Games on Graphs

The Complexity of Pure Nash Equilibria

Antonis Thomas

Technical Report UU-CS-2011-024

July 2011

Department of Information and Computing Sciences

Utrecht University, Utrecht, The Netherlands

www.cs.uu.nl

ISSN: 0924-3275

MSc thesis by Antonis Thomas
Supervised by Prof. dr. Jan van Leeuwen

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Games on Graphs
The Complexity of Pure Nash Equilibria

Antonis Thomas

Department of Information and Computing Sciences
Utrecht University, The Netherlands
athomas@cs.uu.nl

Abstract

In this thesis, we analyze the problem of computing pure Nash equilibria in succinctly representable games, with a focus on graphical and action-graph games. While the problem is NP-Complete for both models, it is known to be polynomial time computable when restricted to games of bounded treewidth. We propose a dynamic programming approach for computing pure Nash equilibria of graphical games. Our algorithm attacks the combinatorics of the problem directly, in contrast to previous algorithms that use mappings to other problems. The analysis yields substantial improvements over the known bounds on the time complexity of the problem. From the viewpoint of parameterized complexity, we prove that computing pure Nash equilibria for graphical games is $W[1]$-Hard when parameterized by treewidth. On the other hand, our algorithm becomes Fixed-Parameter-Tractable for games with bounded cardinality strategy sets. Moreover, we discuss the implication of our algorithm for solving games with $O(\log n)$ bounded treewidth. Finally, it is possible to construct a sample, and the maximum payoff, pure Nash equilibrium without additional computational effort.

1 Introduction

Game theory is receiving a lot of attention in computer science, in areas like the algorithmic study of competitive network routing, market mechanisms

and auctions, the behavior of large multi-agent systems, and interactions over the Internet. Algorithmic Game Theory lies at the points of contact between Game Theory, Economic Theory, Theoretical Computer Science and the Internet [34]. It is one of the most booming fields within computer science research. This is due to its important applications in areas like online auctions and electronic commerce. An indication of this importance is that major IT companies have build high-scale research groups that specialize in algorithmic game theory. The computational problem of utmost importance, in this context, is the computation of solution concepts; the most widely used such concept is the Nash equilibrium.

1.1 Computing Equilibria

In a series of breakthrough papers, Daskalakis *et al.* [11] showed that the problem of computing a mixed Nash equilibrium is $PPAD$-Complete. This holds even for the restricted case of only two players and the case of approximate ϵ-Nash equilibria for n-player games [9]. $PPAD$ stands for "Polynomial Parity Argument Directed". It is a complexity class defined on the basis of the parity argument for directed graphs: "If a directed graph has a node whose in-degree and out-degree are unequal, then it must have another such node". The issue is to find such a node. Additionally, $PPAD$-Completeness holds also for *succinct* representations of games, such as graphical [11] or action-graph games [13]. Interestingly enough, there are even non-game-theoretic problems complete for this class (see for example [29]). The complexity class $FIXP$ was introduced by Etessami and Yannakakis; it captures search problems that can be cast as fixed point computation problems for functions represented by algebraic circuits [15]. They show that computing Nash equilibria for 3 or more players (either exact or approximate), is $FIXP$-Complete. A related survey on the computational complexity of computing equilibria is given in [37].

Another important problem in algorithmic game theory is the computation of equilibrium (market clearing) prices in computational markets. The most popular market models are based on either spending or exchange economies. The problem is known to be $FIXP$-Complete for exchange

markets [15] and $PPAD$-Complete [8, 10] for both exchange and spending economies, in the general case. On the other hand, there exist polynomial time algorithms for solving restricted cases of market models, in particular with linear utilities (e.g. the one described in [14]). Besides their importance in the field, computational markets are out of the scope of this thesis and thus will not be further examined.

1.2 This thesis

In this thesis we will discuss games played on graphs. Our analysis will take into account two different graphical game-theoretic models, namely *graphical games* proposed by Kearns *et al.* [28] and *action-graph games* proposed by Bhat and Leyton-Brown [2]. The focus is on the computational aspects of pure Nash equilibria for these models and the role of *treewidth* in such computations. In particular, our effort will be focused on analyzing the time complexity of the problem from the viewpoint of parameterized complexity, when the parameter is the treewidth of the input graph.

1.2.1 Contributions in this thesis

In the subsequent section we will review several known results from the literature that are essential for the completeness for this thesis. For a number of these results we will add some value, especially in terms of careful analysis, in order to clarify that the results of our approach are not not subsumed by any of the previous ones. In detail:

- We provide the NP-Completeness proof of computing pure Nash equilibria of graphical games with focus on how the parameter of treewidth is preserved (Section 6.2.1). This observation will be used when we argue about the existence of polynomial kernels for the problem under consideration (Section 7.5).
- We provide a unified analysis of the approach of [20], in order to discover the exact time complexity of the implied algorithms (Section 6.2.2). Similarly, for the different approach of [12] (Section 6.2.3).

We will argue that our approach improves on the known bounds for the time complexity of computing pure Nash equilibria.

- We will describe a new interpretation of the well-known mapping from graphical to action-graph games with focus on the preservation of treewidth. It will be shown that the mapping cannot be used to achieve tractability results (Section 6.1.1).

- Based on the existing and new results, we will provide a comparison of the computational power of both graphical and action-graph games from the perspective of pure Nash equilibria and treewidth (Section 8.1).

Besides the review and analysis of existing literature, we will describe a number of *new results*. The following list contains the main contributions of this thesis to the scientific community:

- A W[1]-Hardness proof that indicates that the problem of computing pure Nash equilibria for graphical games is not Fixed-Parameter-Tractable for the parameter of treewidth (Section 6.2.5).

- A simple algorithm that solves the problem in linear time on trees (Section 7.1).

- A generalization of the previous algorithm to graphs of bounded treewidth which leads to a polynomial algorithm with improved bounds for graphical games of bounded treewidth (Section 7.2). The algorithm proves that the problem is Fixed Parameter Tractable for games of bounded treewidth and bounded cardinality strategy sets.

- A proof that most likely there does not exist a polynomial kernel for the problem of interest (Section 7.5).

- A method for games of $O(\log n)$-treewidth that exploits a similar approach suggested in [12] and effectively improves on the time complexity (Section 7.6).

- A proof that a sample pure Nash equilibrium can be constructed without additional computational effort. The same holds for a Maximum Payoff PNE (where each individual receives the maximum possible payoff). Details in Section 7.7.

1.2.2 Outline

The remainder of this document is organized as follows. In Chapter 2 we give several preliminary notions that will be needed for the rest of the thesis. First, we give general notions from game theory, graph theory and parameterized complexity. Then we explain succinctly represented games and the necessity of succinct representations from an algorithmic point of view. Definitions of the models of graphical and action-graph games follow, along with illustrative examples and a review of a well-known mapping from graphical to action-graph games. In Chapter 3 we study pure Nash equilibria, and their computational properties, for the two models under consideration. First, we discuss the importance of pure Nash equilibria as models of the rationality of agents and the difference from mixed Nash equilibria. Subsequently, we focus on action-graph games. We give the NP-Completeness proof for the problem of computing pure Nash equilibria and also give an approach for efficiently deciding instances of symmetric action-graph games with bounded treewidth. Similarly, for graphical games we review the NP-Completeness proof and describe two different approaches for solving the problem of pure Nash equilibria computation on games of bounded treewidth. In addition, we give the $W[1]$-Hardness proof for the parameter of treewidth. In Chapter 4 we describe our new approach. The outline is given in the contributions list in the previous section. Finally, in Chapter 5 we recapitulate the contents of this thesis, give our conclusions -including a comparison of graphical and action-graph games, and describe a number of questions that remain unanswered.

