Forms of representation for simple games: sizes, conversions and equivalences

Xavier Molinero[b,1], Fabián Riquelme[a,1], Maria Serna[a,1]

[a] *Departament of Computer Science, Technical University of Catalonia, Barcelona, Spain*
[b] *Department of Applied Mathematics III. Technical University of Catalonia, Manresa, Spain*

Abstract

Simple games are cooperative games in which the benefit that a coalition may have is always binary, i.e., a coalition may either win or loose. This paper surveys different forms of representation of simple games, and those for some of their subfamilies like *regular games* and *weighted games*. We analyze the forms of representations that have been proposed in the literature based on different data structures for sets of sets. We provide bounds on the computational resources needed to transform a game from one form of representation to another one. This includes the study of the problem of enumerating the fundamental families of coalitions of a simple game. In particular we prove that several changes of representation that require exponential time can be solved with polynomial-delay and highlight some open problems.

Keywords: Simple games, Regular games, Weighted games, Forms of representation, Computational complexity

1. Introduction

Cooperative game theory constitutes a branch of Game Theory where a group of players forming a *coalition* can achieve a common benefit, thus providing a setting for the analysis of cooperative behavior. In this survey we focus on the subclass of *simple games*. Simple games are cooperative games in which the benefit that a coalition may have is always binary, i.e., a coalition may be *winning* or *losing*, depending on whether the players in the coalition are able to benefit themselves from the game by achieving together some goal.

Simple games have a huge relevance in mathematics and to some extent in computer science, being used to solve and represent problems arising in voting theory, logic and threshold logic, circuit complexity, computational complexity theory, artificial intelligence, geometry, linear programming, etc. [92, 30]. They are closely related with other mathematical and computational structures, such as hypergraphs, monotone Boolean functions, Sperner families, distributive lattices, etc. In the preface of [92] we can read:

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

Although a lot of research in Computer Science has been devoted to cooperative game theory (see [25] and references herein) the area of simple games has been less explored. Much of the research on simple games focuses on studying what conditions a simple game should have in order to meet some property or set of properties. Recently, some researchers in computer science have begun to consider the computational complexity of deciding if a simple game meets those properties or not [5, 7, 28, 41, 102, 71]. Several of these questions have been satisfactorily answered either by providing efficient algorithms or proving hardness for the problem. Of course, the way in which the game is represented and given as an input to a program is crucial for this complexity analysis.

For instance, if we want to implement an algorithm to decide if a given simple game is *decisive* (*self-dual* or *zero-sum*) its complexity depends largely on the way in which the input game is given. Thus, if the game is given in *extensive winning form* by a listing of all winning coalitions, the decision problem is in P, i.e., it is solvable in

Email addresses: xavier.molinero@upc.edu (Xavier Molinero), farisori@cs.upc.edu (Fabián Riquelme), mjserna@cs.upc.edu (Maria Serna)

Simple games	
EWF	Extensive or explicit Winning Form
ELF	Extensive or explicit Losing Form
MWF	Extensive or explicit Minimal Winning Form
MLF	Extensive or explicit Maximal Losing Form
PCBF	Partially Condensed Binary Tree Form
BDDF	Binary Decision Diagram Form
VWRF	Vector Weighted Representation Form
coVWRF	co-vector Weighted Representation Form
Regular games	
FCBF	Fully Condensed Binary Tree Form
SWF	Shift-minimal Winning Form
Weighted games	
WRF	Weighted Representation Form

Table 1: Forms of representation considered in this paper.

Output → Input ↓	EWF	MWF	ELF	MLF
EWF	–	P	s–EXP	P
MWF	s-EXP ***Pd***	–	s-EXP	s-EXP
ELF	s-EXP	P	–	P
MLF	s-EXP	s-EXP	s-EXP ***Pd***	–

Table 2: Computational complexity of the conversion problem from the row form to the column form, for representations of simple games based on explicit descriptions of set families.

polynomial time. However, if the game is given in *minimal winning form* by a listing of all *minimal* winning coalitions, the problem is in QP, i.e., solvable in *quasi-polynomial time*, being still an open question if it is in P [34, 86]. At last, if the game is a *weighted game*, an important subclass of simple games studied in several contexts [93, 52, 73, 90], given in *weighted representation*, then the decisive problem is *coNP*-complete [41].

On the other hand, the size of a simple game in *extensive winning form*, i.e., the amount of bits needed in order to write down the family of its winning coalitions, could easily grow exponentially as a function of the number of players. Therefore, it is relevant to ask which forms of representation are the more appropriate when we face a computational problem. Both the feasibility of the representation as well as the computational complexity of the problem, are aspects that must be considered; and the first of these aspects has direct implications about the second one. Our aim is to provide an easy way to access information on the different forms of representing a simple game together with an analysis of their relationships from a computational point of view.

All the forms of representation described above, as well as other usual ones based on data structures for representing sets of sets, will be described and analyzed in this paper. A summary of the forms of representation considered here is given in Table 1. Winning coalitions of a simple game can also be thought as the satisfying assignments of a monotone Boolean function. We do not consider such implicit representation. The interested reader can find a study of monotone Boolean functions and related representation considerations in [95, 26] . Another succinct form of representation for regular games uses an invariant $(\vec{v}, \mathcal{M})$ where $\vec{v}$ is a set of players' classes and $\mathcal{M}$ is a matrix that considers some special type of winning coalitions [24]. This type of representation is less usual in computer science approaches. We leave as an open problem the study of the conversion problems for this representation form.

Besides describing the different forms of representation, we are interested in highlighting which forms of repre-

Output → Input ↓	EWF	MWF	PCBF	BDDF
EWF	–	P	P	P
MWF	s-EXP ***Pd***	–	P	***P***
PCBF	s-EXP ***Pd***	P	–	***P***
BDDF	s-EXP	s-EXP	s-EXP	–

Table 3: Computational complexity of the conversion problem for representations of simple games based on variants of binary trees and winning coalitions.

Output → Input ↓	EWF	MWF	PCBF	BDDF	FCBF	SWF
FCBF	s-EXP ***Pd***	P	P	***P***	–	P
SWF	s-EXP	s-EXP ***Pd***	s-EXP	s-EXP?	s-EXP	–

Table 4: Computational complexity of the conversion problem from the row form to the column form for representations of regular games.

sentation are equivalent, in the sense that the bounds of their sizes are polynomially related. We are also interested in the computational complexity of the *conversion problem*: computing a representation of a simple game given another representation. For equivalent forms of representation the objective is to determine if the conversion problem (in both directions) can or cannot be solved in polynomial time. For non-equivalent forms we know that at least one of the conversion problems could not be solved in polynomial time. Therefore, in these cases, we seek enumeration algorithms with polynomial-delay. Tables 2–5 provide an overview of the results of the conversion problem for the different forms of representation for simple games considered in this paper. Results in bold face are new and question marks correspond to open problems. P indicates that the problems can be solved in polynomial time. s-EXP, called "strict-exponential", denotes the fact that the problem can be solved in exponential time, but can not be solved in sub-exponential time. Finally, Pd indicates that the enumeration problem can be solved with polynomial-delay. For subfamilies of simple games the results are restricted to games in the subfamily. The data structures that constitute a representation form are those for which the problem of deciding if a simple game is in the family can be solved in polynomial time. Observe that the conversion problem includes the problem of enumerating the minimal winning coalitions. This family of coalitions is required to compute several power indices (see, e.g., [49, 36, 12]). Therefore, efficient algorithms to enumerate them are of particular relevance for any form of representation. For each of the conversion problem considered in Tables 4 and 5, the reversed conversion problem can be solved trivially in polynomial time.

The paper is organized as follows. In Section 2 we introduce the basic definitions and concepts related to simple games as well as the computational problems to be studied later. Section 3 reviews several usual forms of representation based on explicit representations of set families and characteristic vectors. We survey several complexity results for the conversion and enumeration problems. Section 4 surveys forms of representations based on binary trees and variations. Section 5 focuses in particular data structures representing regular games. Section 6 is devoted to weighted games. Finally, in Section 7 we state some concluding remarks and open problems.

2. Simple games and representation forms

In this section we introduce the basic definitions and concepts for set families and simple games. We use standard terminology and notation taken from [92]. For a finite set N, $\mathcal{P}(N)$ denotes its power set and n its cardinality. A *set family* $\mathcal{S} \subseteq \mathcal{P}(N)$ is *monotonic* if, for $X \in \mathcal{S}$ and $Z \subseteq N$ with $X \subseteq Z$, $Z \in \mathcal{S}$. A *hypergraph*, defined over a ground

Output → Input ↓	EWF	MWF	PCBF	BDDF	FCBF	SWF
WRF	*s-EXP* ***Pd***	*s-EXP* ***Pd***	*s-EXP*	*s-EXP*	*s-EXP*	*s-EXP*

Table 5: Computational complexity of the conversion problem for representations of weighted games.

finite set N, is a pair $(N, \mathcal{S})$ where $\mathcal{S} \subseteq \mathcal{P}(N)$. The elements $X \in \mathcal{S}$ are called *hyperedges*. A hypergraph is *monotonic* when its set of hyperedges is monotonic.

A *simple game* Γ is a monotonic hypergraph $(N, \mathcal{W})$. The elements in N are called *players*, the subsets of N are called *coalitions* and the elements of $\mathcal{W}$ are called *winning coalitions*. The complement of $\mathcal{W}$, $\mathcal{L} = \mathcal{P}(N)\backslash\mathcal{W} = \{X \subseteq N \mid X \notin \mathcal{W}\}$, is the set of *losing coalitions*. We use $\mathcal{W}(\Gamma)$ and $\mathcal{L}(\Gamma)$ to denote, respectively, the sets of winning and losing coalitions of a simple game Γ, and simply $\mathcal{W}$ or $\mathcal{L}$ when there is no risk of ambiguity.

Simple games were first defined by von Neumann and Morgenstern [93]. However, they considered a more restricted class of games, which nowadays are known as *strong games*, i.e., simple games such that, for $Y \subseteq N$, if $Y \in \mathcal{L}$, then $N\backslash Y = \{i \in N \mid i \notin Y\} \in \mathcal{W}$. According to Isbell [53], the current definition was given by Gillies [43] under the name of *pseudogames*.

Two relevant subsets of coalitions are the *minimal winning* and the *maximal losing* coalitions. Here minimality and maximality are defined with respect to the inclusion ordering. A coalition $X \subseteq \mathcal{W}$ is minimal if, there exists $i \in X$ such that $X \setminus \{i\}$ is a losing coalition. Symmetrically, a coalition $Y \subseteq \mathcal{L}$ is maximal if there exists $j \notin Y$ such that $Y \cup \{j\}$ is winning. Observe that as the set of winning coalitions is monotonic, $\mathcal{W}(\Gamma)$ is defined uniquely by $\mathcal{L}(\Gamma)$, the set of losing coalitions, or by $\mathcal{W}^m(\Gamma)$, the set of minimal winning coalitions, or by $\mathcal{L}^M(\Gamma)$, the set of maximal losing coalitions.

The relationship between (minimal) winning and (maximal) losing coalitions play a fundamental role in the *self-duality problem* [83] of monotone Boolean functions and hypergraphs [30]. In the seminal work on game theory by von Neumann and Morgenstern [93] this problem corresponds to decide whether a simple game is constant-sum, zero-sum or decisive. Now we introduce the duality of simple games.

Let $\Gamma = (N, \mathcal{W})$ be a simple game, $X \subseteq N$ is a *blocking* coalition if $N\backslash X \in \mathcal{L}(\Gamma)$. The *dual* of Γ is the simple game Γ^d such that $\mathcal{W}(\Gamma^d) = \{X \subseteq N \mid X \text{ is a blocking coalition of } \Gamma\}$. It is clear that $\mathcal{L}(\Gamma^d) = \{Y \subseteq N \mid N\backslash Y \in \mathcal{W}(\Gamma)\}$. Hence, the dual of a dual game is the original game, i.e., $(\Gamma^d)^d = \Gamma$.

Duality provides an additional way to define games in a unique way, because the set of blocking coalitions of the dual game determines uniquely the set of winning coalitions. Observe that this subfamily of coalitions has the same size as the set of losing coalitions of the original game.

Now we turn our attention to the ways in which a simple game can be represented as the input to a problem.

Definition 1. Let $\mathcal{G}$ be a class of simple games, a *form of representation* of $\mathcal{G}$ is a finite data structure which allows to represent each simple game $\Gamma \in \mathcal{G}$, as well as verifying in polynomial time if a given instance of the data structure represents a simple game Γ in $\mathcal{G}$ or not.

We use the notation $F(\Gamma)$ to denote a representation of simple game Γ in form F. As usual, $|F(\Gamma)|$ denotes the *size* of $F(\Gamma)$, that is the amount of bits needed to write down the data structure representing Γ.

