y}, \vec{w})$ is a feasible solution of (LP_r). First, by induction on the structure of the QOBDD we show that $x_v < y_v$ for every node $v \in \mathsf{V} \cup \{\mathbb{O}, \mathbb{I}\}$. The induction base is the root r and $x_r < y_r$ holds due to the constraint $x_r \leq y_r - 1$ in (LP_r). For the induction step, let v be an inner node in V with label i. We assume that $x_v < y_v$ holds. Because we have a feasible solution, from (LP_r) it follows

$$\left.\begin{array}{l} x_{\mathsf{then}(v)} + w_i \\ x_{\mathsf{else}(v)} \end{array}\right\} \leq x_v < y_v \leq \left\{\begin{array}{l} y_{\mathsf{then}(v)} + w_i \\ y_{\mathsf{else}(v)} \end{array}\right.$$

and therefore, we have $x_{\mathsf{then}(v)} < y_{\mathsf{then}(v)}$ and $x_{\mathsf{else}(v)} < y_{\mathsf{else}(v)}$.

Second, we show $l_w(v) \leq x_v$ and $y_v \leq u_w(v)$ for every node $v \in \mathsf{V}(r) \cup \{\mathbb{O}, \mathbb{I}\}$. Because we already know $x_v < y_v$, for $v = r$ it then follows

$$l_w(r) \leq x_r < y_r \leq u_w(r)$$

and thus, that w is valid for r, as claimed. This time, we start at the sinks. For the sinks, the statement is directly implied by the definition of the mappings l_w and u_w. Now, we assume to the contrary, that there is a node $v \in \mathsf{V} \cup \{\mathbb{O}, \mathbb{I}\}$ such that at least one of the inequalities does not hold for v. We choose the node v such that i is maximal. Because both inequalities hold for the sinks, v is an inner node. If $l_w(v) \leq x_v$ were violated, then this would be in contradiction to:

$$\begin{aligned} l_w(v) &= \max\{l_w(\mathsf{then}(v)) + w_i, l_w(\mathsf{else}(v))\} && \text{(Def. } l_w(v)) \\ &\leq \max\{x_{\mathsf{then}(v)} + w_i, x_{\mathsf{else}(v)}\} && \text{(Choice of } i) \\ &\leq x_v\,. && (\mathrm{LP}_r) \end{aligned}$$

Similarly, if $y_v \leq u_w(v)$ were violated, then this would be in contradiction to:

$$\begin{aligned} y_v &\leq \min\{y_{\mathsf{then}(v)} + w_i, y_{\mathsf{else}(v)}\} && (\mathrm{LP}_r) \\ &\leq \min\{u_w(\mathsf{then}(v)) + w_i, u_w(\mathsf{else}(v))\} && (\text{Choice of } i) \\ &= u_w(v)\,. && (\text{Def. } u_w(v)) \end{aligned}$$

Hence, $l_w(r) < u_w(r)$ and w is valid for r. □

The number of constraints and the number of variables in (LP_r) is bounded by $\mathcal{O}(\mathsf{size}(r))$. Because a linear program can be solved in polynomial time in its size we obtain we following complexity result.

Corollary 7.4. *We can decide in time $\mathcal{O}(\mathsf{poly}(\mathsf{size}(r)))$ whether r represents a weighted simple game, where $\mathsf{poly}(\mathsf{size}(r))$ is a polynomial in $\mathsf{size}(r)$.* □

If the types $N_1, \ldots, N_t$ are known in advance, then we can use one weight for each type in the linear program (LP_r), so that we end up with a type preserving real weighted representation if there is a feasible solution. To this end, we use the function $\tau : N \to \{1, \ldots, t\}$ with $\tau(i) = k$ if and only if $i \in N_k$. In (LP_r), each appearance of w_i is then replaced by $w_{\tau(i)}$ and the number of weights is reduced from n to t.

7.2. Experiments

In this section we present some experimental results which compare the *existing approach* with ($\mathrm{LP}_{\mathrm{models}}$) and the new QOBDD *approach* with (LP_r) that is based on the structure of the QOBDD for the winning coalitions. For our experiments we have used the dual simplex solver that comes with *Gurobi*[1] *4.5.* Our test machine has a AMD Phenom II X4 955 processor and 16 GB of main memory.

	$\min w(N)$		$\min Q$	
	Time (s)	Opt. Obj.	Time (s)	Opt. Obj.
UN Security Council	< 0.01	45	< 0.00	39
Canadian Constitution (1995)	< 0.01	-	< 0.01	-
Treaty of Nice	0.03	-	0.03	-
Treaty of Lisbon	0.29	-	0.57	-
US Electoral College (2004-2008)	0.73	538	1.32	270

Table 7.2.: Solver times for some real world voting systems.

Table 7.2 shows the solver times and the optimal objective values (or the symbol "-" if not weighted) for some real world simple games and the linear program (LP_r). The objective functions have been the sum of weights of all players, denoted by $\min w(N)$, and the quota, denoted by $\min Q$, in Table 7.2.

[1]See `http://www.gurobi.com`.

In the following, we compare both approaches by means of random weighted voting games with varying maximum weight for each player and quota $\lfloor \frac{1}{2} w(N) \rfloor$. For the QOBDD approach the players were always ordered by non-increasing weights. The maximum weights were 50, 100, 500, 1000, 5000, 10000 and 50000. For each number of players and each maximum weight we have evaluated 50 samples. The samples have been the same for both approaches.

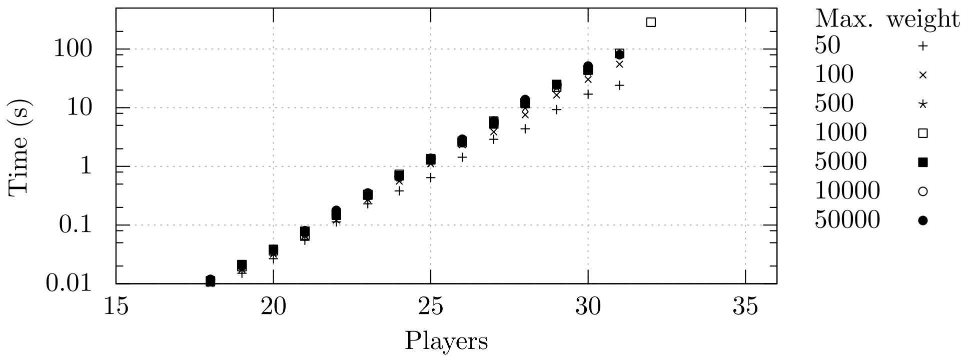


Figure 7.1.: Solver times for the existing approach with ($\mathrm{LP}_{\mathrm{models}}$).

Figure 7.1 shows the solver times for the existing approach with ($\mathrm{LP}_{\mathrm{models}}$). The time axis has a logarithmic scale. As can be seen, the solver time grows rapidly if the number of players increases, even if the maximum weight for each player and thus, the quota remains small. However, for games with up to 18 players, solving ($\mathrm{LP}_{\mathrm{models}}$) requires slightly more than 10 ms.