2 Preliminaries

In this section we will introduce a number of notions needed to understand the mechanics of multiplayer games. In a multiplayer game $\mathcal{G}$ we have n players and each player $p \in P$ has a finite set of strategies, $St(p)$, with $|St(p)| \geq 2$. The cardinality of the largest strategy set is denoted with $\alpha = \max_{p \in P} |St(p)|$. In contrast to Gottlob *et al.* [20], we do not differentiate between an "action" and a "strategy". Actually, the two terms will be treated equivalently in this thesis, while in [20] a strategy implies a player choosing an action. Thus, when we denote an action as a_p, we do this to clearly indicate that $a_p \in St(p)$ for player $p \in P$. For a non-empty set of players $P' \subseteq P$ a combined strategy or *configuration* C is a set containing exactly one strategy for each player $p \in P'$, i.e. $C = (a_p)_{p \in P}$. The set of all combined strategies of players in P' is denoted as $St(P')$ and thus we write $C \in St(P')$. Moreover, we write $C^{P'}$ to indicate, without explicitly mentioning it, that $C^{P'} \in St(P')$. For a player p, C_p denotes the strategy of player p with respect to configuration C and C_{-p} denotes the configuration resulting when we remove the strategy suggested for p in C. Additionally, for every $a_p \in St(p)$ and $C_{-p} \in St(P \backslash \{p\})$ we denote by $(C_{-p}; a_p)$ the configuration in which p plays a_p and every other player $p' \neq p$ plays according to C. Abusing notation, we use $C \cup \{a_p\}$ to denote the configuration resulting by adding strategy $a_p \in St(p)$ to configuration $C \in St(P')$ where $p \notin P'$. A configuration C is termed *global* if it contains exactly one strategy for each player $p \in P$ ($C \in St(P)$). The global configurations are the possible outcomes of the game. The utility function of a player $p \in P$ is denoted as u_p and in the general case it is defined as $u_p : C^P \to \mathbb{R}$.

Definition 1. *The best response function of a player p is a function $\beta_p : C_{-p} \to 2^{St(p)}$ such that:*

$$\beta_p(C) = \{a_p | a_p \in St(p) \text{ and } \forall a'_p \in St(p) : u_p(C) \geq u_p(C_{-p}; a'_p)\}$$

Intuitively, the $\beta_p(C)$ is the set of pure strategies that maximize the payoff of player p when the players in p's neighborhood play according to C. It follows, that a pure Nash equilibrium (PNE for short) is a global configuration C such that for every player $p \in P$, $C_p \in \beta_p(C_{-p})$. Alternatively:

Definition 2. *A configuration C is a pure Nash equilibrium if for every player p and strategy $a_p \in St(p)$ we have $u_p(C) \geq u_p(C_{-p}; a_p)$.*

2.1 Graph Theory

We define the neighborhood of player $p \in P$ as $\mathcal{N}(p) = \{u \in V | (p, u) \in E\}$. Following Kearns *et al.* [28], for a game $\mathcal{G}$ with set of players P we define the *strategic dependency graph* as the undirected graph $G(\mathcal{G}) = (P, E)$ that has P as its set of vertices and $E = \{(p, q) | p, q \in P \text{ and } p \in \mathcal{N}(q)\}$ as its set of edges. Similarly, we define the *strategic dependency hypergraph* $H(\mathcal{G}) = (P, E)$ that has P as the set of vertices and whose set of hyperedges is $E = \{\{p\} \cup \mathcal{N}(p) | p \in P\}$. These types of graphs will be useful only for the graphical games model in this thesis, and thus we will refer to them as the *graph* and the *hypergraph* of game $\mathcal{G}$, respectively.

Definition 3 ([35])**.** *A tree decomposition of a graph $G = (V, E)$ is a pair $(\{X_i | i \in I\}, T = (I, F))$, where each node $i \in I$ has associated to it a subset of vertices $X_i \subseteq V$, called the bag of i such that*

1. *Each vertex belongs to at least one bag, $\cup_{i \in I} X_i = V$;*
2. *$\forall \{v, u\} \in E$, $\exists i \in I$ with $v, u \in X_i$;*
3. *$\forall v \in V$, the set of nodes $\{i \in I | v \in X_i\}$ induces a subtree of T.*

The width of a tree decomposition $\mathcal{T}$ is $\max_{i \in I} |X_i| - 1$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G. In this thesis, w denotes the treewidth of the graph under consideration (for graph G, $w = twd(G)$).

Definition 4 ([21])**.** *Let $\mathcal{H} = (N, H)$ be a hypergraph. A hypertree decomposition of $\mathcal{H}$ is a triplet $\langle T, \chi, \lambda \rangle$, where $T = (V, E)$ is a rooted tree and χ, λ are labelling functions associating each vertex $v \in V$ with two sets $\chi(v) \subseteq N$ and $\lambda(v) \subseteq H$, such that*

1. *$\forall h \in H$, $\exists v \in V : h \subseteq \chi(v)$*
2. *$\forall n \in N$, the set $\{v \in V | n \in \chi(v)\}$ induces a connected subgraph of T.*
3. *$\forall v \in V, \chi(v) \subseteq \bigcup_{h \in \lambda(v)} h$*

4. $\forall v \in V, \chi(T_v) \cap \bigcup_{v' \in vert(T_v)} \chi(v')$

The width of $\langle T, \chi, \lambda \rangle$ is $\max_{v \in V}\{|\lambda(v)|\}$. The hypertreewidth of a hypergraph $\mathcal{H}$, $hwd(\mathcal{H})$, is the minimum width over all possible hypertree decompositions.

2.2 Parameterized Complexity

We begin with the basic definition of a parameterized and a fixed parameter tractable problem respectively.

Definition 5 ([33]). *A parameterized problem is a language $L \subseteq \Sigma^* \times \Sigma^*$, where Σ is a finite alphabet. The second component is called the parameter of the problem.*

The only parameter we consider in this thesis (treewidth) is integer and therefore we will write $L \in \Sigma^* \times \mathbb{N}$ from now on. For $(x, k) \in L$, the two dimensions of parameterized complexity analysis are constituted by the input size n, that is, $n = |(x, k)|$ and the parameter value k (usually a nonnegative integer).

Definition 6 ([33]). *A parameterized problem L is fixed-parameter tractable if, for all (x, k), it can be determined in $f(k) \cdot n^{O(1)}$ time whether $(x, k) \in L$, where f is a computable function depending only on k.*

The class of decision problems of the form (x, k), that are solvable in time $f(k) \cdot n^{O(1)}$, is denoted as FPT. Moreover, to prove hardness for parameterized problems we need a reducibility concept.

Definition 7 ([33]). *Let (Q, k) and (Q', k') be parameterized problems over the alphabets Σ and Σ'. An fpt-reduction is a mapping $R : \Sigma^* \to (\Sigma')^*$ such that*

- *For all $x \in \Sigma*$ we have $(x \in Q \Leftrightarrow R(x) \in Q')$.*
- *$R(x)$ is computable in time $f(k(x)) \cdot p(x)$.*
- *There is computable function $g : \mathbb{N} \to \mathbb{N}$ such that $k'(R(x)) \leq g(k(x))$, $\forall x \in \Sigma^*$.*

The lower class of fixed-parameter intractability, $W[1]$, can be defined as the class of parameterized problems that are fpt-reducible to Weighted 3SAT. That is, given a 3SAT formula, decide if it has a satisfying assignment of Hamming weight k. In parameterized complexity, the fundamental conjecture is $FPT \neq W[1]$. It is analogous to the $P \neq NP$ assumption, but essentially weaker [33]. A parameterized problem is $W[1]$-Hard if it lets Weighted 3SAT reduce to it by a fpt-reduction. For example, the well-known problems Clique and Independent Set are both $W[1]$-Hard, parameterized by the size of the solution set.

Furthermore, we give a number of more involved notions from parameterized complexity. These notions will be used in Section 7.5 to prove the non-existence of a polynomial kernel for the problem of interest. A *polynomial kernel* is a data reduction algorithm that replaces the input instance by a reduced instance with size polynomially bounded in the parameter.