Definition 2. Let F_1 and F_2 be two forms of representation for a class of simple games $\mathcal{G}$, the *conversion problem* from F_1 to F_2, denoted by $F_1 \rightsquigarrow F_2$, is the following: Given a simple game Γ in F_1 form, computing a representation of Γ in F_2 form, i.e., given $F_1(\Gamma)$ compute $F_2(\Gamma)$.

In general, the relative sizes of the different representations of a simple game show the impossibility of having polynomial time algorithms for the conversion problem between some forms of representation. Nevertheless, generating explicitly a given form of representation from another one is an interesting problem in combinatorics and computer science [56]. Several representation forms presented in this survey are associated with set families. In those cases where the conversion problem requires exponential time, we turn our attention to enumeration algorithms and provide, when possible, polynomial-delay algorithms.

We state now some relevant definitions about the complexity of enumeration algorithms taken from [56]. Let us consider that we have an input I and we want to obtain the output $O = \{o_1, o_2, \ldots, o_{|O|}\}$. An algorithm $\mathcal{A}$ runs in *polynomial total time* if the time required by $\mathcal{A}$ to print O is upper bounded by a polynomial in $|I|$ and $|O|$. $\mathcal{A}$ runs in *incremental polynomial time* if, given I and $O_i = \{o_1, o_2, \ldots, o_i\}$, computing o_{i+1} or determining that such element does not exists can be done in polynomial time in the combined sizes of I and O_i. Finally, $\mathcal{A}$ runs with *polynomial-delay* if, given O_i, finding o_{i+1} or deciding that such element does not exists can be done in time polynomial in $|I|$. The third condition implies the second one, and the second one implies the first one. In the context of this paper our input will be a simple game, given in some form of representation, and the output will be some of the fundamental set families defining the game.

It is well known that simple games can be described by monotone Boolean functions [92], and therefore by several kinds of logical formulas [30]. A *monotone Boolean function* or *positive Boolean function* [64] is a binary function $f : \{0,1\}^n \to \{0,1\}$ such that, for $v, w \in \{0,1\}^n$ with $v \leq w$, $f(v) \leq f(w)$. Therefore, vectors x with $f(x) = 1$ represent the winning coalitions of the simple game, while vectors x with $f(x) = 0$ represent the losing coalitions. As mentioned before, we do not consider the succinct implicit representation provided by monotone Boolean functions and refer the reader to any of the references on Boolean formulas [95, 26].

3. Simple games represented by explicit set families and incidence vectors

In this section, we describe several usual forms of representation for simple games corresponding to standard explicit forms of representation of set families [93]. We use matrix notation as an explicit representation of sets. This form of representation is mostly used in the context of hypergraphs and monotone Boolean functions [60, 30, 82]. Recall that a set $X \in \mathcal{P}(N)$ can be represented by an *incidence vector* $x(X) \in \{0,1\}^n$, where $x(X) = (x_1, \ldots x_n)$ with $x_i = 1$ if and only if $i \in X$, for $1 \leq i \leq n$. For an incidence vector x we use x_i or $x(i)$ to denote the i-th component of x. For comparing incidence vector we use the usual lexicographic order: for $x, y \in \{0,1\}^n$, $x \leq y$ if and only if, for each $i \in N$, $x_i \leq y_i$.

We start presenting the earliest forms of representation defined by von Neumann and Morgenstern [93].

Definition 3. A simple game Γ is represented in:

- *(Extensive or explicit) winning form (EWF)* by a pair $(N, \mathcal{W})$. N is its set of players and $\mathcal{W}$ is the set of winning coalitions of Γ.
- *(Extensive or explicit) minimal winning form (MWF)* by a pair $(N, \mathcal{W}^m)$. N is its set of players and $\mathcal{W}^m$ is the set of minimal winning coalitions of Γ.

Observe that both forms of representations are valid since, given a set family, we can check in polynomial time whether it is monotonic or whether it is minimal. Indeed, to check minimality, we just have to test whether no set is a proper subset of another.

We can represent a simple game by an *incidence matrix* with a column for each player and rows formed by the incidence vectors of the winning (or minimal winning) coalitions.

Example 1. The game $\Gamma = (N, \mathcal{W})$, with $N = \{a, b, c, d, e\}$ and $\mathcal{W} = \{\{a, b, c, d, e\}, \{a, b, c, d\}, \{a, b, c, e\}, \{a, b, d, e\}, \{a, b, e\}, \{a, c, d, e\}, \{a, c, d\}, \{a, c, e\}, \{a, d, e\}, \{b, c, d, e\}, \{b, c, d\}, \{b, c, e\}, \{b, d, e\}, \{c, d, e\}, \{c, e\}, \{d, e\}\}$ can be represented in MWF as the pair $(N, \mathcal{W}^m)$ with $\mathcal{W}^m = \{\{a, b, e\}, \{a, c, d\}, \{b, c, d\}, \{c, e\}, \{d, e\}\}$. The incidence matrix of $(N, \mathcal{W}^m)$ is the following:

N	$abcde$
	00011
	00101
$\mathcal{W}^m$	01110
	10110
	11001

Assuming an incidence matrix representation, we have that $|\text{EWF}(\Gamma)| = n \cdot |\mathcal{W}|$ and $|\text{MWF}(\Gamma)| = n \cdot |\mathcal{W}^m|$. Since $\mathcal{W}^m(\Gamma) \subseteq \mathcal{W}(\Gamma)$, $|\text{EWF}(\Gamma)| \leq |\text{MWF}(\Gamma)|$. The following example provides a simple game for which $|\text{MWF}(\Gamma)|$ is exponentially smaller than $|\text{EWF}(\Gamma)|$.

Example 2. Let Γ be a simple game which contains the *empty coalition* as winning coalition, i.e., such that all players are *dummies* [93]. Hence, $\emptyset$ is the unique minimal winning coalition, thus $|\mathcal{W}^m| = 1$. However, the number of winning coalitions is $|\mathcal{W}| = 2^n$.

From the previous example, we have the following result (see also [41]).

Lemma 1. *The problem MWF$\rightsquigarrow$EWF can be solved in exponential time but it can not be solved in sub-exponential time. The problem EWF$\rightsquigarrow$MWF can be solved in polynomial time.*

Besides EWF and MWF, there are other two classical forms of extensive representations, based on losing coalitions. Those representation forms were also defined by von Neumann and Morgenstern [93].

Definition 4. A simple game Γ is represented in:

- *(Extensive or explicit) losing form (ELF)* by a pair $(N, \mathcal{L})$. N is its set of players and $\mathcal{L}$ is the set of losing coalitions of Γ.
- *(Extensive or explicit) maximal losing form (MLF)* by a pair $(N, \mathcal{L}^M)$. N is its set of players and $\mathcal{L}^M$ is the set of maximal losing coalitions of Γ.

Observe that both forms of representations are valid since, given a set family, we can check in polynomial time whether its complement is monotonic or whether it is maximal. Indeed, to check maximality, we just have to test whether no set is a proper superset of another. Note that the respective sizes of a simple game Γ are $|\text{ELF}(\Gamma)| = n \cdot |\mathcal{L}|$ and $|\text{MLF}(\Gamma)| = n \cdot |\mathcal{L}^M|$.

The conversion problem among representations based on losing coalitions and those based on winning coalitions was studied in [41]. There it was shown that EWF$\rightsquigarrow$MLF, ELF$\rightsquigarrow$MWF and ELF$\rightsquigarrow$MLF can be solved in polynomial time. The remaining cases considered here can be solved in exponential time, but can not be solved in sub-exponential time [41]. The polynomial time algorithms come from the monotonicity of simple games. The exponential time requirement is related to the respective size of the representations, as in Lemma 1.

Lemma 2 ([41])**.** *The problems EWF$\rightsquigarrow$MLF, ELF$\rightsquigarrow$MLF and ELF $\rightsquigarrow$MWF can be solved in polynomial time. The problems EWF$\rightsquigarrow$ELF, MWF $\rightsquigarrow$ELF, MWF $\rightsquigarrow$MLF, ELF $\rightsquigarrow$EWF, MLF$\rightsquigarrow$EWF, MLF$\rightsquigarrow$MWF and MLF$\rightsquigarrow$ELF can be solved in exponential time but they can not be solved in sub-exponential time.*

Moreover, based on a result of reliability functions from [10], Aziz proved that the conversion problem MWF $\rightsquigarrow$ EWF is #*P*-complete [3].

Note that as $\mathcal{W} \cup \mathcal{L} = \mathcal{P}(N)$, there are three possibilities for a given simple game which belongs to a subclass:

1. Both $|\mathcal{W}|$ and $|\mathcal{L}|$ grow exponentially in terms of n.
2. $|\mathcal{W}|$ is polynomially bounded and $|\mathcal{L}|$ grows exponentially.
3. $|\mathcal{W}|$ grows exponentially and $|\mathcal{L}|$ is polynomially bounded.

When the first possibility happens, we need a more accurate study to understand the differences between the two forms of representation. When the second possibility happens, it is most succinct to represent the game in EWF than in ELF. Finally, when the third condition succeeds, the dual game verifies the second condition, so the dual game can be represented more succinctly in EWF. Note that, according to the definition of duality, $|\text{EWF}(\Gamma)| = |\text{ELF}(\Gamma^d)|$. Although duality allows us to consider other forms of representation for Γ, as any of the representations of Γ^d represent uniquely Γ, we will not deal with them in this paper.

We analyze now the computational complexity of the enumeration problem, for some of the conversion problems that require exponential time.

Lemma 3. *MWF$\rightsquigarrow$EWF can be solved with polynomial-delay.*

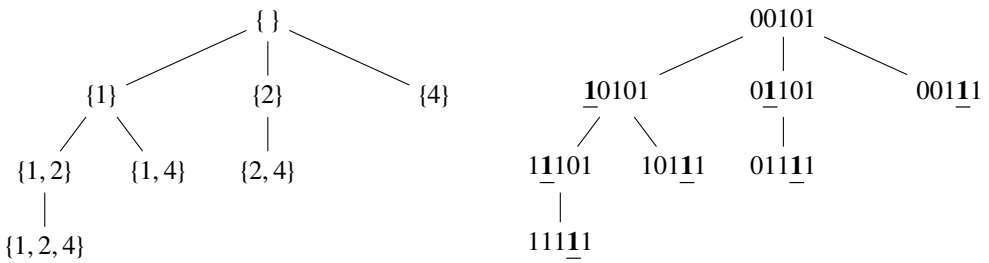

Figure 1: Backtrack tree for $\{1, 2, 4\}$ and backtrack tree for the enumeration without repetitions of all the winning coalitions that contain 00101.

Algorithm 1 GenerateEWFfromMWF

Input: A simple game Γ in MWF with $\mathcal{W}^m = \{X_1, \ldots, X_m\}$ sorted in lexicographic order.
Output: Γ in EWF.

```
1:  GENERATE(X, R, i)
2:    for all j ∈ R in increasing order
3:      X = X ∪ {j};
4:      if for all k > i, X_k ⊄ X
5:        print X;
6:        R = N_j \ X;
7:        GENERATE(X, R, i);
8:        X = X \ {j};
9:  {main}
10: for i = 1, ..., m do
11:   print X_i;
12:   R = N \ X_i;
13:   GENERATE(X_i, R, i);
```

Proof. First note that every minimal winning coalition is a winning coalition, so they have to be printed. Let $(N, \mathcal{W}^m)$ be a simple game, we assume that $N = \{1, \ldots, n\}$ and $\mathcal{W}^m = \{X_1, \ldots, X_m\}$ according to increasing lexicographical order.

In order to enumerate all the winning coalitions without repetitions, our algorithm enumerates only a subset of the $2^{n-|X|} - 1$ winning coalitions that contain $X \in \mathcal{W}^m$ (see Algorithm 1).

The algorithm is a branch and cut algorithm that uses the usual backtrack tree providing an enumeration of the subsets of a set without repetitions. Recall that in such a tree a node has as children the superset obtained by adding one candidate element. The set of candidates is formed by those elements that are not in the current subset and that are posterior to all the elements in the current subset. An example of the backtracking tree for a set with four elements is given in Figure 1. We consider such a tree for every given winning coalition.

We first sort the set of minimal winning coalitions in increasing lexicographical order. Then, for each minimal winning coalition X_i, we perform a traversal of the backtrack tree corresponding to all the subsets of $N \setminus X_i$ as described before. On the traversal our algorithm backtracks whenever it reaches a set X that is a superset of X_j, for $j > i$. Thus our algorithm prints, for any minimal winning coalition X_i, all the coalition that are supersets of X_i but not supersets of any minimal winning coalition X_j with $i < j$. The first property guarantees that the winning coalitions are printed without repetitions. Furthermore, monotonicity guarantees that the algorithm prints all the winning coalitions.