The benefit from using models instead of coalitions depends mainly on the number of types t in comparison to the number of players n. If $t = n$ then there are as many models as coalitions and any positive effect vanishes. While equally desirable players in real world games appear frequently, in random weighted voting games this is not the case anymore if the maximum allowed weight for each player grows.

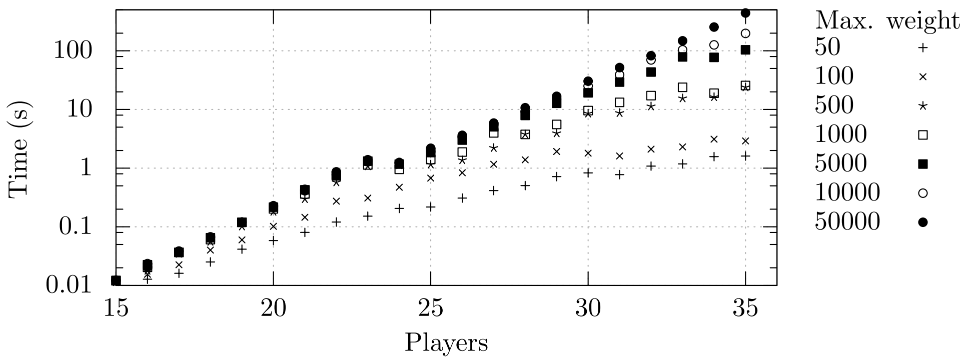


Figure 7.2.: Solver times for the QOBDD approach with (LP_r).

Figure 7.2 shows the solver times for the QOBDD approach with (LP$_r$). In comparison to the existing approach with (LP$_{\text{models}}$), the solver times for the QOBDD approach are strongly influenced by the maximum weight for the players. This is because small weights imply a small quota and the size of a QOBDD for a WVG is bounded in the size of the quota as we have seen in Section 5.3.

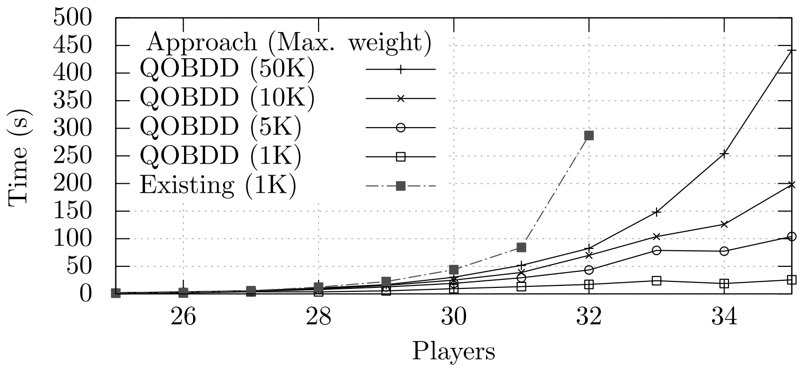


Figure 7.3.: Direct comparison of the existing approach with (LP$_{\text{models}}$) and the QOBDD approach with (LP$_r$).

Depending on the maximum weight, for just a few players the existing approach is superior over the QOBDD approach while the converse holds if the number of players increases or the maximum weight for each player remains small. This observation is justified by Figure 7.3 where we compare the solver times for the existing approach and a maximum weight of 1000 with the solver times of the QOBDD approach and different maximum weights.

7.3. Identifying non-weighted Simple Games using non-flat QOBDDs

Solving a linear program is usually considered to be efficient, because polynomial time algorithms exist and the (dual) simplex algorithm has good runtime behavior in practice. However, in our case the QOBDD size can grow rapidly in the number players, so that heuristics become necessary to decide if a simple game is not weighted. In this section we present a promising heuristic based on the fact, that weighted voting games have flat QOBDDs, regardless of the ordering of the players.

There are some necessary conditions for a weighted simple game that we have discussed and each of which can be used as a heuristic:

1. A weighted simple game is complete.
2. A weighted simple game is strong, proper or both.
3. A QOBDD for a weighted simple game is flat for any ordering of the players.

Here we discuss the 3rd necessary condition. In Section 5.1 we have seen, that the QOBDD for a simple game is flat for each ordering of the players if and only if the desirability relation on coalitions $\preceq_L$ is complete on 2^N. Taylor and Zwicker (1996) have noted in this respect, as a rule of thumb, that any non-weighted simple game that arises in practice as a voting system has a non-complete relation $\preceq_L$. Therefore, in practice we can expect to prove a simple game to be non-weighted by just testing its QOBDD representations to be flat for a sufficiently large number of orderings of the players. Before we shed light on the question of what "sufficiently large" could mean in practice, we discuss Algorithm 12 which can be used to decide if a QOBDD is flat.

Algorithm 12 *IsFlat*(v)

```
Require: $v$ is the root of a QOBDD with label 1.
1: for each $u \in \mathsf{V}(v) \cup \{\mathbb{I}, \mathbb{O}\}$ compute $c(u) := |\mathsf{set}(u)|$ as described in Section 6.2
2: for $i = n$ to 2 do
3:     sort $\mathsf{V}_i(v) = \{u_1, \dots, u_{|\mathsf{V}_i(v)|}\}$ such that $c(u_k) \leq c(u_{k+1})$
4:     for $k = 1$ to $|\mathsf{V}_i(v)| - 1$ do
5:         if $c(\mathsf{then}(u_k)) \not\leq c(\mathsf{then}(u_{k+1}))$ or $c(\mathsf{else}(u_k)) \not\leq c(\mathsf{else}(u_{k+1}))$ then
6:             return false
7: return true
```

In the proof of the correctness of algorithm *IsFlat*, we will make use of the following lemma that can easily be shown.

Lemma 7.5. *Let v be a* QOBDD *node and let i be an element from $\{\mathsf{var}(v), \dots, n+1\}$. If $\mathsf{V}_i(v)$ is totally ordered w.r.t. $\subset$, then it follows:*

$$\forall u, u' \in \mathsf{V}_i(v) : (u \subseteq u' \iff |\mathsf{set}(u)| \leq |\mathsf{set}(u')|) . \qquad \square$$

The correctness and running time of the algorithm are considered next.

Theorem 7.6. *Given the root v of a* QOBDD *with label 1, the algorithm IsFlat(v) returns true if and only if the* QOBDD *is flat. The running time is $\mathcal{O}(\mathsf{size}(v) \cdot \log \mathsf{width}(v))$.*

Proof. For $i = 1, \dots, n+1$ let m_i denote $|\mathsf{V}_i(v)|$. We show that the nodes on each level V_i are totally ordered w.r.t. $\subset$ if and only if *IsFlat*(v) returns true. For this purpose we use induction on the levels, starting with the sinks at level $n+1$. In the base case, we obviously have $\mathsf{set}(\mathbb{O}) \subset \mathsf{set}(\mathbb{I})$ for the sinks. For the induction hypothesis, let $i \in \{1, \dots, n\}$ and assume that $\mathsf{V}_{i+1} = \{x_1, \dots, x_{m_{i+1}}\}$ is totally ordered w.r.t. $\subset$, that is, $x_1 \subset \cdots \subset x_{m_{i+1}}$.