Definition 8 (And-Composition [3])**.** *An and-composition algorithm for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that takes as input a sequence $((x_1, k), \ldots, (x_t, k))$, with $(x_i, k) \in \Sigma^* \times \mathbb{N}^+$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^{t} |x_i| + k$ and outputs $(y, k') \in \Sigma^* \times \mathbb{N}^+$ with:*

- *$(y, k') \in L \iff (x_i, k) \in L$ for all $1 \leq i \leq t$*
- *k' is bounded by a polynomial in k*

Definition 9 (And-Distillation Conjecture [3])**.** *Let R be an NP-Complete problem. There is no algorithm D, that gets as input a series of m instances of R, and outputs one instance of R, such that*

- *If D has as input m instances, each of size at most n, then D uses time polynomial in m and n, and its output is bounded by a function that is polynomial in n.*
- *If D has as input instances $x_1, \ldots, x_m$, then $D(x_1, \ldots, x_m) \in R$ if and only if $\forall_{1 \leq i \leq m} x_i \in R$.*

Finally, let (Q, k) and (Q', k') be parameterized problems. A *polynomial time and parameter transformation* is a polynomial time many-one transformation from Q to Q', with the additional condition that $k' \leq p(k)$ for a polynomial $p : \mathbb{N} \to \mathbb{N}$ [6].

3 Succinctly representable multiplayer games

The standard representation of games is the *normal form* representation. Traditionally, 2-player games are presented by way of a matrix, that includes the actions available to each player and the corresponding payoff of each outcome. For multi-party game theory this representation includes all perceptible and conceivable strategies and the corresponding payoffs for each player. The computational problem in this thesis is computing a pure Nash equilibrium. One can easily observe that computing such an equilibrium, when the input game is in normal form, would take time linear in the size of the representation, simply by going through all possible configurations. The problem may be thought of as computationally easy, but it actually is not. The impracticality of normal form representation lies in the multi-dimensionality of the input matrices (curse of dimensionality). In fact, the input length is exponential; for n players we need to describe $n\alpha^n$ utility values.

A succinctly representable game, or simply *succinct game*, is a game whose size of representation is much smaller than its normal form. Such representations are studied since, in real-world applications, games have structured utility functions. The aim of succinct game representations is to effectively capture this structure. The most ancient form of succinct games are the *symmetric games* which were already studied by von Neumann and Nash. A game is symmetric if one player's payoffs can be expressed as a transpose of the other player's payoffs. Famous examples used in literature include the prisoner's dilemma, the game-of-chicken and the battle-of-sexes game. In such games players are identical, in the sense that a player's utility depends only on her strategy and on the number of other players that have chosen each of available the strategies. Therefore, to describe a symmetric game with n players we need α^s numbers, which can be significantly better than α^n of normal form.

There are several types of succinct games found in literature, many of which have been introduced recently. For the computer science community the quest for a succinct representation that captures well-structured games

and that has nice computational properties, is more active than ever. Besides action-graph and graphical games that will be described extensively in the next sections, other important classes of succinct games include:

Circuit games. A very flexible representation of succinct games is obtained by describing each player by a polynomial-time bounded Turing machine, which takes as input the actions of all players and outputs the player's utility. Such a machine is equivalent to a Boolean circuit and such games are known as circuit games. Deciding the existence of pure Nash equilibria in such games is NP-Complete in the general case [38].

Anonymous games. In such games the utilities do not depend on the identity of the players making these choices. The players do not distinguish between other players but are only interested in how many other players chose which strategy. Note that this is more general than symmetric games: each player can evaluate the situation in her own, individual way. Anonymous games are of great interest since they capture important aspects of auctions and markets. Deciding the existences of pure Nash equilibria for anonymous games is NP-Complete [7].

Congestion games. Games that have to do with allocation of resources across a network of agents. This kind of games was introduced by Rosenthal as a class of games that always contains a pure Nash equilibrium [36]. The strategies of each player are subsets of the available resources and each resource has a delay that is a function of the number of players that use it. The utility of the player is the summation of the delays of the resource she chose. Even though the existence of a pure Nash equilibrium is guaranteed, finding such an equilibrium is PLS-Complete in the general case [16].

4 Graphical Games

In this section we will discuss the *graphical game* model introduced by Kearns *et al.* in [28]. A graphical game is played on an undirected graph $G = (V, E)$, where each player is represented by a vertex.

Definition 10 ([28]). *A graphical game is a pair $(G, \mathcal{M})$, where $G = (V, E)$ is an undirected graph and $\mathcal{M}$ is a set of $n = |V|$ local matrices. For any combined strategy C, the local game matrix $M_p \in \mathcal{M}$ specifies the payoff $M_p(C)$ for player $p \in V$, which depends only on the actions taken by p and the players in $\mathcal{N}(p)$.*

Graphical games are potentially succinct representations of games that are more compact than standard normal form. Instead of requiring a number of parameters that is exponential in n, a graphical game needs a number of parameters that is exponential in $\Delta + 1$, where Δ is the largest degree and $\Delta+1$ the size of the largest neighborhood. For a player $p \in P$ a local matrix holds payoff values for any possible configuration of the vertices in $\mathcal{N}(p)$. Therefore, $|M_p| = |St(p)| \cdot |St(\mathcal{N}(p))| \leq \alpha^{\Delta+1}$. When players are directly influenced by a number of others, much smaller than the overall population size, then the graphical game representation is dramatically smaller than the normal form. Of course, any normal form game can be represented as a graphical game played on a complete graph. Therefore, the graphical game representation is only meaningful when the interactions between players are limited. We also note that even though the payoffs of a player are determined according to the behavior of her neighborhood, a (pure) Nash equilibrium still requires *global* coordination over all players. Even the players that are not connected in the game graph, with their choices, may change indirectly the incentives of all other participants. One of the computational challenges imposed by graphical games is how local influences propagate to determine global equilibrium outcomes [28].

Furthermore, note that the interests of players are necessarily symmetric for graphical games on undirected graphs. That is, for any pair of players p_1, p_2: $p_1 \in \mathcal{N}(p_2)$ if and only if $p_2 \in \mathcal{N}(p_1)$. On the other hand, there is also the choice of representing the game structure using a directed graph, which also takes into account that the dependencies between payoffs do not have to be symmetric. Although, directed graphs arguably do not help with efficiently computing equilibria [20]. In this thesis we follow the original model of Kearns *et al.* [28] and use undirected graphs to represent the game

structure. Finally, we denote the size of the collection of matrices as

$$|\mathcal{M}| = \sum_{p \in V} |M_p|$$

An illustrative example. Five coworkers, who live in the same block, want to carpool to work using two company cars. Each one has the choice to either drive (d) or be driven to work by someone else (e). Alice (A) prefers to not drive, be with her husband Bob(B) who does not know anyone else, and does not mind the company of Daniel(D). The latter wants to be with Chris(C) and does not care if he drives. Christine would like to be with her friend Eva(E) and prefers to not drive. Eva cannot stand Alice but enjoys driving and the company of Chris. The dependency graph of this game is depicted in Figure 1.

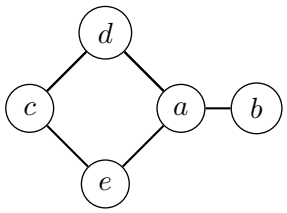


Figure 1: The graph of the game described in the example.

In this game when player A prefers player B it means that A wants to do the opposite of the action of B and thus they are more likely to end up in the same car. Below, we give the payoff matrices for Alice and Eva; the rest can be defined analogously.

A	B_dD_d	B_dD_e	B_eD_d	B_eD_e
d	0	0	0	0
e	1	1	1	-1

E	A_dC_d	A_dC_e	A_eC_d	A_eC_e
d	1	1	0	1
e	1	0	1	0

For the above game we have a significant save in size by using the graphical game representation. Let $|\mathcal{G}|$ denote the size of the game description. By representing $\mathcal{G}$ as a graphical game, $|\mathcal{G}| = 3 \cdot 2^3 + 2^4 + 2^2$. The normal-form would require size $5 \cdot 2^5$. Thus, we save $160 - 44 = 116$ values with graphical games.

The complexity of the best response function. The best response function is in the core of the algorithms presented subsequently; thus, it is crucial to characterize the time complexity needed to compute it for graphical games. Let $\mathcal{G} = (G, \mathcal{M})$ be a graphical game and $p \in P$ a player of the game. In addition, let $P' = \mathcal{N}(p) \subseteq P$ and $C \in St(P')$. To compute $\beta_p(C)$ we need to perform $|St(p)|$ steps; that is, to perform a pass of the column of M_p corresponding to configuration C. Now, let $C' \subset C$. To compute $\beta_p(C')$ we have to read the columns (configurations) that adhere to C' and also contain all possible actions $St(v)$ for each player $v \in \mathcal{N}(p)$ such that $v \notin C'$. The worst case is when C' contains information only for one player v (thus, $C' = C_v$). Then, computing $\beta_p(C')$ takes

$$\frac{|M_p|}{|St(v)|} < |M_p|$$

computational steps in the worst case.