Finally, take into account that in any of the backtracking trees constructed in an execution of Algorithm 1, the height is at most n. Therefore, any backtrack path has length bounded by n. In consequence, the number of steps performed by the algorithm between the printing of a winning coalition and the next one is $O(n)$ and the claim follows. □

Example 3. Consider the simple game $(N, \mathcal{W}^m)$ of Example 1. Figure 2 shows the enumeration without repetition of the winning coalitions of the game. They are printed in the following order: 00011, 10011, 01011, 00101, 10101,

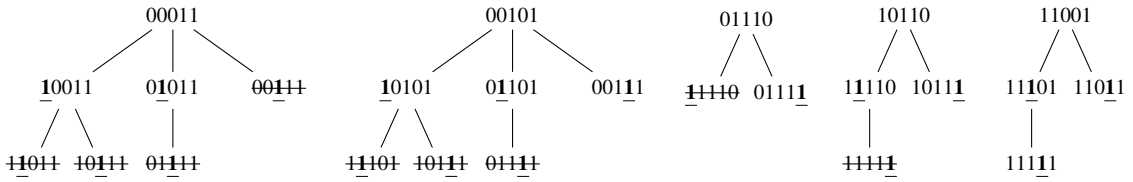

Figure 2: Computation of MWF⇝EWF for the simple game of Example 1.

Algorithm 2 GenerateELFfromMLF

Input: A simple game Γ in MLF with $\mathcal{L}^M = \{X_1, \ldots, X_m\}$ sorted in lexicographic order.
Output: Γ in ELF.

```
1:  GENERATE(X, R, i)
2:    for all j ∈ R
3:      X = X \ {j};
4:      if for all k > i, X_k ⊄ X
5:        print X;
6:        R = X ∩ N_{j+1};
7:        GENERATE(X, R, i);
8:        X = X ∪ {j};
9:  {main}
10: for i = 1, ..., m do
11:   print X_i;
12:   R = X_i;
13:   GENERATE(X_i, R).
```

01101, 00111, 01110, 01111, 10110, 11110, 10111, 11001, 11101, 11111, 11011.

For representations forms based on sets of losing coalitions, we get an equivalent result.

Lemma 4. *MLF⇝ELF can be solved with polynomial-delay.*

Proof. The proof is similar to that of Lemma 3 and corresponds to Algorithm 2. Analogously, our algorithm starts by sorting in lexicographic order the set of maximal losing coalitions. Algorithm 2 backtracks over the subsets of each maximal losing coalition. As before, to avoid repetitions, when dealing with a maximal losing coalition X_i, we backtrack when the algorithm reaches a set that is also a subset of X_j, for $j > i$. Again, all the backtracking trees have height at most n and thus Algorithm 2 works with polynomial-delay. □

It remains open to determine whether the conversions MWF⇝ELF, MWF⇝MLF, MLF⇝EWF, and MLF⇝MWF can be solved with polynomial-delay.

4. Binary tree representations

In this section, we review several usual forms of representation for simple games using different families of (extended) binary trees [64]. There are several forms of representation based on directed binary trees [75]. Recall that a (*rooted*) *binary tree* is a data structure in which each node has at most two child nodes, usually distinguished as "left" and "right" or by labels 0/1 on the corresponding arc. Nodes with children are called *inner nodes*. Nodes without children are called *terminal nodes*, also known as *leaf nodes*, *outer nodes* or *external nodes*. The *root node* is the ancestor of all nodes. Any node in the data structure can be reached by starting at the root node and repeatedly following pointers to either the left or right child. A tree which does not have any node other than the root node is called a null tree. In a binary tree, the degree of every node is at most two. In some cases the terminal nodes are labeled. Recall that a tree with n nodes has $n - 1$ edges. The *size* of a binary tree is its number of nodes.

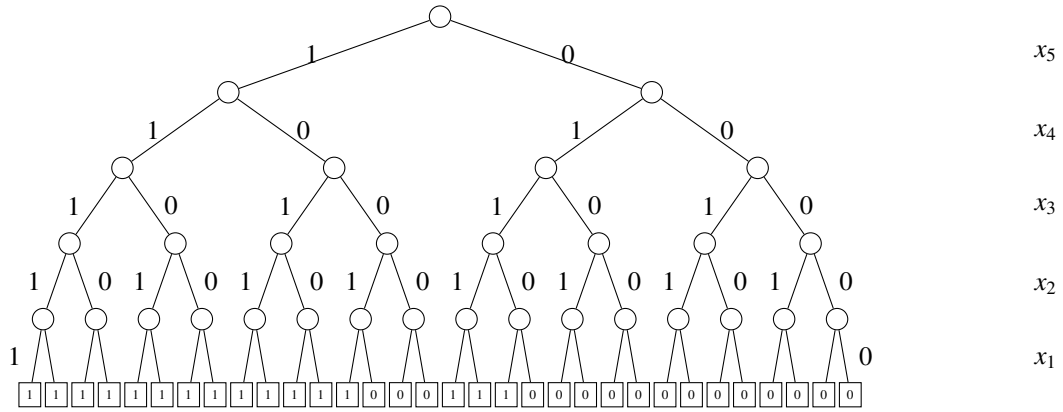

Figure 3: Complete binary tree $B_c(\mathcal{W})$ representing the winning coalitions for the simple game Γ of Example 1. The missing labels in the last level of edges can be deduced from the other levels.

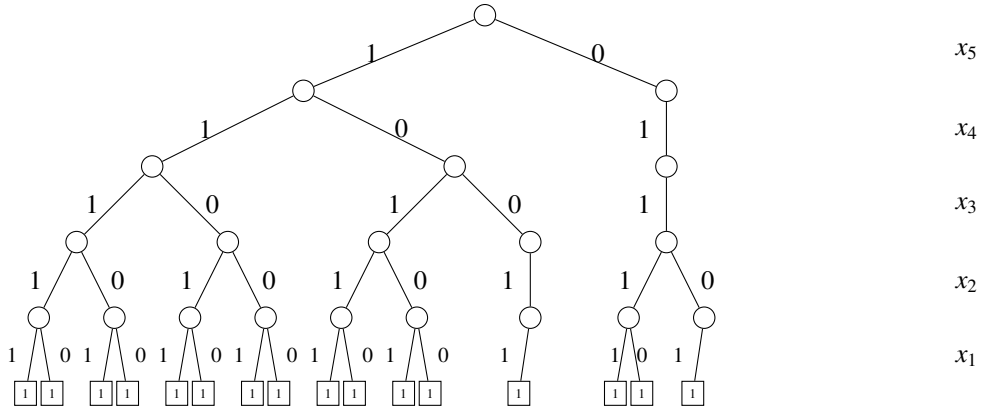

Figure 4: Binary tree $B(\mathcal{W})$ representing the winning coalitions for the simple game Γ of Example 1.

As usual, we assume a lexicographic order on the set of players. Any set family $\mathcal{F}$ can be represented by a binary tree in different ways. The simplest (and the most costly) such form of representation uses a *complete binary tree* B_c with height n. In such a tree B_c, for any node $t \in B_c$ at depth j, the left edge (respectively, right edge) from t represents that $x_{n-j} = 1$ (respectively, $x_{n-j} = 0$). Terminal nodes are labeled with either 0 or 1. Thus, a terminal node $t \in B_c$ at depth n represents a vector with n components corresponding to the edge labels found in the path from the root to t. Those sets corresponding to paths ending in a terminal node with label 1 belong to the represented family.

Example 4. Figure 3 illustrates the complete binary tree for the set of winning coalitions of the simple game given in Example 1.

Note that every complete binary tree representing a set family has 2^n terminal nodes and $2^n - 1$ inner nodes, including the root node. Thus, its size depends on the number of players n and not on the number of sets in the represented family. Therefore, complete binary trees are not the best form of representation based on trees. However, we include them here in order to introduce other variants of binary trees.

A first variation removes the vector components represented by the terminal nodes with label 0. Figure 4 represents a non-complete binary tree for the set of winning coalitions of the simple game of Example 1. We keep the edge labels, with the same meaning, i.e., an edge from a node at depth j to a node at depth $j + 1$ labeled by $l \in \{0, 1\}$ represents the fact that $x_{n-j} = l$. Terminal nodes do not have assigned labels but remain at depth n. As before, a terminal node $t \in B$ represents a vector with components resulting by the edge labels in the path from the root to t. Those sets corresponding to paths ending in a terminal node belong to the set family represented by the tree. Observe that now

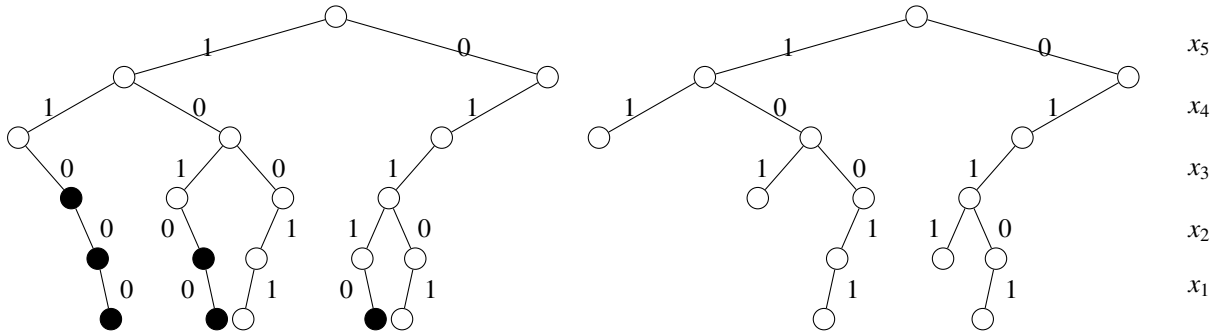

Figure 5: $B(\Gamma)$ (left) and $PCB(\Gamma)$ (right) representing the simple game Γ of Example 1. $PCB(\Gamma)$ is obtained from $B(\Gamma)$, by removing the marked nodes.

the size of the tree is polynomially related to the number of sets in the family. This binary tree data structure has been used in relation with simple games in the context of monotone Boolean functions as a representation of the set of minimal winning coalitions [64].

From the computational point of view, any binary tree representation of one of the fundamental sets describing a simple game is related to the corresponding representation form in the same way. In what follows, as it has been done in the literature, we only consider binary tree representations of the subset of minimal winning coalitions.

Definition 5. A simple game Γ is given in *binary tree form (BF)* by a binary tree representing the set $\mathcal{W}^m(\Gamma)$.

We use the notation $B(\Gamma)$ to denote a simple game Γ given in binary tree form. Observe that we can check in polynomial time whether a set belongs or not to the set represented by a binary tree. We can check in polynomial time the minimality of the represented set. Thus, the binary tree form is a valid representation for simple games. Figure 5 at left depictes the binary tree $B(\Gamma)$ for the game $\Gamma = (N, \mathcal{W}^m)$ given in Example 1.

In the following we introduce two other forms of representation based on binary trees. Whereas the first one reduces the size of the binary trees, the second one, based on binary decision diagrams, reduces the size of a complete binary tree representation. In both cases the data structure allows to check in polynomial time if a given set belongs to the represented family. Therefore, both data structure are valid representation forms for simple games.

Makino [64] uses a more succinct data structure based on binary trees: A *partially condensed binary tree* (*PCB*) for a set family $\mathcal{F}$ is the subgraph of $B(\mathcal{F})$ which is obtained after removing recursively all the leaves whose parent has no edge labeled 1, i.e., whose parent has no left-child.

Definition 6. A simple game Γ is given in *partially condensed binary tree form (PCBF)* by the *PCB* obtained from $B(\Gamma)$.

We denote by $PCB(\Gamma)$ the partially condensed binary tree representation of Γ. Recall that the tree provides a representation of $\mathcal{W}^m(\Gamma)$.

Example 5. The right tree in Figure 5 represents $PCB(\Gamma)$ for the simple game given in Example 1. This tree is obtained from $B(\Gamma)$, the left tree in the same figure.

Our last representation form is based on *binary decision diagrams*, also called *branching programs* [96, 69]. Binary decision diagrams were introduced in the context of monotonic functions by Lee [62]. They were studied in depth by Akers [1] and Boute [18]. The representation form considered in this paper provides a representation of a set family, in a similar way as complete binary trees, but more succinctly. Thus representing minimal winning coalitions by paths leading to a 1 value.

A *binary decision diagram* (*BDD*) is a directed, acyclic and labeled graph with decision nodes and two terminal nodes called 0-terminal and 1-terminal. A BDD is *ordered* (*OBDD*) if the players appear in the same order on all paths from the root; and is *reduced* (*RBDD*) if all its isomorphic subgraphs are merged and it does not have any node with

Algorithm 3 GeneratingBDDfromBT

Input: A binary tree representing a subset set S.
Output: A BDD representing S.
1: Merging duplicate terminal nodes that share the same label (0 or 1).
2: For each level from the bottom to the top:
3: **loop**
4: Merging duplicate inner nodes whose left-child and right-child are connected with the same node.
5: Removing inner nodes with the same left-child and right-child.
6: **end loop**

two isomorphic children. A binary decision diagram represents the set family formed by all the vectors extracted from paths ending in the 1-terminal. For each given ordering on N, there is a unique *reduced and ordered BDD* (*ROBDD*) representing a set family [21, 22]. That is why a BDD is usually the ROBDD corresponding to the lexicographic order.