In the induction step we assume that the nodes $\mathsf{V}_i = \{u_1, \dots, u_{m_i}\}$ are ordered:

$$c(u_1) \leq c(u_2) \leq \cdots \leq c(u_{m_i}) .$$

This is the case after line 3. For $k \in \{1, \dots, m_i - 1\}$ it then holds:

$$\begin{aligned} & u_k \subseteq u_{k+1} \\ \iff & \mathsf{then}(u_k) \subseteq \mathsf{then}(u_{k+1}) \wedge \mathsf{else}(u_k) \subseteq \mathsf{else}(u_{k+1}) && \text{(Property } \subseteq\text{)} \\ \iff & c(\mathsf{then}(u_k)) \leq c(\mathsf{then}(u_{k+1})) \wedge c(\mathsf{else}(u_k)) \leq c(\mathsf{else}(u_{k+1})) . && \text{(Lemma 7.5)} \end{aligned}$$

For the last equivalence, Lemma 7.5 can be applied because of the induction hypothesis. Hence, after the inner **for**-loop in line 4, we have $u_1 \subseteq u_2 \subseteq \cdots \subseteq u_{m_i}$. Because the nodes $u_1, \ldots, u_{m_i}$ are different, the subset relations are proper.

To see the running time, we first notice that by the results in Section 6.2 the values $|\mathsf{set}(u)|$ for all $u \in \mathsf{V}(v) \cup \{\mathbb{O}, \mathbb{I}\}$ can be computed in time $\mathcal{O}(\mathsf{size}(v))$. For $i \in \{2, \ldots, n\}$, sorting the nodes in $\mathsf{V}_i(v)$ takes time $\mathcal{O}(m_i \log m_i)$. Therefore, the total running time for sorting all levels is

$$\mathcal{O}(\sum_{i=2}^{n} m_i \log m_i) \leq \mathcal{O}((\sum_{i=2}^{n} m_i) \cdot \log \mathsf{width}(v)) \leq \mathcal{O}(\mathsf{size}(v) \cdot \log \mathsf{width}(v)) . \qquad (7.2)$$

The **for**-loop in line 4 has running time $\mathcal{O}(m_i)$. The total running time of all iterations of line 5 is therefore, $\mathcal{O}(\sum_{i=2,\ldots,n} m_i) \leq \mathcal{O}(\mathsf{size}(r))$ and hence, dominated by (7.2). □

We have performed experiments on the average number of orderings of the players necessary to find a non-flat QOBDD for a non-weighted simple game. To this end, we randomly generated m-vector-weighted representations for $m = 1, 2, 3, 4, 5$. We have evaluated 1000 samples for each m and each number of players n. In an iteration we have generated a random permutation π of the players and tested the π-QOBDD for being flat.

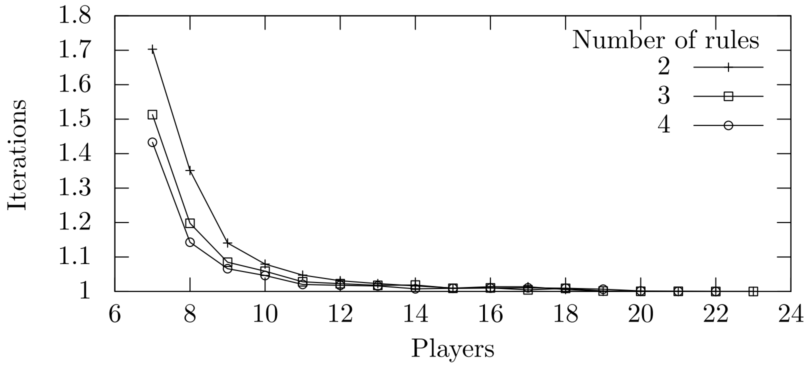


Figure 7.4.: Average number of orderings necessary to find a non-flat QOBDD for random vector-weighted representations.

Figure 7.4 shows the average number of iterations necessary to find a non-flat QOBDD for the simple game. Here, the necessary number of iterations falls very fast and starting at 13 players the average is nearly 1.

For Figure 7.5 only instances were created with weights $w_k(1) \geq \cdots \geq w_k(n)$ for each rule $k \in \{1, \ldots, m\}$. The simple game associated with a vector-weighted representation fulfilling this property is always complete. Here the number still falls quickly, but slower than in Figure 7.4. This is probably the case, because these games are more likely to have a flat QOBDD representation.

In conclusion it seems reasonable first to try to find a non-flat QOBDD representation before using the linear program to decide if a simple game is weighted.

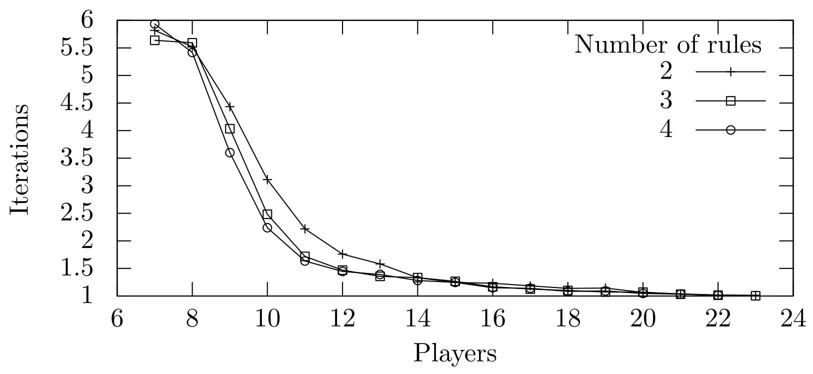


Figure 7.5.: Average number of orderings necessary to find a non-flat QOBDD for random vector-weighted representations with decreasing weights in each rule.

7.4. Counterexamples for being Weighted using non-flat QOBDDs

Having a simple game for which no weighted representation is known, for political scientists it can be considered as a standard procedure to decide whether the present simple game is weighted or to find a counterexample otherwise. The latter is usually in the form of exchanges of players between a collection of winning coalitions, a concept that has been introduced by Taylor and Zwicker (1992) for simple games. In this section we show how a counterexample, a so-called 2-trade, can be constructed if the QOBDD for the simple game is not flat.

Definition 7.2. A list of $k \geq 1$ coalitions $X_1, \dots, X_k$, $Y_1, \dots, Y_k \subseteq N$ is said to be a *(k-)trade*, if for each player $i \in N$ it holds:

$$|\{j \in \{1, \dots, k\} \mid i \in X_j\}| = |\{j \in \{1, \dots, k\} \mid i \in Y_j\}| . \qquad \square$$

The intuition behind a k-trade is the following. Take k coalitions $X_1, \dots, X_k$ and move players around as you like. The only restriction is, that every player has to appear the same number of times in the resulting coalitions $Y_1, \dots, Y_k$ as in the original ones.