5 Action Graph Games

In this section we introduce the notions relevant to the action-graph game model introduced by Bhat and Leyton-Brown [2]. The central notion, here, is the action-graph.

Definition 11. *An action graph $G = (S, E)$ is a directed graph where:*

- *S is the set of nodes. Each node $s \in S$ represents an action and S the set of distinct actions. For each agent i, S_i is the set of actions available to i and $S_i \subseteq S$. Agents' actions may partially or completely overlap.*
- *E is a set of directed edges, with self-edges allowed. The notion of neighborhood here has to take into account that the graph is directed. Thus, for action s, $\mathcal{N}(s) = \{s' \in S | (s', s) \in E\}$.*

Given an action, an action graph and a set of agents, we can define a configuration. For an action graph game, that is, a feasible disposition of agents over the nodes in the action graph. Let $\mathcal{C}$ denote the set of configurations of agents over actions.

Definition 12. *Given an action graph $G = (S, E)$, a configuration C is an $|S|$-tuple of integers $(C(s))_{s \in S}$, where $C(S)$ denotes the number of agents that choose action $s \in S$. For a subset of actions $S' \subset S$, let $C^{S'}$ denote the restriction of C over S', i.e. $C^{S'} = (C(s))_{s \in S'}$. Similarly, let $\mathcal{C}^{S'}$ denote the set of restricted configurations over S'.*

Then, an action graph game (AGG for short) can be defined as follows:

Definition 13. *An action graph game is a tuple $\langle p, \mathcal{S}, G, u \rangle$ where*

- *$P = \{1, \ldots, n\}$ is the set of agents.*
- *$\mathcal{S} = \prod_{i \in P} S_i$ is the set of action profiles, where $\prod$ is the Cartesian product and S_i is agent i's set of actions.*
- *$G = (S, E)$ is the action graph, where $S = \bigcup_{i \in P} S_i$ is the set of distinct actions.*
- *u is a $|S|$-tuple $(u^s)_{s \in S}$, where each $u^s : \mathcal{C}^{\mathcal{N}(s)} \to \mathbb{R}$ is the utility function for s. Semantically, $u^s(C^{\mathcal{N}(s)})$ is the utility of an agent that choses s when the configuration over $\mathcal{N}(s)$ is $C^{\mathcal{N}(s)}$.*

Finally, we give two definitions of different types of symmetry that can appear in this model. The first is total symmetry, where all players have the same action set, and the second is selective symmetry, where the players can be partitioned in categories with each category having the same action set.

Definition 14. *An action game is symmetric if all players have identical action sets, i.e. if $S_i = S$, $\forall i \in P$.*

Definition 15. *An action game is k-symmetric if there is a partition $\{P_1, \ldots, P_k\}$ of P such that $\forall l \in \{1, \ldots, k\}$ if $i, j \in P_l$ then $S_i = S_j$.*

5.1 Some interesting properties

First we would like to point out that action graph games are fully expressive. That is, any (symmetric) game can be represented as an (symmetric) AGG. Intuitively, two types of structure can be effectively captured with the action-graph game representation:

- A game's *anonymity* structure can be capture with shared actions. Each agent cares only about the number of players that play each action and not the identities of those players.

- AGGs allow for *context-specific independencies* of utilities with the lack of edges between nodes. The configuration over actions not in $\mathcal{N}(s)$ does not affect the utility of any player that plays s. Note that $\mathcal{N}(s)$ is defined only on incoming edges.

As happens with graphical games the size of an action-graph game representation is dominated by the size of its utility functions, since the latter is worst-case exponential. Then, for an AGG $\mathcal{G}$ we let the size of the representation $||\mathcal{G}||$ be equal to the number of utility values the representation needs to be described:

$$||\mathcal{G}|| = \sum_{s \in S} |\mathcal{C}^{\mathcal{N}(s)}|$$

It is argued in [26] that this is the number of ordered combinatorial compositions of $n-1$ into $|\mathcal{N}(s)|+1$ non-negative integers

$$||\mathcal{G}|| \leq |S| \binom{n-1+\mathcal{I}}{\mathcal{I}} = |S| \frac{(n-1+\mathcal{I})!}{(n-1)!\mathcal{I}!}$$

where $\mathcal{I} = \max_{s \in S} |\mathcal{N}(s)|$ is the maximum in-degree of the action graph G. In addition, if $\mathcal{I}$ is bounded by a constant, $||\mathcal{G}|| = O(|S|n^{\mathcal{I}})$ which is polynomial in n.

Proposition 1. *Given an AGG, the number of payoff values stored by its utility functions is at most* $|S| \frac{(n-1+\mathcal{I})!}{(n-1)!\mathcal{I}!}$. *If* $\mathcal{I}$ *is bounded by a constant then the number of payoff values is* $O(|S|n^{\mathcal{I}})$.

An illustrative example. The example we describe in this section is taken from [25]. Suppose there are n agents who are interested in opening a business that can be located in either side of a road of length m (there are m blocks for businesses in each side of the road). The blocks are considered to be non-exclusive and thus multiple agents are allowed to choose the same block. The payoff of each agent depends on the number of agents who chose the same block or chose one of the adjacent ones. This game can be

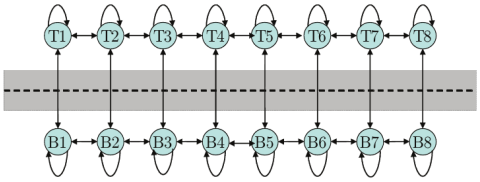


Figure 2: A road game with $m = 8$, taken from [25].

represented succinctly as a symmetric AGG, whose action-graph is depicted in Figure 2. Observe that each node has at most four incoming edges and thus for road length m the respective AGG representation holds $O(|S|n^4) = O(2mn^4)$ payoff values. On the contrary, a graphical game representation would induce a clique since any pair of agents potentially affect each other's payoffs. Therefore, a graphical game would have as much space complexity as the normal form since it is unable to capture the anonymity structure of the road game example.

Mapping graphical games to AGGs. It is well-known that graphical games can be represented as AGGs by replacing each node i of the graphical game by distinct set of nodes S_i representing the action set of agent i [26]. If in the graphical game there exists the edge (i, j), in the AGG we take a number of edges such that $\forall s_i \in S_i$ and $\forall s_j \in S_j$, $s_i \in \mathcal{N}(s_j)$. The resulting AGG representation is as compact as the original graphical game representation. In the next section we are going to argue on why this mapping is not profitable in computing pure Nash equilibria for graphical games.

6 Pure Nash Equilibria

Mixed Nash equilibria are guaranteed to exist but are very fragile as models of behavior and rationality. On the other hand, *pure Nash equilibria* are intuitive and comprehended in a straightforward manner. A complete definition of how a player will play a game is provided by a pure strategy. It determines how a player would behave in any situation she could face. This determinism comes with a trade-off: pure Nash equilibria are *not* guaran-

teed to exist. Following, we give two examples, famous in the game-theoretic literature.

	H	T
H	$+1, -1$	$-1, +1$
T	$-1, +1$	$+1, -1$

(a) Matching Pennies

	C	D
C	$+3, +3$	$0, +5$
D	$+5, 0$	$+1, +1$

(b) Prisoner's Dilemma

Figure 3: Two well-known examples of 2-player games: (a) does not have a PNE; (b) has a single PNE, (D, D).

The examples of Figure 3 demonstrate the possible lack of pure Nash equilibria in the input game. This possibility is increased for multiplayer games with complex interactions. We will argue in the following sections that deciding the existence of a pure Nash equilibrium is an NP-Complete problem, in the general case, for both graphical and action-graph games. Of course, the problem is computationally trivial for normal form games, due to the exponentially large input size.