Definition 7. A simple game Γ is given in *binary decision diagram form (BDDF)* by a BDD representing $\mathcal{W}^m(\Gamma)$.

Given a simple game Γ, we can compute $BDD(\Gamma)$ from $B(\Gamma)$. This can be done through Algorithm 3 [22]. After each step of the procedure, it is necessary to redirect all incoming arcs to the corresponding new nodes. Steps 4 and 5 must be repeated as many times as necessary. In general, starting with the level of the first component x_1, and ending with the level of the n-th component x_n. Usually left-children are denoted by a solid edge, whereas the right-children are denoted by a dashed edge. In what follows, we use a double edge to represent a node in which the left-child and the right-child are the same.

We can also consider the OBDD obtained by a procedure in which the rule implemented in step 5 of Algorithm 3 is not used. In such a BDD every path from the root to a terminal node has length $n+1$. The so obtained BDD is called a *quasi-reduced binary decision diagram (QOBDD)* [11, 16]. Note that for a fixed variable ordering, both QOBDD and ROBDD are canonical representations. If we consider a QOBDD and its corresponding ROBDD (sharing inner nodes), checking whether both BDDs are equivalent is trivial in the sense of computational complexity [48]. On the other hand, let Γ_1 and Γ_2 be two simple games, such that the sizes of their respective BDDs are r and s, according to [17], determining wheter both BDDs are equal requires $O(\min\{r, s\})$ steps.

Example 6. We continue with the simple game of Example 1. Figure 6 illustrates the BDD obtained from the complete binary tree of Figure 3, after applying Algorithm 3.

As we have mention before, for a simple game Γ, we could construct, instead of $BDD(\mathcal{W}(\Gamma))$, any of the BDDs associated to the other fundamental set families, $BDD(\mathcal{L}(\Gamma))$, $BDD(\mathcal{W}^m(\Gamma))$, or $BDD(\mathcal{L}^M(\Gamma))$. However, the differences between them and their corresponding forms of representation are similar to those for $BDD(\mathcal{W}(\Gamma))$ and MWF. For the particular case of regular games, as we shall see in Section 5, some of those forms of representation are of equivalent size.

4.1. Representation sizes and the conversion problem

Now we summarize several known results related to the size of the forms of representation defined before. We introduce some new results and we use them to analyze the computational complexity of the corresponding conversion problems. As the representation based on binary trees and variants are based on the winning coalitions, we consider only the conversion problems with respect to EWF and MWF.

First of all, it is necessary to remark that both the size of the PCBs and BDDs (ROBDDs) depend on the ordering of players. While a specific ordering could lead, in terms of the number of players, to a very small size, another ordering might lead to exponential size. It is known that the problem of finding the best variable ordering for a BDD is *NP*-hard [14]. There exist simple games Γ for which $|BDD(\Gamma)|$ grows exponentially in terms of n, independently of the order of the variables. For instance, consider the Theorem 4 of [50], which shows a subclass of simple games called *weighted games*, for which $|BDD(\Gamma)| \in \Omega(2^{\sqrt{n}/2})$. Additionally, there exists a known result by Wegener which

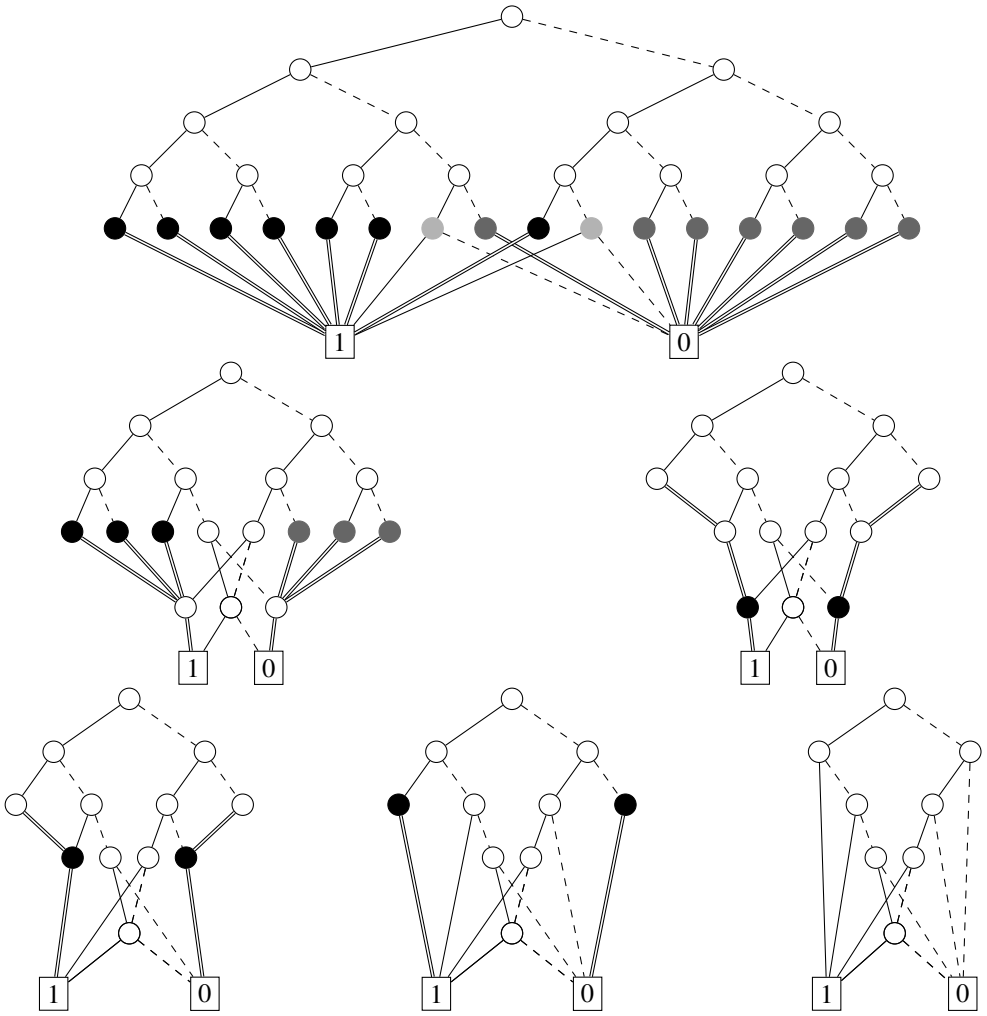

Figure 6: $BDD(\Gamma)$ obtained from the given binary tree $B(\Gamma)$ for the game Γ of Example 1. Marked nodes are merged or removed in the next step.

says that almost all QOBDDs for general Boolean functions, not only the monotone ones, have size $2^n/(2n)$ [97]. The same author proved that QOBDDs are at most a factor $n + 1$ larger than their corresponding ROBDDs [98]. The relevance of polynomial size BDDs has motivated Ishiura and Yajima to define the *PolyBDD* family [54]. This class, however, has no explicit characterization, so it will not be discussed here. An interesting open question is to characterize the simple games that can be represented by BDDs with polynomial size. In this paper, we use always a given order of variables and do not care about the best ordering.

It is easy to see that a simple game in MWF, PCBF or BDDF might require smaller size than in EWF. To see this, just consider the simple game Γ of Example 2, where $\mathcal{W}^m = \{\emptyset\}$. In this case, $|\mathcal{W}^m| = |PCB(\Gamma)| = |BDD(\Gamma)| = 1$, but $|\mathcal{W}| = 2^n$ is an exponential amount in terms of n.

There are simple games whose representation in BDDF requires less space than in MWF. For instance, the simple game given in Example 12 in Section 6.2. For a simple game with $n > 1$, to represent a minimal winning coalition in $PCB(\Gamma)$ it is always required to have at least two nodes. Therefore, the game defined in Example 12 also shows that there exist simple games for which their representation in BDDF grows exponentially in terms of their representation in PCBF.

For a simple gample Γ with n players, $|BDD(\Gamma)| \leq |PCB(\Gamma)| \leq n|\mathcal{W}^m| \leq n|\mathcal{W}|$, for sufficiently large n. Those inequalities imply the following results.

Lemma 5. *PCBF⇝EWF, BDDF⇝EWF, BDDF⇝MWF and BDDF⇝PCBF can be solved in exponential time but they can not be solved in sub-exponential time.*

The following result establishes those conversion problems that can be solved in polynomial time.

Lemma 6. *EWF⇝PCBF, EWF⇝BDDF, MWF⇝PCBF, MWF⇝BDDF, PCBF⇝MWF and PCBF⇝BDDF can be solved in polynomial time.*

Proof. Polynomiality of MWF⇝PCBF comes from [64]: Since the number of nodes of $PCB(\Gamma)$ is at most $O(n|\mathcal{W}^m|)$, then MWF⇝PCBF can be computed in $O(n|\mathcal{W}^m|)$ time.

For PCBF⇝MWF, observe that the number of paths from the root to each terminal node is $|\mathcal{W}^m|$. We can perform a breadth-first traversal of $PCB(\Gamma)$ in $O(|PCB(\Gamma)|)$ time. We need at most n steps per path to add the ending zeros when needed. Thus, we can compute PCBF⇝MWF in $O(n \cdot |PCB(\Gamma)|)$ time.

Polynomiality of EWF⇝PCBF and EWF⇝BDDF follow from the above.

For MWF⇝BDDF, we have $|BDD(\Gamma)| \leq |PCB(\Gamma)| \leq n|\mathcal{W}^m|$. We can construct $B(\Gamma)$ in $O(n|\mathcal{W}^m|)$ time. Then $BDD(\Gamma)$ can be constructed using a breadth-first traversal on $B(\Gamma)$ to join each node without right-child to the 0-terminal and each leaf with the 1-terminal. Hence, as $|V(B(\Gamma))| \leq n|\mathcal{W}^m|$, MWF⇝BDDF can be computed in $O(n|\mathcal{W}^m|)$ time.

For PCBF⇝BDDF, since all the leaves of $PCB(\Gamma)$ have label 1, they can be removed and replaced by a 1-terminal node, keeping the corresponding edges. Finally, we have to connect each node without right-child to the 0-terminal merging duplicate inner nodes as in step 4 of Algorithm 3. All these steps can be computed in polynomial time. □

Note that, as MWF⇝PCBF and PCBF⇝MWF can be solved in polynomial time, the computational complexity of any problem on simple games is the same, for games given in any of these two forms. Furthermore, using the conversion PCBF⇝MWF and the Algorithm 1 we have the following result.

Lemma 7. *PCBF⇝EWF can be solved with polynomial-delay.*

Note that Algorithm 1 cannot be applied for BDDs because we do not have explicitly all the minimal winning coalitions, and by Lemma 5, BDDF⇝EWF cannot be computed in polynomial time. The existence of an algorithm with polynomial-delay for the conversion problems BDDF⇝EWF, BDDF⇝MWF and BDDF⇝PCBF remains open.

5. Regular games

An important subclass of simple games is the subfamily of *regular games*, also known as *directed games* [60]. Regular games have been studied without a particular name at least since 1966 by Maschler and Peleg [65] in the study of the kernel of a game. In the context of Boolean functions, they are known as *regular functions* at least since 1969 by Sheng [87]. They have also been used to solve other problems, such as the regular set-covering problem [80] or the problem of separating hyperplanes [29].

In this section, we assume that $N = \{1, \ldots, n\}$ is ordered and we denote this ordering by $\leq$. Let $\Gamma = (N, \mathcal{W})$ be a simple game and $i, j \in N$ be two players. Player j is at least as *desirable* or *influent* as player i, which is denoted by $i \preceq j$, if, for $X \in \mathcal{W}$ with $i \in X$ and $j \notin X$, $X \backslash \{i\} \cup \{j\} \in \mathcal{W}$.

The *desirability relation* $\preceq$ was firstly introduced for simple games in 1958 by Isbell [53]. It is also known under other names: *desirability order* [92], *dominance relation* [32], or *Winder order* [100] in threshold logic. The operation $X \backslash \{i\} \cup \{j\}$ is called a *right-shift of* X [77, 60, 24], which has been studied in other contexts, such as non-cooperative games [19], fair division [20] or Boolean functions [47]. In [65], the notion of desirability relation was generalized to cooperative games. The desirability relation can be generalized to coalitions [92]: given $Y, Z \in \mathcal{W}$, $Y \preceq Z$ if and only if there exists a finite sequence of right-shifts on Y which produces a $Y' \subseteq Z$. This relation is not linear (total).

A simple game with a set of players N is *regular* [81, 64] if the desirability relation is a total linear order and compatible with the order on N, i.e., for $i, j \in N$, $i \leq j$ implies $i \preceq j$. Observe that the required property is equivalent to say that, for any winning coalition, every right-shift is a winning coalition.