Definition 7.3. A simple game $(N, \mathcal{W})$ is called *trade robust* if for any k there is no k-trade $X_1, \dots, X_k$, $Y_1, \dots, Y_k$ such that all the $X_1, \dots, X_k$ are winning and all the $Y_1, \dots, Y_k$ are losing. $\square$

Using this definition, Taylor and Zwicker (1992, Theorem A) have shown that a simple game is weighted if and only if it is trade robust. An important special case is a 2-trade with the additional restriction that only a one-for-one exchange of two players, a so-called *swap*, is allowed. A simple game is called *swap robust* if there is no such 2-trade for which the two coalitions are winning before the trade and they are losing after the swap. It is well-known that a simple game is swap robust if and only if it is complete (Taylor and Zwicker 1996). The concept of trade robustness for simple games is very much related to the concept of asummability for Boolean functions and threshold functions. See Sheng (1969, Section 4.4) for details on the latter.

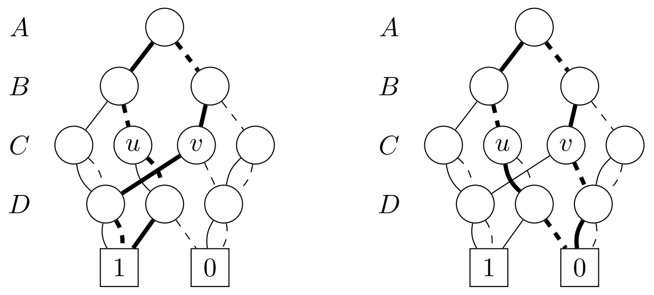


Figure 7.6.: QOBDD for $[2; 1, 2, 0, 1] \wedge [3; 2, 1, 2, 1]$ in Example 7.2 with winning coalitions AD and BC (left) and losing coalitions BD and AC (right).

Example 7.2. We consider the simple game associated with the multiple weighted representation

$$[2; 1, 2, 0, 1] \wedge [3; 2, 1, 2, 1] \tag{7.3}$$

and players A, B, C and D. The QOBDD for this game is depicted in Figure 7.6. Let X_1, X_2 denote the coalitions AD and BC, respectively. Both coalitions are winning. By exchanging player A for B we obtain the losing coalitions $Y_1 := BD$ and $Y_2 := AC$. Therefore, the simple game has no weighted representation. Because we have a 2-trade and an one-for-one exchange of players, the game is not even swap robust and therefore not complete. □

Taylor (1995) exemplifies the use of trades to show that the System to amend the Canadian Constitution and the US Federal Legislative System are not weighted. Felsenthal and Machover (2001, Section 4) use a trade to reason that the game for the Council of the European Union, as defined in the Treaty of Nice, is not weighted. Beside that, Freixas (2004) shows that the dimension of this game is in fact 3 after the enlargement to 27 members.

As we have already mentioned in the previous section, most real world simple games that are not weighted, have an ordering of the players such that the corresponding QOBDD is not flat. Therefore, in the following we can assume that the QOBDD with root r for the simple game $(N, \mathcal{W})$ is not flat. What remains is to find a witness for having no weighted representation. We construct a 2-trade that contradicts trade robustness.

Because the QOBDD is not flat, there is a level i and two different nodes $u, v \in \mathsf{V}_i(r)$ such that neither $\mathsf{set}(u) \supset \mathsf{set}(v)$ nor $\mathsf{set}(u) \subset \mathsf{set}(v)$. Consequently, both sets are not empty. Let $B_1 \in \mathsf{set}(u) \setminus \mathsf{set}(v)$ and let $B_2 \in \mathsf{set}(v) \setminus \mathsf{set}(u)$. For the players to be exchanged we choose $\Delta_1 := B_1 \setminus B_2$ and $\Delta_2 := B_2 \setminus B_1$. It holds:

$$(B_1 \setminus \Delta_1) \cup \Delta_2 = B_2\,,$$
$$(B_2 \setminus \Delta_2) \cup \Delta_1 = B_1\,.$$

To complete our 2-trade we choose any coalitions $A_1, A_2 \subseteq \{1, \dots, i-1\}$ such that A_1 is a path from the root to u, formally, $r \overset{A_1}{\rightarrow} u$, and A_2 is a path from the root to v, formally, $r \overset{A_2}{\rightarrow} v$. Then by Theorem 3.3 it follows that $A_1 \cup B_1$ and $A_2 \cup B_2$ are elements of $\mathsf{set}(r)$ and therefore, both coalitions are winning. On the other hand, by the choice of B_1, B_2 we get $A_1 \cup B_2 \notin \mathsf{set}(r)$ and also $A_2 \cup B_1 \notin \mathsf{set}(r)$ and therefore, the coalitions after the trade are losing. The trade can be formulated as:

$$\Delta_2 \cup (A_1 \cup B_1) \setminus \Delta_1 = A_1 \cup B_2 \quad \text{and} \quad \Delta_1 \cup (A_2 \cup B_2) \setminus \Delta_2 = A_2 \cup B_1 \,.$$

We close this section with an example that illustrates the construction.

Example 7.3. We consider the game (7.3) from Example 7.2 again, whose QOBDD is depicted in Figure 7.6. The nodes u, v are incomparable w.r.t. $\subset$, because $\mathsf{set}(u) = \{D, CD\}$ and $\mathsf{set}(v) = \{C, CD\}$. Therefore, the QOBDD is not flat. The only options for B_1 and B_2 are $\{D\}$ and $\{C\}$, respectively. It follows $\Delta_1 = B_1 \setminus B_2 = \{D\}$ and $\Delta_2 = \{C\}$. As can be seen in Figure 7.6 (left), we have $r \overset{\{A\}}{\rightarrow} u$ and $r \overset{\{B\}}{\rightarrow} v$. We set $A_1 := \{A\}$ and $A_2 := \{B\}$. In Figure 7.6 (right) it can now be seen that neither $A_1 \cup B_2 = AC$ nor $A_2 \cup B_1 = BD$ is winning. □

7.5. Conclusions

It is an important problem to identify weighted voting games and to prove the opposite otherwise. The method based on QOBDDs and the linear program (LP$_r$), that we have presented in this chapter, can be seen complementary to the existing approach with the linear program (LP$_{\text{models}}$) that uses the models of the shift-minimal winning and the shift-maximal losing coalitions, as it is somewhat slower on instances with only a few players, but it significantly outperforms the existing method when the number of players increases while the quota, and hence the QOBDD size, remains small.

We were able to test most of the real world simple games in this thesis in a couple of seconds and thus, from a practical point of view our contribution fills an important gap.

The formulation of (LP$_r$) offers some room for optimizations, that have not been considered in Section 7.1. For instance, it seems that for many QOBDDs it holds that most nodes v fulfill $u(v) = l(v) + 1$ and hence, the weighted representation for the simple game represented by v is *normal* (Sheng 1969). If this would be known in advance, the variables x_v and y_v could be replaced by a single variable in the linear program. Another possible optimization is to merge the constraints of nodes which have only a single incoming edge.