The rest of this chapter is outlined in the following manner: First, we review the complexity results for the problem of deciding the existence of a pure Nash equilibrium for action-graph games. The problem is shown to be NP-Complete. Then, we discuss restrictions that lead to tractable classes of action-graph games. Subsequently, we review the complexity of deciding the existence of PNE for graphical games. Again, the problem is shown to be NP-Complete. In addition, two different approaches are described for graphical games. The first is by a mapping to Constraint Satisfaction problems and the second by a mapping to Markov Random Fields. Both lead to polynomial algorithms for graphical games of bounded treewidth. A different characterization, for broader classes of graphical games, is also described. Finally, we parameterize the problem of interest by treewidth and prove that it is $W[1]$-Hard. The latter result is originally presented in this thesis.

6.1 Action-graph games

In this section we will describe the computational properties of pure Nash equilibria for action-graph games. First, we argue that the problem is NP-Complete both in the general and the symmetric case. Then, we describe an approach that leads to the tractability of symmetric action-graph games of bounded treewidth. Observe that in the asymmetric case the hardness holds even for instances with treewidth 1 (the input graph is a directed tree).

6.1.1 NP-Completeness

Computing PNE for Action Graph Games has been shown NP-Complete, both in the general and the symmetric case [13]. The proofs of the two subsequent results rely on reductions from the NP-Complete problem CIRCUITSAT, were presented in [13] and follow an approach that was first employed in [38]. It is well-known in complexity theory that deciding satisfiability of a circuit is NP-Complete, even when all gates have maximum degree 3.

Theorem 1. *Given a circuit C, that consists of AND, OR and NOT gates, with maximum degree 3 (in plus out degree), it is NP-Complete to decide if C is satisfiable.*

In the reductions described below, given a circuit C, we construct an AGG A_C that corresponds to C in the sense that pure Nash equilibria of A_C map to valid circuit evaluations. Two extra agents are added, that have a simple pure strategy equilibrium if C evaluates to true. In the negative case they are forced to play a game of *matching pennies* that does not have a pure equilibrium. In this section we denote a configuration with D to avoid confusion with a circuit C.

The Copy Gadget Before we proceed to the NP-Completeness result we will describe a *copy gadget*. This is used to simulate action graph games of arbitrary treewidth by games of treewidth 1 and was first described in [13] to attain hardness results even in the restricted case where the input graph has treewidth 1. Using the gadget, we create several copies of each

player, but only one copy of each edge relating different players (specifically the pure strategies corresponding to those players). This results in a very "sparse" simulation whose treewidth can be controlled. Given an AGG A and an agent i with strategy set $S_i = \{f_i, t_i\}$ used only by player i, the copy gadget will add two additional players c which is the "copy" and a which is an auxiliary player. The inclusion of a allows the strategies of player i to be disconnected from those of player c.

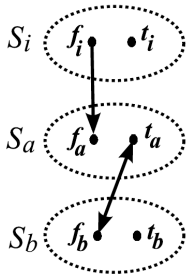


Figure 4: A depiction of the copy gadget taken from [13].

Definition 16. *Given an AGG $A = \langle P, \mathcal{S}, G, u\rangle$ and an agent i with two strategies $\{f_i, t_i\}$ such that no other player may play strategy f_i, we create AGG $A' = \langle P', \mathcal{S}', G', u'\rangle$ via the addition of a copy gadget on i as follows:*

- $P' := P \cup \{a, c\}$
- $Sc' := (S_1, \ldots, S_{|P|}, S_a, S_c)$*, where* $S_a = \{f_a, t_a\}$ *and* $S_c = \{f_c, t_c\}$*.*
- G' *consists of the graph* G *with the additional vertices* f_a, t_a, f_c, t_c *and the directed edges* (f_i, f_a)*,* (t_a, f_c)*,* (f_c, t_a)*.*
- u' *and* u *are identical for all strategies in* $S'\backslash\{S_a \cup S_c\}$*. Moreover, for configuration* D *we have* $u'(f_a) = D(f_i)$*,* $u'(t_a) = D(f_c)$*,* $u'(f_c) = 1 - 2D(t_a)$ *and* $u'(t_c) = 0$*.*

An illustration of the copy gadget can be found in Figure 4. Note that the incentives of the players are set such that in any pure Nash equilibrium if i plays f_i then a plays f_a and c plays f_c. If i plays t_i then a is indifferent between f_a or t_a but can only play t_a and consequently c plays t_c in a pure Nash equilibrium: when c plays f_c then a has an incentive to play f_c but then a would have the incentive to play t_a. Below we describe two theorems

from [13] that settle the computational complexity of deciding whether a pure Nash equilibrium exists in a given action graph game.

Theorem 2. *Deciding the existence of a PNE for AGGs with strategy graph G_A is NP-Complete even if $twd(G_A) = 1$, and G_A has constant degree.*

Proof. Membership in NP is clear: given any pure strategy profile we can verify whether it is a Nash equilibrium in polynomial time and strategy profiles are polynomial in the number of available actions. Given a circuit C we construct the associated AGG $A_C = \langle P, \mathcal{S}, G, u \rangle$ as follows:

- $P := \{1, \ldots, n, p_1, p_2\}$, where n is the number of gates in C and the gate corresponding to n is the output gate.
- $\mathcal{S} := ((f_1, t_1), \ldots, (f_n, t_n), (f_{p_1}, t_{p_1}), (f_{p_2}, t_{p_2}))$.
- If a pair of gates i, j is such that the output of i is an input to j, then G has the edges (f_i, f_j) and (f_i, t_j). Moreover, we add the edges $(f_n, f_{p_1}), (f_n, t_{p_1}), (f_n, f_{p_2})$, (f_n, t_{p_2}) and the edges $(f_{p_1}, f_{p_2}), (f_{p_1}, t_{p_2})$, $(f_{p_2}, f_{p_1}), (f_{p_2}, t_{p_1})$.
- The utility function u is defined in the following manner:
  - if agent i corresponds to an input gate, both strategies f_i, t_i have utility 0;
  - if agent i corresponds to a non-input a gate of C, the payoff of strategy t_i is 1 or 0 according to whether *true* is the correct output value of gate i given the values corresponding to the strategies played by agents with neighboring actions. Similarly for the payoff of strategy f_i.
  - If $D(f_n) = 0$ (there is input for which C evaluates to true), then f_{p_1} and t_{p_1} have utility 0. Otherwise the utility of p_1 is 1 when $D(f_{p_1}) = D(f_{p_2})$ and 0 otherwise.
  - The utility of p_2 is 1 if $D(f_{p_1}) \neq D(f_{p_2})$ and 0 otherwise.

Claim 1. *Using the construction above, A_C has a pure Nash equilibrium if and only if C is satisfiable.*

Proof. If C is satisfiable then there is a pure strategy profile where agent n plays t_n such that agents $1, \ldots, n$ cannot improve their utility by deviating from their strategies. In addition, p_1 will be indifferent between her

strategies, and p_2 will play the opposite of p_1. This profile is a pure Nash equilibrium.

If C is not satisfiable, then any pure strategy profile that is an equilibrium for agents $1, \ldots, n$ will have $D(f_n) = 1$, and thus p_1 will have the incentive to agree with p_2 who will have the incentive to disagree. This consists of a matching pennies games between p_1 and p_2 and does not have a pure Nash equilibrium. □

To complete the proof, we will apply the copy gadget to each agent of A_C to yield a game A'_C that has action graph of treewidth 1. We obtain A'_C by making three copies of each player $i \in P$ via the copy gadget. For each i we add agents i^1, i^2, i^3 with $S_{i^k} = \{f_i^k, t_i^k\}$. In addition, we replace the (at most three) outgoing edges of f_i that are not part of the copy gadgets: each edge of the form (f_i, f_j) is replaced by an edge (f_i^k, f_j) with each f^k having at most one outgoing edge. The utility function u is modified analogously so as to have the action f_j depend on f_i^k rather than f_i. Similar edge replacements happen for the outgoing edges of t_i. The copied strategies f_i^k, t_i^k are disconnected from the original strategies f_i, t_i and thus the longest path in G has length at most 4 (from an auxiliary vertex, to a copy vertex, to a non-copy vertex, to an auxiliary vertex). Moreover, G has maximum degree 6 (3 that originally had from the transformation from CIRCUITSAT and another 3 from applying the copy gadget) and treewidth at most 1. Finally, from Definition 16, the representation size of A'_C is at most a constant larger than that of A_C. □

At this point, let us turn back to the mapping from graphical to action-graph games that we described earlier in Section 5. Since every action node in each cluster is connected to all the action nodes in the clusters that represent neighboring players of the original graphical game, every node has at most α time the degree it had originally. Therefore, the treewidth of the resulting graph, say w', is at most $w' \leq \alpha w$, where w is the treewidth of the original graph. This means that for game instances with bounded α, this reduction preserves treewidth linearly. Since each player of the graphical game has her own cluster of nodes in the action-graph, the game is necessarily asymmetric. In light of Theorem 2 we cannot expect to compute efficiently pure Nash equilibria of graphical games with bounded treewidth

via this mapping. Different, efficient approaches will be discussed in the next chapters; for the time being, let us characterize the complexity of computing PNE for a symmetric AGG.