When in a simple game the desirability relation is a total linear order, the set of players can be sorted in such a way that the game becomes regular. Such games are called *linear* or *swap-robust* [92], *complete* [23] or *ordered* [60]. We prefer to use the term linear instead of complete to avoid confusion with the games corresponding to complete

hypergraphs [82]. Complete hypergraphs, according to [86], corresponds to *strong* games [93], a kind of simple game which we do not consider in this paper. Linear games can be represented by *2-monotone Boolean functions* [101], and they have been studied in various contexts since long ago [53, 73].

It is known that given a simple game in MWF, it can be decided whether the game is regular (linear) or not in polynomial time. In fact, regularity can be decided in linear time and linearity in $O(n^2 + n|\mathcal{W}^m|)$ time [64].

5.1. Regular games represented by sets and incidence vectors

The following definition, based on [92], is important because, unlike there exist other versions more similar to MWF (see, for instance, [39]), it produces a more succinct form of representation.

A winning coalition X is *shift-minimal* if, for $Z \in \mathcal{W}$, $Z \not\prec X$. Shift-minimal winning coalitions are relevant to regular games. Given a simple game Γ, $\mathcal{W}^s(\Gamma)$ denotes the set of all shift-minimal winning coalitions of Γ. According to Krohn and Sudhölter each regular game Γ is completely determined by $\mathcal{W}^s(\Gamma)$ [60] (which is based on the unpublished result of [78]). Shift-minimal winning coalitions in the context of monotone Boolean functions are related with the *shelters* [81]. Observe that, once the desirability ordering is known, checking shift-minimality can be implemented in polynomial time. Thus, the family of shift-minimal winning coalitions is a valid form of representation for regular games.

Definition 8. A regular game Γ is represented in *shift-minimal winning form (SWF)* by a pair $(N, \mathcal{W}^s)$. N is its set of players and $\mathcal{W}^s$ is the set of shift-minimal winning coalitions of Γ.

As before, a matrix notation is useful to represent regular games in a computational context. We use an injective function $f : \{0,1\}^n \to \mathbb{N}^n$ associating a winning coalition x to a unique integer vector $\bar{x}$. For $a \in N$, define $f(x)(a) = \bar{x}(a) = \Sigma\{x(b) \mid b \in N, a \preceq b\}$. Observe that, $X \preceq Y$ if and only if $\bar{x} \leq \bar{y}$.

Example 7. It is easy to see that the simple game $\Gamma = (N, \mathcal{W}^m)$ of Example 1 is regular. Let be $\overline{\mathcal{W}}^m = \{\bar{x} \in \mathbb{N}^n \mid x \in \mathcal{W}^m\}$, then:

N	*abcde*
$\overline{\mathcal{W}}^m$	22221 : $\bar{x}_1$
	22211 : $\bar{x}_2$
	33210 : $\bar{x}_3$
	32210 : $\bar{x}_4$
	32111 : $\bar{x}_5$

Therefore, as $\bar{x}_2 \leq \bar{x}_1$ and $\bar{x}_4 \leq \bar{x}_3$, it holds that:

N	*abcde*
$\mathcal{W}^s$	00101
	10110
	11001

It is clear that for every regular game, its set of shift-minimal winning coalitions is a subset of all its minimal winning coalitions.

5.2. Fully Condensed Binary Trees

We present here a representation based on trees that is more succinct than PCBF for regular games. It does not represent the set of shift-minimal winning coalitions but uses the desirability ordering to represent the set of minimal winning coalitions.

Given a simple game Γ and some total order over N, a *fully condensed binary tree* $FCB(\mathcal{W}^m(\Gamma))$ is a binary tree obtained from $PCB(\mathcal{W}^m(\Gamma))$ by recursively removing all edges (t_i, t_{i+1}) such that t_i has no right-child, and for each removed edge, merging both nodes t_i and t_{i+1}. It is clear that this can be carried out in polynomial time.

FCBs were defined by Makino [64] to decide in linear time whether a monotone Boolean function, i.e., a simple game, is regular or not. In a non-binary, but alphanumeric context, they were simultaneously defined in 1968 as *Patricia tries* [72] and without name [46], being nowadays also known as *radix trees*. Like in PCBs, each minimal winning coalition is represented by a leaf and its path from the root. Analogously to previous data structures we have that determining whether a coalition belongs to the represented set can be done in polynomial time.

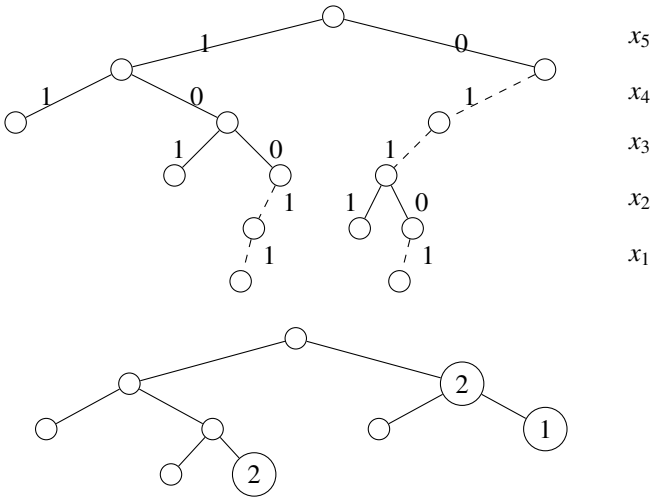

Figure 7: *FCB*(Γ) obtained from *PCB*(Γ) for the game Γ of Example 1. Nodes between dashed edges on the first tree are merged in the second one.

Definition 9. A regular game Γ is given in *fully condensed binary tree form (FCBF)* by a *FCB* representing $\mathcal{W}^m(\Gamma)$.

For any regular game Γ, *FCB*(Γ) is always *complete*, in the sense that it has no inner nodes having only one child. For non-regular simple games this may be not true [64].

Example 8. Starting with *PCB*(Γ) for the game given in Example 1 (see Figure 5) we can construct the *FCB*(Γ) illustrated in Figure 7. The nodes have been labeled with integers representing the number of left-children merged with the current node. Each node is labeled by a number representing the merged components 1. All the missing endings are formed with the adequate number of $0's$.

5.3. Representation sizes and the conversion problem

For a simple game Γ, $|FCB(\Gamma)| \leq |PCB(\Gamma)|$. Moreover, the number of leaves in *FCB* and *PCB* is equal to $|\mathcal{W}^m|$. However, the total number of nodes is lower than $n\,|\mathcal{W}^m|$. Thus, we can conclude that the difference between $|FCB(\Gamma)|$ and $|PCB(\Gamma)|$ is always polynomial in terms of n. As we shown before, $\mathcal{W}^s(\Gamma) \subseteq \mathcal{W}^m(\Gamma)$, so $|\mathcal{W}^s(\Gamma)| \leq |\mathcal{W}^m(\Gamma)|$. However, we can find regular games in which the difference between $|\mathcal{W}^s|$ and $|\mathcal{W}^m|$ grows exponentially in terms of n.

Example 9. Consider the simple game Γ constructed by Sperner [88], see also [42]. The game has $\mathcal{W}^m(\Gamma) = \{X \subseteq N \mid |X| = \lfloor \frac{n}{2} \rfloor\}$. Observe that, $|\mathcal{W}^m(\Gamma)| = \binom{n}{\lfloor n/2 \rfloor}$, but $|\mathcal{W}^s(\Gamma)| = 1$, because $\mathcal{W}^s(\Gamma) = \{1^{n/2}0^{n/2}\}$.

Note that $|PCB(\Gamma)| \geq \binom{n}{\lfloor n/2 \rfloor}$ and $|FCB(\Gamma)| \geq \binom{n}{\lfloor n/2 \rfloor}$, so the difference between $|\mathcal{W}^s(\Gamma)|$ and $|PCB(\Gamma)|$, as well as between $|\mathcal{W}^s(\Gamma)|$ and $|FCB(\Gamma)|$, can also grow exponentially in terms of n.

Example 10. For $n = 8$, the regular game Γ given in Figure 8 has $|\mathcal{W}^s(\Gamma)| = 14$ and this is the maximum number of shift-minimal winning coalitions for a regular game of size 8. Note that the number of shift-minimal winning coalitions almost doubles the number of players. Obviously, the number of minimal winning coalitions is even bigger.

Actually, as proved by Krohn and Sudhölter [60] (see also [61]), there are subclasses of regular games for which the number of shift-minimal winning coalitions grows exponentially in terms of n. This number grows more slowly than the number of minimal winning coalitions.

From the previous considerations and Lemmas 5 and 6, we have the following results.

Lemma 8. *For regular games, SWF⇝EWF, SWF⇝MWF, SWF⇝PCBF, SWF⇝FCBF, BDDF⇝FCBF and FCBF ⇝ EWF can be solved in exponential time but they can not be solved in sub-exponential time.*

N	$abcdefgh$
	00001110
	00010101
	00100011
	00111100
	01011010
	01100110
$\mathcal{W}^s$	01101001
	10010110
	10011001
	10100101
	11000011
	11011100
	11101010
	11110001

Figure 8: The regular game with the maximum number of shift-minimal winning coalitions, for $n = 8$.

Our next results establishes the polynomially solvable conversion problems.

Lemma 9. *For regular games, EWF⇝SWF, MWF⇝SWF, PCBF⇝SWF, PCBF⇝FCBF, FCBF⇝PCBF, EWF⇝FCBF, MWF⇝FCBF, FCBF⇝MWF, FCBF⇝SWF and FCBF⇝BDDF can be solved in polynomial time.*

Proof. The polynomial time algorithm for the first two problems follows the same steps as Algorithm 1 solving EWF⇝MWF. Just compare all (minimal) winning coalitions to each other, but deleting those whose respective vectors $\bar{x}$ are bigger in the lexicographic order. $\bar{x}$ can be computed when the comparison with a (minimal) winning coalition is performed. Thus, the procedure takes $O(n \cdot |\mathcal{W}|^2)$ time, for EWF⇝SWF, and $O(n \cdot |\mathcal{W}^m|^2)$, for MWF⇝SWF.

As PCBF⇝MWF and MWF⇝SWF can be solved in polynomial time, then PCBF⇝SWF can also be solved in polynomial time.

From the definition of FCBF, it is straightforward to see that PCBF⇝FCBF and FCBF⇝PCBF are polynomial time solvable. The remaining results follow from this fact and Lemma 6. □

In addition observe that it is known that, for regular games, MWF⇝MLF can be solved in polynomial time [81]. Recall that this conversion problem requires exponential time for simple games.

The size relationship between SWF and BDDF is less clear. As we mentioned in Section 4.1, Theorem 4 of [50] defines a subclass of regular games for which the size of a BDD representation grows exponentially as a function of n. However, it does not seems that, for this subclass, $|\mathcal{W}^s(\Gamma)|$ grows much slower than $|BDD(\Gamma)|$. In Example 9 we provide a game Γ with $|\mathcal{W}^s(\Gamma)| = 1$, however $|BDD(\Gamma)|$ does not seems to increase too much in terms of n. For instance, with $n = 8$ and $n = 9$ we obtain respectively $|BDD(\Gamma)| = 27$ and $|BDD(\Gamma)| = 32$, in contrast to the sizes of each game in MWF, which are $n \cdot |\mathcal{W}^m| = 560$ and 1134, respectively. Note that, even if it were shown that both $|\mathcal{W}^s|$ and $|BDD(\Gamma)|$ have an exponential growth, we could not deduced that SWF⇝BDDF is polynomial time solvable.

To compute SWF⇝BDDF (or BDDF⇝SWF), we can always compute first SWF⇝MWF (resp. BDDF⇝MWF) and then MWF⇝BDDF (resp. MWF⇝SWF). But note that those processes require exponential time. The absence of known algorithms that are able to skip this intermediate step, leads us to state the following conjecture.

Conjecture 1. *For regular games, BDDF⇝SWF and SWF⇝BDDF can be solved in exponential time but they can not be solved in sub-exponential time.*

It is interesting to note that there are some efficient algorithms that benefit themselves from working with a smaller class of simple games such as regular games. For instance, given a regular game Γ, computing either $BDD(\mathcal{W}^m(\Gamma))$ or $BDD(\mathcal{L}^M(\Gamma))$ from $BDD(\mathcal{W}(\Gamma))$ can be done in linear time [13]. Further, there exists an algorithm to compute $BDD(\mathcal{W}^s(\Gamma))$ from $BDD(\mathcal{W}^m(\Gamma))$ [13].