Solving a linear program can still be considered intractable for very large QOBDDs. For these instances, we have presented a heuristic that can be used to identify a QOBDD that does not represent a weighted simple game. To this end, we have used the fact that a QOBDD for a WVG is flat regardless of the ordering of the players. We have evaluated the performance of this approach by random experiments with vector-weighted representations. It would be especially interesting to obtain deeper insight into the number of orderings of the players for which a QOBDD is flat.

In the case that the simple game is not weighted and the QOBDD representation is not flat, we have presented a method to derive a witness in form of a 2-trade. This form of a witness is widely used to prove, that a simple game is not weighted. It is an open question how to find a minimum 2-trade though.

8. Conclusions and Future Work

Nowadays, voting systems play an important role in decision making processes in politics and economy. Unfortunately, voting systems tend to be complicated and elusive. To ensure that voting systems express what we mean, we therefore rely on methods that support the analysis and the understanding of such voting systems. In this thesis we have presented a practically applicable approach to represent and to analyze a particular type of voting systems, namely simple games.

Our main idea has been to represent simple games, which are essentially up-sets, by quasi-reduced and ordered binary decision diagrams, or QOBDDs for short. Problems on simple games have been solved on the basis of the representation as QOBDDs. Examples for such problems have been the computation of power indices to measure a priori voting power and to decide whether a simple game has a weighted representation and if so, to find a minimal one. In practice, various representations of simple games, like weighted representations, are used. A variety of representations in practice can be challenging though, because each of them requires the development of dedicated algorithms for the problems. In contrast to that, we have used QOBDDs to split off the representation which is used in practice from the algorithms to solve the problems on simple games. After we have laid the necessary foundations in Part I of this thesis, in Part II we have discussed the step from a representation of a simple game in practice to a QOBDD, while in Part III we have developed algorithms on the basis of QOBDDs.

By the separation, we are not tight to a particular representation in practice anymore. Once we have found a way to obtain a QOBDD from a particular representation, we can apply all the algorithms that we have presented in this thesis. Therefore, our approach can be considered as an all in all approach by which we can solve a problem for any simple game. Obviously, there are many potential representations of simple games that could have been used for that purpose. For instance, we could have enumerated the set of the minimal winning coalitions, or we could have used a vector-weighted representation with $|\mathcal{L}_{\max}|$ rules. It is well-known that every simple game has such a representation and algorithms exist to solve at least some of the aforementioned problems. The difference is, that QOBDDs are often much more compact for practically relevant instances.

The separation between the representation of simple games in practice and algorithms to solve particular problems involves the risk of losing information that could otherwise be used to solve some problems more quickly. For instance, if a simple game is weighted, it is desirable to exploit that fact. QOBDDs are rather accommodating in this respect. We have seen for some classes of simple games, that the QOBDDs exhibit structural features. Examples include weighted voting games in Section 4.1 and homogeneous simple games in Section 5.4. We have used these features twofold. As for WVGs in Section 5.3 and homogeneous simple games in Section 5.4 we have derived upper bounds for the size of the QOBDD for classes of simple games. We have also used the structure of QOBDDs to derive more efficient algorithms. For the important class of directed simple games we have developed a specialized algorithm to obtain the QOBDD for the minimal winning coalitions with running time linear in the size of the QOBDD for $\mathcal{W}$. Hence, even though QOBDDs can represent any simple game, for classes of simple games, they often exhibit structural features that facilitate the solution of particular problems.

One of our main goals has been to solve problems for real world simple games. Even though in general almost all QOBDDs have a size that is exponential in the number of decision variables, QOBDDs for real world simple games are often surprisingly small. As we have discussed in the introduction to Chapter 4, simple games in practice are often made of various weighted representations, each of which is inherently simple. This simplicity induces, that also the QOBDD for the overall simple game has a particular structure and small size. We have evaluated our approach by a list of different real world simple games. Our results suggest, that except for very large instances, our approach solves most problems in acceptable time. The only exception is the computation of the QOBDDs for the shift-minimal winning and the shift-maximal losing coalitions.

Most algorithms in this thesis are short and concise. The algorithms to compute the successor function in Section 6.8 are an exception here. Therefore, it is without difficulty to implement the algorithms if a QOBDD package for the fundamentals, like computed tables, binary synthesis and manipulators, is provided. By virtue of manipulators and the binary synthesis, for some problems we have been able to derive an algorithm just by rewriting the formal specification as in the case of the desirability relation on individuals in Section 6.3. It has been especially easy to perform the transition from weighted voting games to multiple weighted voting games with a formula, because the binary synthesis for QOBDDs is well-known.

For some algorithms, we have been able to establish useful formal upper bounds for the (expected) running times. In Section 6.7 we have seen, that we can compute some power indices in time polynomial in the size of the QOBDD for $\mathcal{W}$. Consequently, for any class of simple games whose QOBDD representations have size polynomial in the number of players n, we immediately obtain algorithms with running time polynomial in n to compute the considered power indices. Examples for such classes are homogeneous simple games and weighted voting games with polynomially bounded weights in n.

Many results in this thesis are not limited to simple games. Even though many results have been stated in the context of simple games, this is mainly due to the topic of the thesis. The notions of a simple game and a monotone Boolean function and, respectively, a weighted voting game and a threshold function, are interchangeably in many statements. Additionally, QOBDDs are well established in a wide variety of research fields. For that reason, some of our results directly contribute to other research areas like threshold logic. In threshold logic it is a major problem to decide whether a Boolean function is representable as a threshold function and to find such a representation.

Even though the QOBDD for $[Q; w_1, \ldots, w_n]$ has size $\mathcal{O}(\min\{2^{n/2}, \max\{n - \log Q, 1\}Q\})$, it is sometimes too large for some problems to be solved. We have encountered this issue for the International Monetary Fund and the computation of the QOBDD for $\mathcal{W}_{\text{shift}}$. For the considered problems, it usually does not matter if it takes a second, a minute, or even an hour to solve a problem. Unfortunately, this does not hold for memory usage. The available amount of memory is the major limitation of our approach if the instances are very large. For most instances, however, the space complexity is not an issue.

Future Work

In general, it would be interesting to see more applications of the QOBDD-specific methods and ideas that we have presented in this thesis to other research problems. The linear programming approach and the heuristic to identify QOBDDs, that do not represent a weighted voting game (resp. a threshold function) from Section 7 could be interesting to the field of electrical engineering. Especially, because reduced and ordered binary decision diagrams, or ROBDDs for short, are very well established there. Another example is the computation of the Chow parameters for a Boolean function represented by a ROBDD or a QOBDD. This is an important step to identify non-symmetric variables in general Boolean functions (Möller, Mohnke, and Weber 1993). While current methods have time complexity $\mathcal{O}(n \cdot \mathsf{size}(r))$ for an ROBDD and QOBDD with root r, respectively, our approach for QOBDDs could be an alternative to those methods, because it takes only linear time in the size of the input QOBDD. An adaption of our approach to ROBDDs requires some effort though.