Theorem 3. *Deciding the existence of a PNE for symmetric AGGs is NP-Complete even if the strategy graph G_A has bounded degree.*

Proof. We use the same technique as for Theorem 2. To make A_C symmetric while retaining the same number of agents, we allow them to pick any of the strategies. The strategy graph G is modified by adding edges $(f_x, t_x), (t_x, f_x), (f_x, f_x), (t_x, t_x)$ for each player $x \in P$. In addition, we extend the utility function u such that if $D(f_x) + D(t_x) > 1$ then strategies f_x and t_x have utility -1. Thus, in any pure Nash equilibrium $D(f_x) + D(t_x) = 1$ (only one player plays either f_x or t_x and thus the value of the corresponding gate is correctly either true or false). The reasoning of Theorem 2 applies to complete the reduction. □

Observe that the copy gadget is not applicable in the case of symmetric AGGs and thus the NP-Completeness result does not necessarily hold for action-graphs of bounded treewidth. In fact, we will describe an approach that computes pure Nash equilibria of symmetric AGGs in polynomial time for action-graphs of bounded treewdith.

6.1.2 Tractable Cases

The results in this section, about tractable cases of action-graph games, were first described by Jiang and Leyton-Brown in [25].

Theorem 4. *Deciding the existence of a PNE in a symmetric AGG with bounded $|S|$ is in P.*

Proof. If $|S|$ is bounded, then the number of possible configurations $\binom{n+|S|-1}{|S|-1} = O(n^{|S|-1})$ is polynomial. Thus, a polynomial algorithm is to check all configurations. □

This can be easily extended for k-symmetric games.

Lemma 1. *Deciding the existence of PNE in a k-symmetric AGG with bounded $|S|$ and bounded k is in P.*

Proof. If $|S|$ is bounded, for each $l \in \{1, \ldots, k\}$ the number of distinct D_l is $\binom{|N_l|+|S^l|-1}{|S^l|-1} = O(|N_l|^{|S^l|-1})$. Therefore, the number of distinct k-configurations is $O(n^{k(|S|-1)})$ which is polynomial for bounded k. Checking if a k-configuration is a PNE also takes polynomial time. □

Consequently, the next theorem follows directly from Lemma 1 and shows that the full class of AGGs with bounded $|S|$ can be solved efficiently.

Theorem 5. *Deciding the existence of a PNE in an arbitrary AGG with bounded $|S|$ is in P.*

Proof. Observe that any AGG $\mathcal{G}$ is k-symmetric by definition, where k is the number of distinct action sets. Since $S_i \subseteq S$ for all i, the number of distinct nonempty action sets is at most $2^{|S|} - 2$. Thus, the number of distinct action sets is bounded since $|S|$ is bounded. We conclude that $\mathcal{G}$ is k-symmetric with bounded k and Lemma 1 applies. □

To generalize the simple results above, in [25] is given a dynamic programming approach which constructs pure Nash equilibria of the game from first considering restricted parts of the action-graph and then synthesizing the information. The algorithm uses similar techniques to [12]: it runs on a tree decomposition of the primal graph of the game; more details on the primal graph will be given in Section 6.2.3.

Theorem 6. *Deciding the existence of PNE in a symmetric AGG with bounded treewidth is in P.*

The algorithm hidden behind this theorem is quite technical and we will not present the proof in this thesis. An analysis, including the algorithm in full detail, can be found in [25]. The time complexity of the algorithm is $O(||\mathcal{G}||^{w+1})$, which is polynomial for bounded treewidth. Therefore, the road game example, described in the previous chapter, can be solved in polynomial time since road games have treewidth 2 for any value of m.

Finally, it is discussed in [25] that after the dynamic programming approach is finished, one can construct a pure Nash equilibrium using a top-down pass of the tree decomposition. The number of PNE could be exponential in the representation size $||\mathcal{G}||$ but the resulting set of tables, after

the bottom-up pass, contain information for *all* PNE and thus constitute a succinct description of the set of PNE. More details on succinct descriptions and this method will be explained in Section 7.7.

6.2 Graphical Games

Given a graphical game $\mathcal{G}$ and a global strategy profile C we can decide in polynomial time whether C is a pure Nash equilibrium. This holds for any approximation scheme and any graph input [38]. Since there are only a polynomial number of players and each has a polynomial number of strategies, a strategy can be evaluated in polynomial time by checking if each player's strategy with respect to C is a best response to the strategies of her neighbors. In the subsequent sections we will prove that computing PNE for graphical games is NP-Complete, argue that the problem becomes tractable for graphs of bounded treewidth and describe a complexity-theoretic characterization. Finally, we argue that the problem is $W[1]$-Hard, with treewidth as the parameter, by giving an original proof.

6.2.1 NP-Completeness

Theorem 7 ([20])**.** *Deciding whether a graphical game $(G, \mathcal{M})$ has a PNE is NP-complete even for the restricted cases of 3-bounded neighborhood and fixed number of actions.*

Proof. Membership in NP should is clear. Given a global configuration C we can verify if it is a pure Nash equilibrium of G by checking for each player p and action $a \in St(p)$ that choosing this action does not lead to an increment of her payoff. This can done in polynomial time.

For the hardness part we use a reduction from the NP-Complete problem 3SAT. The input consists of a Boolean formula in conjunctive normal form $\Phi = c_1 \wedge \ldots \wedge c_m$, over the variables $X_1, \ldots, X_n$, where each clause contains at most three distinct variables and each variable occurs in at most three clauses. W.l.o.g assume that Φ contains at least one clause per variable.

Given formula Φ, we define a graphical game that is played on the incidence graph of Φ. The incidence graph G_Φ is the bipartite graph where each variable and clause appears as a vertex and each variable vertex connects to all the vertices that correspond to the clauses that it appears. In our case

$G_\Phi = (P_v \cup P_c, E)$ is the game graph. For each player $c \in P_c$, $\mathcal{N}(c)$ is the set of players corresponding to the variables in clause c, and for each player $v \in P_v$, $\mathcal{N}(v)$ is the set of players corresponding to the clauses in which v occurs. The set of available actions for all the players is $\{t, f, u\}$, in which t can be interpreted as the value *true* and f as the value $false$ for variables and clauses.
Let x be a global strategy. For each player $c \in P_c$, u_c is such that

(i) $u_c(x) = 3$ if c plays t, and all of her neighbors play an action in $\{t, f\}$ in such a way that at least one of them makes the corresponding clause true;

(ii) $u_c(x) = 2$ if c plays u, and all of her neighbors play an action in $\{t, f\}$ in such a way that none of them makes the corresponding clause true;

(iii) $u_c(x) = 2$ if c plays f and there exists $v \in \mathcal{N}(c)$ such that v plays u;

(iv) $u_c(x) = 1$ in all other cases.

For each player $v \in P_v$, her utility function u_v is such that

(v) $u_v(x) = 3$ if v plays an action in $\{t, f\}$ and all of her neighbors play an action in $\{t, f\}$;

(vi) $u_v(x) = 2$ if v plays u and there exists $c \in \mathcal{N}(v)$ such that c plays u;

(vii) $u_v(x) = 1$ in all other cases.

Claim 2. Φ *is satisfiable if and only if* $\mathcal{G}$ *admits a pure Nash equilibrium.*

Let σ be a satisfying truth assignment of formula Φ. Consider the global strategy x of $\mathcal{G}$ where each $v \in P_v$ chooses the action according to σ and each $c \in P_c$ chooses t. Then, all players receive payoff 3 according to rules (i) and (v) above. Since 3 is the maximum payoff, x is a pure Nash equilibrium.