Algorithm 4 GenerateMWFfromSWF

Input: A simple game Γ in SWF with $\mathcal{W}^s = \{X_1, \ldots, X_m\}$ sorted in card-lexicographic order.
Output: Γ in MWF.

```
1:  GENERATE(X, R, i)
2:    for all j ∈ R in increasing order
3:      X = X ∪ {j + 1} \ {j};
4:      if (for all k > i, X_k ⊄ X) ∧ (for all k < i, X_k ⊄ X)
5:        print X;
6:        R = R \ {j};
7:        if (j > 1) ∧ (j − 1 ∉ X)
8:          R = R ∪ {j − 1};
9:        if j + 2 ∉ X
10:         R = R ∪ {j + 1};
11:       GENERATE(X, R, i);
12:       X = X ∪ {j} \ {j + 1};
13: {main}
14: for i = 1, ..., m do
15:   print X_i;
16:   R = {i ∈ X_i | i + 1 ∉ X_i};
17:   GENERATE(X_i, R, i);
```

We finish this section with some results on the enumeration problems.

Lemma 10. *For regular games, SWF⇝MWF and FCBF⇝EWF can be solved with polynomial-delay.*

Proof. The algorithm for SWF⇝MWF is given in Algorithm 4 having as input the set $\mathcal{W}^s = \{x_1, \ldots, x_m\}$. Observe that it follows similar ideas to those in Algorithm 1.

Given $X \in \mathcal{W}^s$, let $R = \{i \in X \mid i + 1 \notin X\}$. This means that for any $j \in R$ we can do a 1-*right-shift* applied to j, i.e., to replace player j by player $j + 1$. Note that given a winning coalition, to do a 1-right-shift implies that the new coalition is still winning. Steps 7-10 update the set R. Steps 7-8 consider the case where, being $j > 1$, $j - 1 \in X$, $j \in X$ and $j + 1 \notin X$, then a 1-right-shift applied to j implies that $j \notin R$ but $j - 1 \in R$. Steps 9-10 consider the case where $j \in X$, $j + 1 \notin X$ and $j + 2 \notin X$, then a 1-right-shift applied to j implies that $j \notin R$ but $j + 1 \in R$.

Again, the algorithm is a branch and cut algorithm that uses the usual backtrack tree providing an enumeration of all possible 1-right-shifts of a minimal winning coalition without repetitions. Now, for each new minimal winning coalition X, we perform a traversal of the backtrack tree, unless we reach a set X_k such that:

i) $X_k \preceq X$, for any $k > i$, thus it will be generated later, or

ii) $X_k \subset X$, for any $k < i$, so the coalition is not minimal.

These properties and the monotonicity guarantee that we generate all minimal winning coalitions without repetitions. Following the same reasoning as for Algorithm 1 of Lemma 3, the number of steps between the printing of one minimal winning coalition and the next one is polynomial.

For FCBF⇝EWF, just compute FCBF⇝MWF in polynomial time by Lemma 9 and then apply Algorithm 1. □

Example 11. Consider the simple game $(N, \mathcal{W}^s)$ given in Example 7. Figure 9 shows the enumeration without repetition of all the minimal winning coalitions of the game. They are printed in the following order: 00101, 00011, 10110, 01110, 11001.

Whether the conversion problem SWF⇝EWF can be solved with polynomial-delay is left as an open problem. Similar to what happens for BDDs (see Conjecture 1) Algorithm 4 cannot be used with BDDs. Therefore, whether for regular games BDDF⇝SWF can be solved with polynomial-delay remains as an open problem.

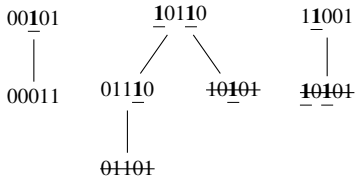

Figure 9: Computation of SWF⇝MWF, for the simple game given in Example 1.

6. Weighted games

Weighted games, also called *weighted voting games* or *weighted majority games*, are an interesting subclass of simple games in which the players have an assigned weight. For a weight assignment $w : N \to \mathbb{R}$, we use the notation $w(U) = \Sigma\{w(i) \mid i \in U\}$ to denote the sum of the weights of the elements in $U \subseteq N$. A simple game $\Gamma = (N, \mathcal{W})$ is a *weighted game* if there exists a *weight function* $w : N \to \mathbb{R}$ and a *quota* $q \in \mathbb{R}$ such that, for $X \subseteq N$, $X \in \mathcal{W}$ if $w(X) \geq q$.

Probably the first use of weighted games was the modeling of the first artificial neuron, the so called *Threshold Logic Unit (TLU)* [67]. In game theory, they were defined by von Neumann and Morgenstern [93] and deeply studied by Isbell [52]. Since then, they have been studied under different names in many applications: *linearly separated truth functions* [68] or *linearly separable switching functions* [51] to contact and to rectify nets; *trade robustness* [92] in voting theory and trade exchanges; *linearly separable switching functions* or *threshold Boolean functions* [51] to separate circuits in switching circuit theory and to analyse the threshold synthesis problem; or *threshold hypergraphs* [44, 84] to synchronize parallel processes, just to name a few.

All weighted games are linear games [74], but the opposite is not true [65, 84]. As in regular games, we assume that the set N is ordered but here players are sorted in *decreasing* order of weights. Thus if $N = \{1, \ldots, n\}$ we assume $w_i \geq w_{i+1}$, for $1 \leq i < n$.

6.1. Weighted representation

According to Hu [51] (see also [35, 39]) the weight function can be restricted to be a *natural weight function* $w : N \to \mathbb{N}$, where $\mathbb{N}$ includes the zero. This leads to the following form of representation for weighted games.

Definition 10. A weighted game Γ is given in *weighted representation form (WRF)* by a tuple $[q; w_1, \ldots, w_n]$, where $q \in \mathbb{N}$ is the quota and $w_1, \ldots, w_n \in \mathbb{N}$ are the weights of its players.

Observe that any tuple $[q; w_1, \ldots, w_n]$ represents a weighted game and thus the WRF is a valid representation for weighted games.

Given a simple game in MWF, it can be decided in polynomial time whether the game is weighted or not. This problem is known in the context of Boolean functions as *threshold synthesis problem* and was solved by Peled and Simeone [80]. One common way to do this is solving the following system of linear inequalities:

$$w(X) > w(Y) \qquad \forall X \in \mathcal{W}^m, Y \in \mathcal{L}^M \tag{1}$$

where $w = (w_1, \ldots, w_n)$ are the unknowns. The game will be weighted if and only if the linear system has solution. The weighted realization $[q; w_1, \ldots, w_n]$ can be obtained from a solution $w_1, \ldots, w_n$ of the system taking $q = \min_{X \in \mathcal{W}^m} w(X)$. Hence each solution of this linear system defines a representation of the weighted game.

Considering that N is an ordered set, each simple (or regular) game can be univocally represented in EWF, MWF, PCBF or BDDF (SWF or FCBF). However, a weighted game may be represented in infinite ways in WRF respecting the given order. This is true even for our restriction over natural numbers. Actually, given a weighted game with realization $[q; w_1, \ldots, w_n]$, the realizations $[cq; cw_1, \ldots, cw_n]$, for $c \in \mathbb{N}^+$, are equivalent. Nevertheless, there may be further equivalent realizations, as for instance $[2; 2, 1, 1]$ and $[3; 3, 2, 1]$ define the same weighted game. In fact the relationship might be complex as it is known that, given two realizations, determine whether both represent the same weighted game is an *NP*-hard problem [66].

Despite the fact that weighted games are a strict subclass of simple games, a well known result says that every simple game can be expressed as the intersection of a finite number of weighted games. Given k weighted games $\Gamma^t = [q^{(t)}; w_1^{(t)}, \ldots, w_n^{(t)}]$, $1 \le t \le k$, X is a winning coalition in $\Gamma = \Gamma^1 \cap \cdots \cap \Gamma^k$, if $w^{(t)}(X) \ge q^{(t)}$ for $1 \le t \le k$. This leads to another form of representation for simple games.

Definition 11. A simple game Γ is given in *vector-weighted representation form (VWRF)* by a set of k weighted games $\Gamma^1, \ldots, \Gamma^k$ in WRF, that is, $\{[q^{(t)}; w_1^{(t)}, \ldots, w_n^{(t)}] \mid 1 \le t \le k\}$. Such a representation defines the game $\Gamma = \Gamma^1 \cap \cdots \cap \Gamma^k$.

The equivalence between simple games and vector-weighted games was firstly shown in [55] for hypergraphs, and then expressed for simple games in [91, 92]. Therefore, as the intersection of weighted games is well defined, the VWRF is a valid form of representation for simple games.

Vector-weighted games are also known as *vector-weighted systems* [91], *weighted multiple majority games* [2], *multiple weighted voting games* [4] or by some other combination of these words; in the context of hypergraphs, the VWRF is known as *threshold intersection* [99, 63] and in switching functions as *canonical conjunctive form* [73]. Given a simple game, the minimal integer k for which is possible to express the game as an intersection of k weighted games is known as the *dimension* of the game. It is known that given k weighted games, to decide whether the dimension of their intersection exactly equals k is *NP*-hard [28].

In a similar way, we can consider another game represented by a set $\{[q^{(t)}; w_1^{(t)}, \ldots, w_n^{(t)}] \mid 1 \le t \le k\}$, using the union of games instead of intersection. We refer to such representation as *co-vector-weighted representation form (co-VWRF)*. It is known that any simple game can be expressed as the union of a finite family of weighted games [38]. Therefore, co-VWRF is a valid representation form for simple games.

Given a simple game, the minimal integer k for which is possible to express the game as a union of k weighted games is known as the *codimension* of the game [38]. Given k weighted games, it remains open to show whether the problem of deciding if the codimension of their union exactly equals k is *NP*-hard.

Observe that, for any representation of weighted games that is closed under intersection or union, i.e., a representation of the intersection or union of two games that can be obtained in polynomial time, the conversion problem for simple games given in VWRF or co-VWRF has the same complexity than for weighted games given in WRF.

We have considered only the operations of intersection and union. For a generalization to representations of games including more involved combinations of boolean operations on weighted games, we refer the reader to [31]. To the best of our knowledge, the conversion problem from other forms of representation to those based on boolean operations has not been explored.

6.2. Representation sizes and the conversion problem

Ishiura and Yajima [54] proved that, for WRF in which weights are bounded by a polynomial of n, the sizes of their respectives BDDs grow polynomially in terms of n. That is the case of the following example. Observe that in the example the number of minimal winning coalitions grows exponentially in terms of n.

Example 12. Consider the family of weighted games $\Gamma = [q; n, \ldots, 1]$. The number of minimal winning coalitions for Γ is specified by the following recurrence $s(q, n)$:

$$s(q,n) = \begin{cases} 0 & \text{if } q = 0 \text{ or } n = 0 \text{ or } q > \sum_{i=1}^{n} i \quad \text{(weights are not enough to get } q\text{)} \\ n - q + s(q,q) & \text{if } q > n \quad (\{n\}, \{n-1\}, \ldots, \{q+1\} \in \mathcal{W}^m) \\ 1 + s(q, n-1) & \text{if } q = n \quad \text{(take } n \text{ or not)} \\ s(q-n, n-1) + s(q, n-1) & \text{if } q < n \quad \text{(take } n \text{ or not)} \end{cases}$$

Observe that taking $\bar{q} = \lceil (\sum_{i=1}^{n} i)/2 \rceil$, $s(\bar{q}, n)$ grows exponentially in n as we can see in Table 6. The reported values can be checked with the code provided in [15].

There are subclasses of weighted games whose BDDF grows exponentially in terms of n [50]. This behavior is independently of the ordering of the players and the weights in a WRF. Nevertheless, there are exponential algorithms to solve WRF$\rightsquigarrow$BDDF [11, 16].

n	10	11	12	13	14	15	16	17	18	19	20
$\lvert BDD(\Gamma)\rvert$	74	91	117	150	184	223	274	331	388	459	545
$\lvert \mathcal{W}^m\rvert$	77	133	240	429	772	1414	2588	4742	8761	16273	30255

Table 6: Growth of BDDF and MWF for the class of weighted games considered in Example 12.

As there are weighted games with a WRF of polynomial size in n and with other representations of exponential size in n (see Sections 4.1 and 5.3), the first part of the following lemma holds trivially. For the

As there are weighted games with a WRF of size polynomial in n and an exponential size in other representations (see Sections 4.1 and 5.3), the first part of the following lemma holds trivially. For the second part, just note that every weighted game is a simple game with dimension 1.

Lemma 11. *For weighted games, WRF$\rightsquigarrow$EWF, WRF$\rightsquigarrow$MWF, WRF$\rightsquigarrow$PCBF, WRF$\rightsquigarrow$BDDF, WRF$\rightsquigarrow$FCBF and also WRF$\rightsquigarrow$SWF can be solved in exponential time but they can not be solved in sub-exponential time. For simple games, the same occurs replacing WRF by VWRF or co-VWRF.*

Despite of this result, there exist some subclasses of weighted games for which WRF$\rightsquigarrow$BDDF turns out to be polynomial. Such is the case of *homogeneous games*, i.e., weighted games $[q; w_1, \ldots, w_n]$ such that, for $X \in \mathcal{W}^m$, $q = w(X)$. Homogeneus games were defined for the first time by von Neumann and Morgenstern [93], being one of the most studied subclasses of weighted games [89].