In the introduction to Chapter 4 we have briefly presented two types of WVGs (resp. decision rules), that are often used in practice to design voting systems. Even though most problems on weighted representations are NP-hard, it is an open question whether "hard" instances even exist in the real world. In the same vein, it would be also interesting to identify further paradigms that are used in practice to design voting systems and to derive a class of simple games from that, if possible.

The algorithm to build a QOBDD with root r from a weighted representation, which we have discussed in Section 4.2, has (expected) running time $\mathcal{O}(\mathsf{size}(r) \cdot \log \mathsf{width}(r))$. The logarithmic factor is due to the use of AVL trees for the identification of previously computed results. It is an open question, if there is a data structure that exploits the specific structure of the problem, that has (maybe amortized) constant time operations for insertion and lookup.

In Section 5.2 we have defined the class of flat QOBDDs. These QOBDDs rise several questions. For instance, is it NP-hard to find an ordering of the players (resp. decision variables) such that the QOBDD is flat? Hosaka, Takenaga, Kaneda, and Yajima (1997) study similar questions in the context of threshold functions. It seems interesting to generalize some of their results to the class of flat QOBDDs. Furthermore, what is the number of orderings so that the QOBDD is flat? An answer to this question would be interesting in the context of our heuristic to identify QOBDDs that do not represent WVGs in Section 7.3.

We have considered the problem whether a simple games has a weighted representation and how to find one in Section 7. The same problem can be asked for a given number of rules m and a formula φ. The question then is:

Is there a multiple weighted representation with formula φ and m rules?

For the Council of the European Union as defined in the Treaty of Nice, we have seen an alternative representation in Chapter 1. This representation has been obtained by an integer linear program (ILP), the models of the coalitions $\mathcal{W}_{\min}$ and $\mathcal{L}_{\max}$ and the so-called Big-M method. The latter has been used to model disjunctions inside the ILP (Chen, Batson, and Dang 2010). However, things are much more complicated here and there are some fundamental questions. For WVGs it is well-known, that there always is a type preserving weighted representation, that is, there is a weighted representation $[Q; w_1, \ldots, w_n]$ such that $i \approx_I j$ implies $w_i = w_j$ for all players $i, j = 1, \ldots, n$. This does not hold for vector-weighted representations and therefore, multiple weighted representations anymore. To get the idea, we show that the directed simple game

$$[3; 3, 0, 1, 1, 1, 1] \wedge [4; 1, 3, 1, 1, 1, 1]$$

with 6 players and types $N_1 = \{A, B\}$ and $N_2 = \{C, D, E, F\}$ does not possess a type preserving m-vector-weighted representation for any m. The models of the minimal winning coalitions include $(2, 0)$ and $(0, 4)$. The vector $(1, 2)$ is a model of a maximal losing coalition. It can be represented as convex combination of $(2, 0)$ and $(0, 4)$ by

$$(1, 2) = \lambda(2, 0) + (1 - \lambda)(0, 4) \tag{8.1}$$

with $\lambda = 1/2$. We assume to the contrary that there is a type preserving m-vector-weighted representation with rules $g_1, \ldots, g_m$ and $g_k = [Q_k; w_k]$. Because the representation is type preserving it holds $w_k(A) = w_k(B)$ and $w_k(C) = w_k(D) = w_k(E) = w_k(F)$. The necessary information for the representation can be stated more compactly as:

$$Y := \begin{pmatrix} w_1(A) & w_1(C) \\ \vdots & \vdots \\ w_m(A) & w_m(C) \end{pmatrix} \quad \text{and} \quad \vec{Q} := \begin{pmatrix} Q_1 \\ \vdots \\ Q_m \end{pmatrix}.$$

We use the superscript ^{t} to transpose a vector. It holds $Y(2, 0)^t \geq \vec{Q}$ and $Y(0, 4)^t \geq \vec{Q}$ for the models of the coalitions in $\mathcal{W}_{\min}$ and $Y(1, 2)^t \ngeq \vec{Q}$ for the model of a coalition in $\mathcal{L}_{\max}$. By using the convex combination in (8.1) we get a contradiction with:

$$Y \begin{pmatrix} 1 \\ 2 \end{pmatrix} = Y(\lambda \begin{pmatrix} 2 \\ 0 \end{pmatrix} + (1 - \lambda) \begin{pmatrix} 0 \\ 4 \end{pmatrix}) = \frac{1}{2}(Y \begin{pmatrix} 2 \\ 0 \end{pmatrix} + Y \begin{pmatrix} 0 \\ 4 \end{pmatrix}) \geq \frac{1}{2}(\vec{Q} + \vec{Q}) = \vec{Q}\,.$$

The main idea of the previous example has been, that the set $\mathsf{models}(\mathcal{L}_{\max})$ and the convex hull of $\mathsf{models}(\mathcal{W}_{\min})$ are not disjoint. Hence, this property is a necessary condition for having a type preserving vector-weighted representation. A sufficient condition for this property remains as future work.

Bibliography

Adelson-Velskii, M. and E. M. Landis (1963). An Algorithm for the Organization of Information. In *Proceedings of the USSR Academy of Sciences*, Volume 146, pp. 263–266. Russian.

Akers, S. B. (1978). Binary Decision Diagrams. *IEEE Trans. Comput. 27*(6), 509–516.

Akers, S. B. and B. H. Rutter (1964). The use of threshold logic in character recognition. *Proceedings of the IEEE 52*(8), 931–938.

Algaba, E., J. Bilbao, J. Fernández García, and J. López (2003, August). Computing power indices in weighted multiple majority games. *Mathematical Social Sciences 46*(1), 63–80.

Algaba, E., J. M. Bilbao, and J. R. Fernández (2007). The distribution of power in the european constitution. *European Journal of Operational Research 176*(3), 1752–1766.

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

Alonso-Meijide, J. M., J. M. Bilbao, B. Casas-Méndez, and J. R. Fernández (2009, October). Weighted Multiple Majority Games with Unions: Generating Functions and Applications to the European Union. *European Journal of Operational Research 198*(2), 530–544.

Alonso-Meijide, J. M. and M. G. Fiestras-Janeiro (2002). Modification of the Banzhaf Value for Games with a Coalition Structure. *Annals of Operations Research 109*(1), 213–227.

Alonso-Meijide, J. M. and J. Freixas (2010). A new power index based on minimal winning coalitions without any surplus. *Decision Support Systems 49*(1), 70–76.

Avedillo, M. J. and J. M. Quintana (2004). A threshold logic synthesis tool for rtd circuits. *Proceedings of the Digital System Design, EUROMICRO Systems*, 624–627.

Aziz, H. and M. Paterson (2008). Computing voting power in easy weighted voting games. arXiv:0811.2497v2 [cs.GT]. 1–12.

Bachrach, Y., J. S. Rosenschein, and E. Porat (2008). Power and Stability in Connectivity Games. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pp. 999–1006. International Foundation for Autonomous Agents and Multiagent Systems.

Banzhaf, J. F. (1965). Weighted Voting Doesn't Work: A Mathematical Analysis. *Rutgers Law Review 19*(2), 317–343.