Let x be a global configuration that is a pure Nash equilibrium. Using a series of properties, we will prove that x corresponds to a satisfying assignment of Φ.

P_1: *A strategy x in which a player $v \in P_v$ plays u cannot be a pure Nash equilibrium.* For the sake of contradiction assume that x is a PNE. Then, all $c \in \mathcal{N}(v)$ will have the incentive to play f and receive payoff 2 from rule (iii) and v would receive 1 from rule (vii). However, v

would prefer to play an action in $\{t, f\}$ and increase her payoff to 3 according to rule (v). Thus, x cannot be a PNE.

P_2: *A strategy x in which a player $c \in P_c$ plays u cannot be a pure Nash equilibrium.* Indeed, if there is such a player $c \in P_c$ then each player $v \in \mathcal{N}(c)$ would choose to play u in order to get payoff 2 according to rule (vi). Thus, x cannot be a PNE by property P_1.

P_3: *A strategy x in which all players play an action in $\{t, f\}$ and the corresponding truth assignment makes a clause c false cannot be a pure Nash equilibrium.* In that case c would have an incentive to play u and receive 2 for payoff according to rule (ii). Therefore, x cannot be a PNE by property P_2.

P_4: *A strategy x in which all players play an action in $\{t, f\}$ and there exists a player $c \in P_c$ that plays f cannot be a pure Nash equilibrium.* Since x is a PNE and all players play an action in $\{t, f\}$, by property P_3 the truth assignment that corresponds to x satisfies each clause c. Then, if a player c chooses to play f it contradicts x being a PNE because c could play t and increase her payoff to 3 according to rule (i).

It follows from properties P_1 to P_4 that every pure Nash equilibrium of $\mathcal{G}$ should be a strategy where all players $v \in P_v$ play an action in $\{t, f\}$ and all players $c \in P_c$ play t receive a payoff of 3. This concludes the proof of the claim.

Finally, observe that the matrices representing the entries of the utility functions (rules (i)-(vii) above) can be built in polynomial time from Φ. The assumptions about the structure of Φ result in a game where each player depends on at most 3 other players and therefore each matrix M_v, for $v \in P_v$, and M_c, for $c \in P_c$, has at most a constant number of entries (specifically, 3^4). Thus, the reduction takes at most time polynomial in the size of the input. □

6.2.2 PNE via Constraint Satisfaction Problems

The first attempt to identify classes of instances of graphical games that allow for efficient computation of pure Nash equilibria was also reported by Gottlob *et al.* in [20]. In their paper, they prove with a series of theorems

that the question of existence of a pure Nash equilibrium in a graphical game can be answered in polynomial time for graphs of *bounded treewidth*. The first step in the sequence of results is to fix a fundamental relationship between PNE of graphical games and Constraint Satisfaction Problems (CSP for short).

A CSP instance is a triple $I = (Var, U, C)$, where Var is a finite set of variables, U is a finite domain of values and $C = \{C_1, C_2, \ldots, C_q\}$ is a finite set of constraints. A constraint C_i consists of a pair (S_i, r_i), where S_i -called the constraint scope- is a list of variables of length m_i and r_i is an m_i-ary relation over U. In other words, the tuples of r_i indicate the allowed combinations of simultaneous values for the variables S_i. A solution of such a problem is an assignment $\theta : Var \to U$, such that for each $1 \leq i \leq q$, $S_i\theta \in r_i$.

Following, we will review a mapping from a graphical game to a CSP instance, that was demonstrated in [20], such that a solution of the latter implies a PNE of the former and vice versa. In CSPs associated with games the variables of the CSP instance correspond to the players of the game. Let $\mathcal{G} = (G, \mathcal{M})$ be a graphical game and $p \in P$ a player. Define the Nash constraint $NC(p) = (S_p, r_p)$ as follows: The scope S_p consists of players in $p \cup \mathcal{N}(p)$ and the relation r_p contains exactly all combined strategies $x \in St(p \cup \mathcal{N}(p))$, such that there is no $a_p \in St(p)$ with $u_p(x) < u_p(x_{-p}[a_p])$. Therefore, for each pure Nash equilibrium $x \in St(P)$ of $\mathcal{G}$, $x \cap St(S_p)$ is in r_p.

The CSP instance that corresponds to game $\mathcal{G}$ is denoted by $CSP(\mathcal{G})$ and is the triple (Var, U, C), where $Var = P$, the domain U contains all the possible actions of all players, $St(P)$, and C is the set of Nash constraints for the players of $\mathcal{G}$, $C = \{NC(p) | p \in P\}$. The following theorem, which is a merge of two theorems presented in [20], explains the fundamental relationship between graphical games and Constraint Satisfaction instances.

Theorem 8 (Merge of Theorems 4.3, 4.4 [20])**.** *Given a graphical game $\mathcal{G}$, $CSP(\mathcal{G})$ can be computed in polynomial time. In addition, a strategy $x \in St(P)$ is a PNE for game $\mathcal{G}$ if and only if it is a solution of $CSP(\mathcal{G})$.*

The structure of a CSP instance $I = (Var, U, C)$ is represented by a hypergraph $H(I) = (V, HE)$, where $V = Var$, $HE = \{var(S)|(S, r) \in C\}$, and $var(S)$ is the set of variables in scope S of constraint (S, r). It follows from the construction of $CSP(\mathcal{G})$ that the hypergraph of $\mathcal{G}$ is exactly the same as the hypergraph of $CSP(\mathcal{G})$. Furthermore, let $hwd(H(\mathcal{G}))$ and $twd(G(\mathcal{G}))$ denote the hypertree width and the treewidth of game $\mathcal{G}$, respectively. A basic relationship is revealed between the two notions. Essentially, it is shown that we can map a game of bounded treewidth to one of bounded hypertreewidth [20].

Theorem 9. *For each game $\mathcal{G}$, $hwd(H(\mathcal{G})) \leq twd(G(\mathcal{G})) + 1$.*

Therefore, a game $\mathcal{G}$ of bounded treewidth also has bounded hypertree width. In turn this is mapped to a CSP whose corresponding hypergraph is of bounded hypertree width. The latter can be solved in polynomial time using results by Gottlob *et al.* [22].

Theorem 10. *Given a CSP instance I and a hypertree decomposition of $\mathcal{H}_I$ of width w, I is solvable in $O(||I||^{w+1} \log ||I||)$ time.*

The theorem above indicates that a CSP of bounded hypertree width can be solved in polynomial time; to achieve this a hypertree decomposition is exploitted, in order to obtain an equivalent acyclic CSP. This concludes the series of theorems used in the approach of [20] to show that pure Nash equilibria of graphical games can be decided in polynomial time. To recapitulate, a graphical game $\mathcal{G}$ of bounded treewidth w is known to have bounded hypertree width also w, which is then mapped to a CSP whose hypergraph is the same as $\mathcal{G}$ and thus of bounded hypertree width and can be solved in polynomial time. The solution of the CSP has a one-to-one correspondence with the PNE of $\mathcal{G}$.

On the other hand, the time complexity of the result, Theorem 10, does not imply a fixed-parameter algorithm with respect to the treewidth, since the base of the exponent depends on the size of the instant. In addition, Marx shows that this algorithm is essentially optimal [32]. The following result assumes the Exponential Time Hypothesis (ETH), which asserts that

there is no $2^{o(n)}$-time algorithm for 3SAT with n variables. Observe that this assumption is stronger than $FPT \neq W[1]$ [32].

Theorem 11 (Short version [32]). *If CSP can be solved in time* $f(G) \cdot ||I||^{o(w/\log w)}$ *then ETH fails.*

The theorem above shows that the $n^{O(w)}$-time algorithm is essentially optimal for every class of graphs, up to an $O(\log w)$ factor in the exponent. Thus, there is no other structural information besides treewidth that can be exploited algorithmically. In our context, it means that it is not possible to improve the time complexity bounds of the problem using the method of [20].