Example 13. The regular game Γ of Example 9 is both weighted and homogeneous, and it can be represented in WRF by the vector $[\frac{n}{2}; 1, 1, \ldots, 1]$.

It is known that every homogeneous game can be represented by a QOBDD with size $O(n^2)$, and that from its weighted representations, this QOBDD can be computed in $O(n^2 \cdot \log n)$ time [17]. Therefore, applying Algorithm 3 to the QOBDD, we obtain a BDD representation. Therefore, WRF$\rightsquigarrow$BDDF for homogeneous games can be solved in polynomial time.

Lemma 12. *For weighted games, EWF$\rightsquigarrow$WRF, MWF$\rightsquigarrow$WRF, FCBF$\rightsquigarrow$WRF and PCBF$\rightsquigarrow$WRF can be solved in polynomial time.*

Proof. Fredman and Khachiyan [34] shown that, given a simple game Γ in MWF, $\mathcal{L}^M(\Gamma)$ can be computed in sub-exponential time, but it is still an open problem to show whether it can be computed in polynomial time. However, if Γ is regular or weighted, the problem turns out to be polynomial time solvable [81]. Therefore, we can obtain $\mathcal{L}^M(\Gamma)$ from MWF in polynomial time. Recall that the solutions of the system (1) allow us to compute a WRF for the game. When the game is given in EWF or MWF, we can compute $\mathcal{L}^M(\Gamma)$ and write down the system (1) in polynomial time. Thus, using the fact that linear programming is polynomial time solvable [58], we have that MWF$\rightsquigarrow$WRF and EWF$\rightsquigarrow$WRF are polynomial time solvable.

Recall that FCBF$\rightsquigarrow$PCBF can be solved in polynomial time. To compute PCBF$\rightsquigarrow$WRF, we use an intermediate representation by means of a QOBDD. We construct the corresponding QOBDD from PCBF in polynomial time using Algorithm 3, without step 5. Once we have the QOBDD representation, we compute a WRF using the polynomial time algorithm provided in [17], which consists in solving a linear programming problem associated to the QOBDD. Putting all together FCBF$\rightsquigarrow$WRF and PCBF$\rightsquigarrow$WRF can be solved in polynomial time. □

We finish this section with some enumeration results. The proof of the following Lemma 13 is based on an algorithm of [66], that the authors used to compute the *Deegan-Packel index*, a power index defined in 1978 [27], for weighted games in MWF.

Lemma 13. *For weighted games, WRF$\rightsquigarrow$EWF and WRF$\rightsquigarrow$MWF can be solved with polynomial-delay.*

Proof. The second part of the lemma, WRF$\rightsquigarrow$MWF, was proved by [66] using Algorithm 5. Analogously to Algorithms 1 and 4, steps 1-8 form the recursive procedure started by the main routine of step 11. Each step, excluding step 10, involves a constant number of operations. The whole algorithm requires $O(n|\mathcal{W}^m|)$ steps and uses $O(n)$ memory.

Algorithm 5 GenerateMWFfromWRF

Input: A weighted game $\Gamma = [q; w_1, \ldots, w_n]$, with $\sum_{i=1}^{n} w_i \geq q$ and $w_1 \geq \ldots \geq w_n$.
Output: Γ in MWF.

```
1: WRFtoMWF(X, i, q', S)
2:   if i = n
3:     print X ∪ {n}; return;
4:   else
5:     if (S − w_i ≥ q)  WRFtoMWF(X, i + 1, q', S − w_i);
6:     if (w_i ≥ q')  print X ∪ {i};
7:     else  WRFtoMWF(X ∪ {i}, i + 1, q' − w_i, S − w_i);
8:     return;
9: {main}
10: S = Σ_{j=1}^{n} w_j
11: WRFtoMWF(∅, 1, q, S);
```

Algorithm 6 GenerateEWFfromWRF

Input: A weighted game $\Gamma = [q; w_1, \ldots, w_n]$, with $\sum_{i=1}^{n} w_i \geq q$ and $w_1 \geq \ldots \geq w_n$.
Output: Γ in EWF.

```
1: WRFtoEWF(X, i)
2:   if i ≤ n
3:     if w(X ∪ {i}) ≥ q
4:       print X ∪ {i};
5:     WRFtoEWF(X ∪ {i}, i + 1);
6:     WRFtoEWF(X, i + 1);
7: {main}
8: WRFtoEWF(∅, 1);
```

For WRF$\rightsquigarrow$EWF, we use Algorithm 6. The algorithm requires $O(n|\mathcal{W}|)$ time and $O(n)$ memory. The recursive calls in steps 5 and 6 generate the winning coalitions in order, so as to insure that none is repeated. □

We believe that a similar procedure to Algorithms 5 and 6 could be derived to postulate the following.

Conjecture 2. *For weighted games, WRF$\rightsquigarrow$SWF can be solved with polynomial-delay.*

7. Conclusions

This paper focused on the most usual forms of representation for simple games summarized in Table 1. Furthermore, we review some classic forms to represent regular games and weighted games, probably the most studied subclasses of simple games. We analyzed the sizes of those forms of representation. We surveyed and obtained results about the complexity of the conversion problem, how to obtain one representation form from another one. Additionally, we have considered the design of algorithms to solve the conversion problem with polynomial-delay in those cases in which the conversion problem is not solvable in polynomial time. Tables 2-5 summarize the complexity results presented in this paper. Through these tables, we can realize that MWF, PCBF and FCBF seem to behave similarly. However, we must not forget that FCBF, only valid for regular games, is the most succinct of the three.

On the other hand, BDDF seems to have a behavior similar to SWF. Indeed, both BDDF and SWF can be useful because of their succinctness. However, as discussed in Sections 4.1 and 5.3, games in SWF and BDDF may also have an exponential size in terms of n.

Finally, weighted games have a more succinct form of representation, based on a $(n + 1)$-vector of integers. However, it is known that there are weighted games whose weighted representation requires that $\max_{i \in N}\{w_i\}$ to be $(n+1)^{(n+1)/2}/2$ [79]. Converting any of the forms of representation that we consider to WRF turns out to be polynomial, but converting WRF to any other requires exponential time and can not be done in polynomial time. We provided

enumeration algorithms with polynomial-delay for several cases. We conjecture that WRF$\rightsquigarrow$SWF can also be solved with polynomial-delay.

To conclude, the results presented in this article can be useful for having at hand a comparative of the different usual forms of representation. When we want to solve some decision problem on simple games, if we know how the problem behaves for some particular form of representation, then we can know under which other forms of representation the problem has similar complexity. For instance, if we consider the decisive problem mentioned in Section 1, which we know is in *QP* for simple games in MWF [34, 86], from Tables 2 and 4 we can deduce immediately that it is also in *QP* for simple games in PCBF and for regular games in FCBF.

Besides the classical subfamilies of simple games considered in this paper, any cooperative game can be used to derive a subfamily of simple games using the so-called threshold version. A *cooperative game* is a pair (N, v) where N is a set of players and $v : \mathcal{P}(N) \to \mathbb{R}$ is a *valuation function* providing the benefit of the coalitions. The *threshold* version of a cooperative game (N, v) is the simple game $(q; N, v)$ were a coalition X is winning if and only if $v(X) \geq q$. Observe that weighted games can be considered threshold games associated to a cooperative game where the valuation function is the sum of the weights of the participating players. There has been a lot of attention devoted to families of cooperative games defined on graphs and graph parameters. In this setting the set of players is usually formed by either the vertices or the edges of the graph. The valuation function measures some combinatorial parameter of the subgraph induced by the coalition. We refer the interested reader to [25, 76] for a more exhaustive survey.

Many such cooperative games defined on graphs have lead to the definition of subfamilies of simple games, addressing different contexts as trading, management or flow interactions among many others. We mention here a few. *Vertex connectivity games* [8] were motivated in the context of network reliability. Given a graph in which the set of vertices V is partitioned on three subsets (V_p, V_b, V_s), the set of players of the game is V_s, and a coalition $S \subseteq V_s$ is winning if and only if $S \cup V_b \cup V_p$ *fully connects* V_p, i.e., for $u, v \in V_p$, there exists a path from u to v. *Flow games* constitute a model of cooperative games based on flow networks, which are directed graphs with positive labels on the edges [57]. A simplification of flow games called *connectivity games* was defined in [9]. In this model every edge has the same unit capacity, and the flow also has a unit value. Winning coalitions correspond to paths from the source to the sink and losing coalitions are those subsets of edges that do not induce a source-sink path. *Spanning connectivity games* were introduced in [6]. There the players are the edges of an undirected weighted multigraph so that a coalition is winning when the edges in the coalition form a connected spanning subgraph. Furthermore, a threshold variant of network flow games called *shortest path games* was defined in [76]. Here, given a flow network with multiple sources and sinks, every shortest path $P = (v_1, \ldots, v_m)$ through the edges $(e_1, \ldots, e_{m-1})$ is a coalition with value $\sum_{i=1}^{m-1} w_i - m$, if v_1 is a source and v_m is a sink; or value 0, otherwise [33, 94]. Considering a threshold T, a coalition is winning when $\sum_{i=1}^{m-1} w_i - m \geq T$.

Recently, *influence games* were defined as simple games based on a model of spread of influence in a social network, where influence spreads according to the linear threshold model [45, 59]. Here we have a tuple (G, w, f, q, N), where (G, w, f) is a both vertex and edge-labeled directed graph, $N \subseteq V(G)$ and for any $X \subseteq N$, X represents a winning coalition if and only if it can spread its influence to at least q vertices in the graph [70] (see also [85]). It is also known that influence games capture the complete class of simple games and that some simple games require a representation as unweighted influence games with exponential number of vertices. It would be of interest to determine the complexity of the conversion problem for those families of simple games defined through graphs.

Acknowledgments

The authors thank the editors and the anonymous referees for carefully reading a preliminary version of this article and giving useful comments and suggestions that helped us to improve the contents and the presentation of the paper.

Xavier Molinero has been partially supported by grant MTM2012–34426/FEDER from the Spanish Ministry for Economy and Competitiveness. Fabián Riquelme was supported by grant BecasChile of the "National Commission for Scientific and Technological Research of Chile" (CONICYT). Maria Serna has been partially supported by funds from the Spanish Ministry for Economy and Competitiveness (MINECO) and the European Union (FEDER funds) under grant COMMAS (ref. TIN2013-46181-C2-1-R), and SGR 2014 1137 (ALBCOM) of the Catalan government.

References

[1] S. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, C-27(6):509–516, 1978.

[2] E. Algaba, J. M. Bilbao, J. R. Fernández-García, and J. J. López. Computing power indices in weighted multiple majority games. *Mathematical Social Sciences*, 46(1):63–80, 2003.

[3] H. Aziz. Complexity of comparison of influence of players in simple games. In U. Endriss and P. W. Goldberg, editors, *Proceedings of the 2nd International Workshop on Computational Social Choice, (COMSOC 2008)*, pages 61–72, 2008.

[4] H. Aziz. *Algorithmic and complexity aspects of simple coalitional games*. PhD thesis, Department of Computer Science, University of Warwick, 2009.

[5] H. Aziz, F. Brandt, and P. Harrenstein. Monotone cooperative games and their threshold versions. In W. van der Hoek, G. A. Kaminka, Y. Lespérance, M. Luck, and S. Sen, editors, *9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada, May 10-14, 2010, Volume 1-3*, pages 1107–1114, 2010.

[6] H. Aziz, O. Lachish, M. Paterson, and R. Savani. Power indices in spanning connectivity games. In A. V. Goldberg and Y. Zhou, editors, *Algorithmic Aspects in Information and Management, 5th International Conference, AAIM 2009, San Francisco, CA, USA, June 15-17, 2009. Proceedings*, volume 5564 of *Lecture Notes in Computer Science*, pages 55–67, 2009.

[7] Y. Bachrach, E. Elkind, and P. Faliszewski. Coalitional voting manipulation: A game-theoretic perspective. In T. Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 49–54, 2011.

[8] Y. Bachrach, E. Porat, and J. S. Rosenschein. Sharing rewards in cooperative connectivity games. *Journal of Artificial Intelligence Research*, 47:281–311, 2008.

[9] Y. Bachrach and J. S. Rosenschein. Power in threshold network flow games. *Autonomous Agents and Multi-Agent Systems*, 18(1):106–132, 2009.

[10] M. O. Ball and J. S. Provan. Disjoint products and efficient computation of reliability. *Operations Research*, 36(5):703–715, 1988.

[11] M. Behle. On threshold BDDs and the optimal variable ordering problem. *Journal of Combinatorial Optimization*, 16(2):107–118, 2008.

[12] C. Bertini, J. Freixas, G. Gambarelli, and I. Stach Comparing power indices. *International Game Theory Review*, 15(2):1–19, 2013.