Behle, M. (2008). On threshold BDDs and the optimal variable ordering problem. *Journal of Combinatorial Optimization 16*(2), 107–118.

Behnke, R., R. Berghammer, E. Meyer, and P. Schneider (1998). RELVIEW — A system for Calculating with Relations and Relational Programming. *Lecture Notes in Computer Science 1382*, 318–321.

Berghammer, R. and S. Bolus (2010). Problem solving on simple games via BDDs. In V. Conitzer and J. Rothe (Eds.), *Proceedings of the 3rd International Workshop on Computation Social Choice*, pp. 11–12. Abstract.

Berghammer, R. and S. Bolus (2012). On the Use of BDDs for Solving Problems on Simple Games. *European Journal of Operational Research*, 1–21. Accepted for publication.

Berghammer, R., S. Bolus, A. Rusinowska, and H. de Swart (2011). A relation-algebraic approach to simple games. *European Journal of Operational Research 210*(1), 68–80.

Berghammer, R., B. Leoniuk, and U. Milanese (2002). Implementation of Relational Algebra using Binary Decision Diagrams. In *Proceedings of the 6th International Conference on Relational Methods in Computer Science (RelMICS 2001), LNCS 2561*, pp. 241–257. Springer.

Bohossian, V. (1998). *Neural Logic: Theory an Implementation.* Ph. D. thesis, California Institute of Technology.

Bollig, B. and I. Wegener (1996). Improving the variable ordering of OBDDs is NP-complete. *Computers, IEEE Transactions on 45*(9), 993–1002.

Bolus, S. (2011a). On Finding Weighted Representations of Simple Games by Linear Programming and Binary Decision Diagrams. *Submitted to Mathematical Social Sciences, Sept. 2011*, 1–17.

Bolus, S. (2011b). Power indices of simple games and vector-weighted majority games by means of binary decision diagrams. *European Journal of Operational Research 210*(2), 258–272.

Bolus, S. (2011c). Testing homogeneity of directed simple games. *Submitted to Anals of Operations Research, April 2011*, 1–18.

Brace, K. S., R. L. Rudell, and R. E. Bryant (1990). Efficient Implementation of a BDD Package. In *DAC '90: Proceedings of the 27th ACM/IEEE Design Automation Conference*, pp. 40–45. ACM.

Bryant, R. E. (1986). Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on computers 35*(8), 677–691.

Bryant, R. E. (1992). Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv. 24*(3), 293–318.

Carreras, F. and J. Freixas (1996). Complete simple games. *Mathematical Social Sciences 32*(2), 139–155.

Chakravarty, N., A. M. Goel, and T. Sastry (2000). Easy weighted majority games. *Mathematical Social Sciences 40*(2), 227–235.

Chen, D.-S., R. G. Batson, and Y. Dang (2010, January). *Applied Integer Programming: Modeling and Solution.* Wiley.

Chow, C. K. (1961). On the characterization of threshold functions. In *Proceedings of the Second Annual Symposium on Switching Circuit Theory and Logical Design (SWCT)*, pp. 34–38. IEEE Computer Society.

Coates, C. L., R. B. Kirchner, and P. M. Lewis (1962). A Simplified Procedure for the Realization of Linearly-Separable Switching Functions. *IEEE Transactions on Electronic Computers EC-11*(4), 447–458.

Coates, C. L. and P. M. Lewis (1961). Linearly separable switching functions. *Journal of the Franklin Institute 272*(5), 366–410.

Cormen, T. H., C. E. Leiserson, and R. L. Rivest (2001, September). *Introduction to Algorithms, Second Edition.* MIT Press.

Daskalakis, C., R. M. Karp, E. Mossel, S. J. Riesenfeld, and E. Verbin (2011). Sorting and selection in posets. *SIAM Journal on Computing 40*(3), 597–622.

Deegan, J. and E. W. Packel (1978). A New Index of Power for Simple n-Person Games. *International Journal of Game Theory 7*(2), 113–123.

Deineke, V. and G. Woeginger (2006). On the dimension of simple monotonic games. *European Journal of Operational Research 170*(1), 315–318.

Einy, E. (1985, October). The desirability relation of simple games. *Mathematical Social Sciences 10*(2), 155–168.

Faliszewski, P., E. Elkind, and M. Wooldridge (2009, May). Boolean combinations of weighted voting games. In Decker, Sichman, Sierra, and Castelfranchi (Eds.), *Proc. of 8th Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Budapest, Hungary, pp. 185–192.

Felsenthal, D. S. and M. Machover (2001, July). The Treaty of Nice and qualified majority voting. *Social Choice and Welfare 18*, 431–464.

Felsenthal, D. S. and M. Machover (2004). A Priori Voting Power: What Is It All About? *Political Studies Review 2*(1), 1–23.

Freixas, J. (2004). The dimension for the European Union Council under the Nice rules. *European Journal of Operational Research 156*(2), 415–419.

Freixas, J. and X. Molinero (2008). On the existence of a minimum integer representation for weighted voting systems. *Annals of Operations Research 166*(1), 243–260.

Freixas, J., X. Molinero, M. Olsen, and M. Serna (2012, January). On the complexity of problems on simple games. *RAIRO - Operations Research 45*(4), 295–314.

Gowda, T. and S. Vrudhula (2008). Decomposition based approach for synthesis of multi-level threshold logic circuits. *Proceedings of the 2008 Asia and South Pacific Design Automation Conference*, 125–130.

Heinemann, F. (2003). The political economy of EU enlargement and the Treaty of Nice. *European Journal of Political Economy 19*(1), 17–31.

Holler, M. J. (1982). Forming Coalitions and Measuring Voting Power. *Political Studies 30*(2), 262–271.

Hosaka, K., Y. Takenaga, T. Kaneda, and S. Yajima (1997, June). Size of ordered binary decision diagrams representing threshold functions. *Theoretical Computer Science 180*(1-2), 47–60.

Hosaka, K., Y. Takenaga, and S. Yajima (1994). On the Size of Ordered Binary Decision Diagrams Representing Threshold Functions. In *ISAAC '94: Proceedings of the 5th International Symposium on Algorithms and Computation*, London, UK, pp. 584–592. Springer-Verlag.

Hu, S.-T. (1963). Synthesis of integral minimal weights. Technical report, Lookheed Missiles and Space Company.

Hu, S.-T. (1965). *Threshold Logic.* University of California Press.

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

Iwama, K., M. Nozoe, and S. Yajima (1998). Optimizing OBDDs is still intractable for monotone functions. In *Mathematical Foundations of Computer Science*, Volume 1450, pp. 625–635. Berlin/Heidelberg: Springer Berlin / Heidelberg.

Johnston, R. J. (1978, August). On the measurement of power: some reactions to Laver. *Environment and Planning A 10*(8), 907–914.

Karloff, H. (1991). *Linear Programming* (1 ed.). Birkhäuser Boston.

Kirsch, W. and J. Langner (2011). Invariably suboptimal: An attempt to improve the voting rules of the treaties of nice and lisbon. *Journal of Common Market Studies 49*(6), 1317–1338.