[13] R. Berghammer and S. Bolus. On the use of binary decision diagrams for solving problems on simple games. *European Journal of Operation Research*, 222(3):529–541, 2012.

[14] B. Bollig and I. Wegener. Improving the variable ordering of OBDDs is NP-complete. *IEEE Transactions on Computers*, 45(9):993–1002, 1996.

[15] S. Bolus. SimpleGame Lab. `http://www.informatik.uni-kiel.de/~progsys/simple_games/lab`. February 2015.

[16] S. Bolus. Power indices of simple games and vector-weighted majority games by means of binary decision diagrams. *European Journal of Operation Research*, 210(2):258–272, 2011.

[17] S. Bolus. *A QOBDD-based approach to simple games*. PhD thesis, Department of Computer Science, University of Kiel, 2012.

[18] R. Boute. The binary decision machine as a programmable controller. *EUROMICRO Newsletter*, 1(2):16–22, 1976.

[19] S. Brams and P. Straffin Jr. Prisoners' dilemma and the professional sports drafts. *American Mathematical Monthly*, 86(2):80–88, 1979.

[20] S. Brams and A. Taylor. *The win-win solution: Guaranteeing fair shares to everybody*. W. W. Norton & Company, New York, NY, 1999.

[21] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[22] R. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, 24(3):293–318, 1992.

[23] F. Carreras. A characterization of the Shapley-Shubik index of power via automorphisms. *Stochastica*, 8(2):171–179, 1984.

[24] F. Carreras and J. Freixas. Complete simple games. *Mathematical Social Sciences*, 32(2):139–155, 1996.

[25] G. Chalkiadakis, E. Elkind, and M. Wooldridge. *Computational aspects of cooperative game theory*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011.

[26] Y. Crama and P. L. Hammer. *Boolean functions: Theory, algorithms, and applications*, volume 142 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, New York, NY, 2011.

[27] J. Deegan and E. W. Packel. A new index of power for simple n-person games. *International Journal of Game Theory*, 7(2):113–123, 1978.

[28] V. G. Deĭneko and G. J. Woeginger. On the dimension of simple monotonic games. *European Journal of Operational Research*, 170(1):315–318, 2006.

[29] E. Einy and E. Lehrer. Regular simple games. *International Journal of Game Theory*, 18(2):195–207, 1989.

[30] T. Eiter, K. Makino, and G. Gottlob. Computational aspects of monotone dualization: A brief survey. *Discrete Applied Mathematics*, 156(11):2035–2049, 2008.

[31] E. Elkind, L. A. Goldberg, P. W. Goldberg, and M. Wooldridge. On the dimensionality of voting games. In D. Fox and C. P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 69–74, 2008.

[32] P. Fishburn. Preference, summation, and social welfare functions. *Management Science*, 16(3):179–186, 1969.

[33] V. Fragnelli, I. Garcia-Jurado, and L. Mendez-Naya. On shortest path games. *Mathematical methods of operations research*, 52(2):251–264, 2000.

[34] M. Fredman and L.G. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21(3):618–628, 1996.

[35] J. Freixas. *Estructura de los juegos simples*. PhD thesis, Doctorate Program in Applied Mathematics, Technical University of Catalonia, 1994. In spanish.

[36] J. Freixas. Power indices. In J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, and J. C. Smith, editors, volume 8 of *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, 2011.

[37] J. Freixas and S. Kurz. On minimal integer representations of weighted games. *Mathematical Social Sciences*, 67:9–22, 2014.

[38] J. Freixas and D. Marciniak. A minimum dimensional class of simple games. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 17(2):407–414, 2009.

[39] J. Freixas and X. Molinero. On the existence of a minimum integer representation for weighted voting systems. *Annals of Operation Research*, 166(1):243–260, 2009.
[40] J. Freixas and X. Molinero. Weighted games without a unique minimal representation in integers. *Optimization Methods and Software*, 25(2):203–215, 2010.
[41] J. Freixas, X. Molinero, M. Olsen, and M. Serna. On the complexity of problems on simple games. *RAIRO-Operations Research*, 45(4):295–314, 2011.
[42] E. Gilbert. Lattice theoretic properties of frontal switching functions. *Journal of Mathematical Physics*, 33:57–67, 1954.
[43] D. B. Gillies. *Some theorems on n-person games*. PhD thesis, Department of Mathematics, Princeton University, 1953.
[44] M. Golumbic. *Algorithmic graph theory and perfect graphs*. Computer science and applied mathematics. Academic Press, New York, NY, 1980.
[45] M. Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83(6):1420–1443, 1978.
[46] G. Gwehenberger. Anwendung einer binären verweiskettenmethode beim aufbau von listen (use of a binary tree structure for processing files). *Elektronische Rechenanlagen*, 10(5):223–226, 1968. In German.
[47] P. Hammer, A. Kogan, and U. Rothblum. Evaluation, strength, and relevance of variables of Boolean functions. *SIAM Journal on Discrete Mathematics*, 13(3):302–312, 2000.
[48] K. Hayase, K. Sadakane, and S. Tani. Output-size sensitiveness of OBDD construction through maximal independent set problem. *Lecture Notes in Computer Science*, 959:229–234, 1995.
[49] M. J. Holler. Forming coalitions and measuring voting power. *Political Studies*, 30(2):262–271, 1982.
[50] K. Hosaka, Y. Takenaga, T. Kaneda, and S. Yajima. Size of ordered binary decision diagrams representing threshold functions. *Theoretical Computer Science*, 180(1–2):47–60, 1997.
[51] S. Hu. *Threshold logic*. University of California Press, Berkeley and Los Angeles, CA, 1965.
[52] J. Isbell. A class of majority games. *Quarterly Journal of Mathematics. Oxford Scr.*, 7(1):183–187, 1956.
[53] J. Isbell. A class of simple games. *Duke Mathematical Journal*, 25(3):423–439, 1958.
[54] N. Ishiura and S. Yajima. A class of logic functions expressible by polynomial-size binary decision diagrams. *RIMS Kokyuroku*, 754:65–71, 1991.
[55] R. G. Jeroslow. On defining sets of vertices of the hypercube by linear inequalities. *Discrete Mathematics*, 11(2):119–124, 1975.
[56] D. Johnson, M. Yannakakis, and C. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.
[57] E. Kalai and E. Zemel. Totally balanced games and games of flow. *Mathematics of Operations Research*, 7(3):476–478, 1982.
[58] L.G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.
[59] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors, Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003, pages 137–146, 2003.
[60] I. Krohn and P. Sudhölter. Directed and weighted majority games. *Mathematical Methods of Operations Research*, 42(2):189–216, 1995.
[61] S. Kurz and N. Tautenhahn. On Dedekind's problem for complete simple games. *International Journal of Game Theory*, 42(2):411–437, 2013.
[62] C. Lee. Representation of switching circuits by binary-decision programs. *Bell Systems Technical Journal*, 38(4):985–999, 1959.
[63] M. V. R. Mahadev and U. N. Peled. *Threshold graphs and related topics*, volume 56 of *Annals of Discrete Mathematics*. Elsevier, Amsterdam, 1995.
[64] K. Makino. A linear time algorithm for recognizing regular Boolean functions. *Journal of Algorithms*, 43(2):155–176, 2002.
[65] M. Maschler and B. Peleg. A characterization, existence proof and dimension bounds for the kernel of a game. *Pacific Journal of Mathematics*, 18(2):289–328, 1966.
[66] T. Matsui and Y. Matsui. A survey of algorithms for calculating power indices of weighted majority games. *Journal of the Operations Research Society of Japan*, 43(1):71–86, 2000.
[67] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.
[68] R. McNaughton. Unate truth functions. *IRE Transactions on Electronic Computers*, EC-10(1):1–6, 1961.
[69] C. Meinel. *Modified branching programs and their computational power*, volume 370 of *Lecture Notes in Computer Science*. Springer-Verlag, New York, NY, 1989.
[70] X. Molinero, F. Riquelme, and M. Serna. Cooperation through social influence. *European Journal of Operational Research*, 242(3):960–974, 2015.
[71] X. Molinero, M. Olsen, and M. Serna. On the Complexity of Exchanging. `http://arxiv.org/abs/1503.06052v2`. March 2015.
[72] D. Morrison. Patricia – practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM*, 15(4):514–534, 1968.
[73] S. I. Muroga. *Threshold logic and its applications*. Wiley-Interscience. John Wiley & Sons, New York, NY, 1971.
[74] S. I. Muroga, I. Toda, and S. Takasu. Theory of majority decision elements. *Journal of the Franklin Institute*, 271(5):376–418, 1961.
[75] National Institute of Standards and Technology. Dictionary of algorithms and data structures. `http://xlinux.nist.gov/dads`. February 2015.
[76] F. Nebel. Shortest path games: computational complexity of solution concepts. Master's thesis, Institute for Logic, Language and Computation, University of Amsterdam, 2010.
[77] A. Ostmann. Decisions by players of comparable strength. *Journal of Economic*, 45(3):267–284, 1985.
[78] A. Ostmann. Life-length of a process with elements of decreasing importance. Working Paper 156. Institut für Mathematische Wirtschaftsforschung, Universität Bielefeld, 1987.
[79] I. Parberry. *Circuit complexity and neural networks*. Foundations of Computers. MIT Press, Cambridge, MA, 1994.
[80] U. Peled and B. Simeone. Polynomial-time algorithms for regular set-covering and threshold synthesis. *Discrete Applied Mathematics*, 12(1):57–69, 1985.

[81] U. Peled and B. Simeone. An $O(nm)$-time algorithm for computing the dual of a regular Boolean function. *Discrete Applied Mathematics*, 49(1–3):309–323, 1994.
[82] A. Polyméris. Stability of two player game structures. *Discrete applied mathematics*, 156(14):2636–2646, 2008.
[83] E. Post. Introduction to a general theory of elementary propositions. *American Journal of Mathematics*, 43(3):163–185, 1921.
[84] J. Reiterman, V. Rodl, E. Sinajova, and M. Tuma. Threshold hypergraphs. *Discrete Mathematics*, 54(2):193–200, 1985.
[85] F. Riquelme. *Structural and computational aspects of simple and influence games*. PhD thesis, Department of Computer Science, Universitat Politècnica de Catalunya, 2014.
[86] F. Riquelme and A. Polyméris. On the complexity of the decisive problem in simple and weighted games. *Electronic Notes in Discrete Mathematics*, 37:21–26, 2011.
[87] C. L. Sheng. *Threshold logic*. Electrical Science: A Series of Monographs and Texts. The Ryerson Press; Academic Press, Toronto; London and New York, 1969.
[88] E. Sperner. Ein satz über untermengen einer endlichen menge. *Mathematische Zeitschrift*, 27(1):544–548, 1928. In German.
[89] P. Sudhölter. Star-shapedness of the kernel for homogeneous games and some applications to weighted majority games. *Mathematical Social Sciences*, 32(3):179–214, 1996.
[90] A. Taylor and W. Zwicker. A characterization of weighted voting. *Proceedings of the American Mathematical Society*, 115(4):1089–1094, 1992.
[91] A. Taylor and W. Zwicker. Weighted voting, multicameral representation, and power. *Games and Economic Behavior*, 5(1):170–181, 1993.
[92] A. Taylor and W. Zwicker. *Simple games: Desirability relations, trading, pseudoweightings*. Princeton University Press, Princeton, NJ, 1999.
[93] J. von Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, Princeton, NJ, 1944.
[94] M. Voorneveld and S. Grahn. Cost allocation in shortest path games. *Mathematical methods of operations research*, 56(2):323–340, 2002.
[95] I. Wegener. *The complexity of Boolean functions*. Wiley-Teubner series in computer science. John Wiley & Sons, New York, NY, 1987.
[96] I. Wegener. On the complexity of branching programs and decision trees for clique functions. *Journal of the ACM*, 35(2):461–471, 1988.
[97] I. Wegener. The size of reduced OBDD's and optimal read-once branching programs for almost all boolean functions. *IEEE Transactions on Computers*, 43(11):1262–1269, 1994.
[98] I. Wegener. *Branching programs and binary decision diagrams: Theory and applications*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.
[99] D. B. West. Parameters of partial orders and graphs: packing, covering, and representation. In I. Rival, editor, *Graphs and order*, volume 147 of *NATO ASI Series*, pages 267–350. D. Reidel Publishing, 1985.
[100] R. Winder. *Threshold logic*. PhD thesis, Mathematics Department, Princeton University, 1962.
[101] R. O. Winder. Single stage threshold logic. In *1st Annual Symposium on Switching Circuit Theory and Logical Design, Chicago, Illinois, USA, October 9-14, 1960*, pages 321–332. IEEE Computer Society, 1960.
[102] M. Zuckerman, P. Faliszewski, Y. Bachrach, and E. Elkind. Manipulating the quota in weighted voting games. *Artificial Intelligence*, 180–181:1–19, 2012.