Kirstein, R. (2009, March). Volkswagen vs. Porsche. A Power-Index Analysis. Working Paper 07, Faculty of Economics and Management Magdeburg.

Klinz, B. and G. J. Woeginger (2005). Faster algorithms for computing power indices in weighted voting games. *Mathematical Social Sciences 29*(1), 111–116.

Kurz, S. (2011, March). On minimum sum representations for weighted voting games. arXiv:1103.1445v1 [math.CO]. 1–6.

Kurz, S. and S. Napel (2012, February). Heuristic and exact solutions to the inverse power index problem for small voting bodies. arXiv:1202.6245v1 [math.CO]. 1–16.

Kurz, S. and N. Tautenhahn (2012, April). On dedekinds problem for complete simple games. *Accepted for Publication in International Journal of Game Theory*, 1–19.

Lapidot, E. (1972). The counting vector of a simple game. *Proc. Amer. Math. Soc 31*, 228–231.

Lee, C. Y. (1959). Representation of Switching Circuits by Binary-Decision Programs. *Bell System Technical Journal* (38).

Leech, D. (1998, May). Power relations in the international monetary fund: a study of the political economy of a priori voting power using the theory of simple games. Technical report, University of Warwick. Centre for the Study of Globalisation and Regionalisation, Coventry.

Leech, D. (2002). Computation of Power Indices. The Warwick Economics Research Paper Series (TWERPS) 644, University of Warwick, Department of Economics.

Leech, D. (2002). Designing the Voting System for the Council of the European Union. *Public Choice 113*, 437–464. 10.1023/A:1020877015060.

Lewis, P. M. and C. L. Coates (1967). *Threshold logic.* New York: Wiley.

Liaw, H.-T. and C.-S. Lin (1992). On the OBDD-representation of general Boolean functions. *Computers, IEEE Transactions on 41*(6), 661–664.

Lucchetti, R. and P. Radrizzani (2010). Microarray data analysis via weighted indices and weighted majority games. In *Lecture Notes in Computer Science*, Volume 6160, pp. 179–190. Springer Berlin / Heidelberg.

Matsui, T. and Y. Matsui (2000). A Survey of Algorithms for Calculating Power Indices of Weighted Majority Games. *Journal of the Operations Research Society of Japan 43*, 71–86.

Matsui, Y. and T. Matsui (2001). NP-completeness for Calculating Power Indices of Weighted Majority Games. *Theoretical Computer Science 263*(1-2), 306–310.

Minato, S., N. Ishiura, and S. Yajima (1990, June). Shared binary decision diagram with attributed edges for efficient Boolean function manipulation. In *27th ACM/IEEE Design Automation Conference, 1990. Proceedings*, pp. 52–57. IEEE.

Möller, D., J. Mohnke, and M. Weber (1993). Detection of symmetry of Boolean functions represented by ROBDDs. In *ICCAD '93*, Los Alamitos, CA, USA, pp. 680–684. IEEE Computer Society Press.

Morgenstern, O. and J. von Neumann (1944). *Theory of Games and Economic Behavior.* Princeton University Press.

OEIS Foundation Inc. (2012). The on-line encyclopedia of integer sequences. http://oeis.org.

Ossowski, J. and C. Baier (2008, September). A uniform framework for weighted decision diagrams and its implementation. *Int. J. Softw. Tools Technol. Transf. 10*(5), 425–441.

Östergård, P. R. J. (2002, August). A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics 120*(13), 197–207.

Ostmann, A. (1987). On the minimal representation of homogeneous games. *International Journal of Game Theory 16*(1), 69–81.

Palaniswamy, A. k., M. k. Goparaju, and S. Tragoudas (2010). Scalable identification of threshold logic functions. *Proceedings of the 20th symposium on Great lakes symposium on VLSI*, 269–274.

Peleg, B. and P. Sudhölter (2007). *Introduction to the Theory of Cooperative Games*. Springer US.

Rosenmüller, J. (1984). Weighted Majority Games and the Matrix of Homogeneity. *Zeitschrift für Operations Research (ZOR) 28*(5), 123–141.

Rosenmüller, J. (1987). An algorithm for the construction of homogeneous games. In *Ökonomie und Mathematik*, pp. 65–76. Springer Verlag.

Shapley, L. S. and M. Shubik (1954). A Method for Evaluating the Distribution of Power in a Committee System. *The American Political Science Review 48*(3), 787–792.

Sheng, C. L. (1969). *Threshold Logic*. Academic Press Inc.

Sieling, D. and I. Wegener (1993). NC-Algorithms for Operations on Binary Decision Diagrams. *Parallel Processing Letters 3*, 3–12.

Smaus, J.-G. (2007). On boolean functions encodable as a single linear pseudo-boolean constraint. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pp. 288–302.

Somenzi, F. (1998). CUDD: CU Decision Diagram Package Release. online.

Somenzi, F. (1999). Binary Decision Diagrams. In *Calculational System Design, volume 173 of NATO Science Series F: Computer and Systems Sciences*, pp. 303–366. IOS Press.

Sudhölter, P. (1989). Homogeneous games as anti step functions. *International Journal of Game Theory 18*(4), 433–469.

Takenaga, Y., M. Nouzoe, and S. Yajima (1997). Size and variable ordering of OBDDs representing threshold functions. In T. Jiang and D. Lee (Eds.), *Computing and Combinatorics*, Volume 1276 of *Lecture Notes in Computer Science*, pp. 91–100. Springer.

Taylor, A. D. (1995). *Mathematics and Politics: Strategy, Voting, Power, and Proof* (1 ed.). Springer.

Taylor, A. D. and W. S. Zwicker (1992). A characterization of weighted voting. *Proceedings of the American Mathematical Society 115*(4), 1089–1094.

Taylor, A. D. and W. S. Zwicker (1993, January). Weighted voting, multicameral representation, and power. *Games and Economic Behavior 5*(1), 170–181.

Taylor, A. D. and W. S. Zwicker (1995). Simple games and magic squares. *Journal of Combinatorial Theory, Series A 71*(1), 67–88.

Taylor, A. D. and W. S. Zwicker (1996). Quasi-weightings, Trading, and Desirability Relations in Simple Games. *Games and Economic Behavior 16*(2), 331–346.

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

Uno, T. (2003, July). Efficient Computation of Power Indices for Weighted Majority Games. Technical Report NII-2003-006E, National Institute of Informatics, Tokyo, Japan.

Wegener, I. (1994). The size of reduced OBDD's and optimal read-once branching programs for almost all Boolean functions. *Computers, IEEE Transactions on 43*(11), 1262–1269.

Wegener, I. (2000). *Branching Programs and Binary Decision Diagrams: Theory and Applications.* Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Winder, R. O. (1962). *Threshold Logic.* Ph. D. thesis, Department of Mathematics, Princeton University.

Zhang, R., P. Gupta, L. Zhong, and N. K. Jha (2005). Threshold network synthesis and optimization and its application to nanotechnologies. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 24*(1), 107–118.

Index