6.2.3 PNE via Markov Random Fields

A different approach was introduced in Daskalakis & Papadimitriou [12]. In their paper a new class of algorithms for finding pure Nash equilibria was presented, by mapping graphical games to Markov Random Fields (MRF), such that finding a maximum $a-posteriori$ configuration of the MRF decides also the existence of PNE. The resulting MRF is over an undirected graph G and is a probability distribution that factorizes according to functions defined on a set of cliques of G. MRFs are not going to be discussed in detail here since they are not of interest for this thesis. They are well-known models in the statistics literature. An extensive analysis can be found, for example, in [31].

The approach of this section, uses the *primal graph* of the game hypergraph as the graph of the MRF. The primal graph $G' = (V', E')$ of a hypergraph $H = (V, E)$ has $V' = V$ and two nodes $v_1, v_2 \in V'$ are connected if and only if there is a hyperedge $h \in E$ such that $v_1, v_2 \in h$. It is proved in [12] that if a graphical game $\mathcal{G}$ has treewidth bounded by w then its primal graph has treewidth bounded by $(w+1) \cdot \max_{p \in P} |\mathcal{N}(p)| - 1$. Moreover, given a tree decomposition of G, one for G' can be constructed in polynomial time. Subsequently, the *junction tree* algorithm, one of the most celebrated algorithms for statistical inference, is run to compute a maximum a posteriori configuration. For a detailed description of this algorithm the

interested reader is referred to [24]. Analytically, Algorithm 1 is separated in four steps:

Algorithm 1 [12]

Input: $\mathcal{G} = (G, \mathcal{M})$

1: First check if the input graph G has treewidth bounded by w. If so, find a tree decomposition, say $\mathcal{T}$, of width at most w using the algorithm of [4].
2: From $\mathcal{T}$ we get $\mathcal{T}'$, a tree decomposition of the primal graph of the game having width at most $(w+1) \cdot \max_{p \in P} |\mathcal{N}(p)| - 1$
3: Reduce the input instance to the corresponding MRF. The primal graph of the game is the graph of the MRF.
4: Run the junction tree algorithm on $\mathcal{T}'$ to compute the marginal probabilities distributions of the bags of $\mathcal{T}'$. These answer the question of whether $\mathcal{G}$ has a pure Nash equilibrium.

The correctness of the reduction and the above algorithm is out of the scope of this thesis and thus we will focus on its time complexity, which is of direct interest. In the MRF created at step 3 of Algorithm 1, let X_v be the finite set of values available for the random variable associated with node v. The junction tree algorithm runs in time exponential to the width of $\mathcal{T}'$: If $X_v = \chi$, $\forall v \in V$ then the running time is $O(n \cdot \chi^{width(T)})$. Since the aforementioned reduction sets $X_p = S_p$ where S_p is the set of strategies of player p, we conclude that the running time of the algorithm is $O(n \cdot \alpha^{width(\mathcal{T}')})$.

Theorem 12. *Deciding whether a graphical game has a pure Nash equilibrium is in P for all classes of games with bounded treewidth.*

Proof. The proof consists of a time analysis of Algorithm 1. The algorithm at step 2, transforms a tree decomposition of the input graph of width w to one of the primal graph of the game with width $(w+1) \cdot \Delta$, where Δ is the maximum degree over the vertices of the input graph. Then the running time is of Algorithm 1 is $O(n \cdot \alpha^{\Delta \cdot (w+1)})$ which can re-written as

$$O(n \cdot |M_p|^{w+1})$$

where p is such that $degree(p) = \Delta$, since $|M_p| = \alpha^{\Delta+1}$. □

The running time of this section's algorithm is polynomial for bounded treewidth. However, it is not fixed-parameter tractable and thus not linear for bounded treewidth. Further, in [12] was introduced a method to derive polynomial time algorithms for pure Nash equilibria of games with $O(\log n)$ treewidth. The methodology is described in the proof of the following theorem and will be used in our algorithms presented in the next chapter to attain similar, improved results.

Theorem 13. *Deciding whether a graphical game has a pure Nash equilibrium is in P for all classes of games with $O(\log n)$ treewidth, bounded number of strategies, and bounded neighborhood size.*

Proof. Given a graph of treewidth $c \cdot \log n$ the first step is to obtain a tree decomposition of width $3.67 \cdot c \cdot \log n$ by using a slightly modified version of the algorithm presented in [1]. Similarly, this tree decomposition is transformed to one of the primal graph of the input game, with size $(3.67 \cdot c \cdot \log n + 1) \cdot \Delta$, on which the junction tree algorithm is run. With similar rationale as the proof of Theorem 12 we obtain the following time bound:

$$\begin{aligned} O(n \cdot \alpha^{(3.67 \cdot c \cdot \log n + 1) \cdot \Delta}) &= O(n \cdot \alpha^{3.67 \cdot c \cdot \log n \cdot \Delta + \Delta}) \\ &= O(n \cdot \alpha^{\log n \cdot 3.67 \cdot c \cdot \Delta} \cdot \alpha^{\Delta}) \\ &= O(n \cdot n^{3.67 \cdot c \cdot \Delta} \cdot \alpha^{\Delta}) \end{aligned}$$

which is can be written as $O(n^{\Delta})$. That is, assuming α, Δ are constants and the base of the logarithm is α. □

The algorithm suggested by Theorem 13 is polynomial to n. Note that the assumption that the degree of the graph is bounded is necessary for the description of the input game to be polynomial in the number of players. Finally, let us note that the result of Theorem 13 can be combined with a reduction of the problem of finding approximate mixed Nash equilbria to the problem of finding pure Nash equilibria. This reduction was implicit in the approximation algorithm part of [28]. The combination leads to the following result (PTAS).

Theorem 14. *An ϵ-approximate mixed Nash equilibrium of any graphical game with $O(\log n)$ treewidth, bounded neighborhood size and bounded number of strategies per player can be found in time polynomial in n and $\frac{1}{\epsilon}$.*

6.2.4 Another characterization of Hard and Easy games

A quite different approach is provided by Jiang and Safari [27], who prove that deciding the existence of a PNE is tractable for all classes of graphs that are recursively enumerable and have bounded treewidth. Their result shows that there is no class of problems in that domain that is in FPT but not in P. For the FPT part, the parameter is the representational size of the graph. In the case of undirected graphs the representation size is $p = |G| = |V| + |E|$. The next theorem summarizes the results.

Theorem 15 ([27])**.** *Assume $FPT \neq W[1]$. Then for every recursively enumerable class C of graphs with bounded treewidth the following statements are equivalent:*

1. *PNE-GG is in polynomial time for any graph in C.*
2. *p-PNE-GG is fixed parameter tractable for any graph in C.*

This theorem for undirected graphs follows as a corollary from similar results for directed graphs. In other words, in this domain there is no class of problems that is FPT but not in P. As a final note, we mention a strange relation: Theorem 15 is based on the homeomorphism part of Grohe's celebrated theorem [23], while Theorem 11 is based on the Constraint Satisfaction part of the same theorem. This interrelation of PNE and CSPs can also be witnessed in the results of Gottlob *et al.* [20], presented in Section 6.2.2.

6.2.5 W[1]-Hardness

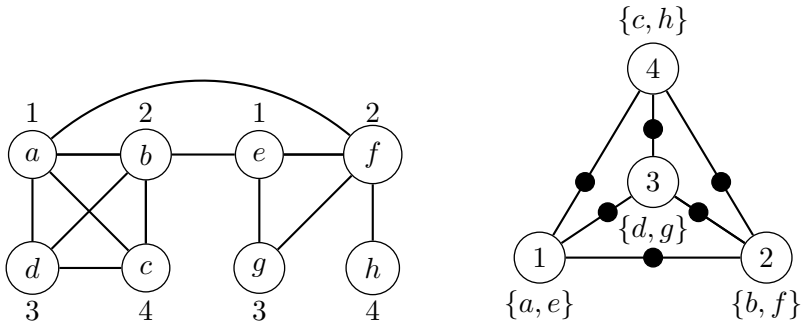
Since treewidth plays an important role in the computation of pure Nash equilibria for graphical games, we will consider the problem from the viewpoint of parameterized complexity. To the best of our knowledge, none of the previous results implies the existence of a fixed-parameter tractable algorithm with respect to the treewidth of the input graph. Here we argue that this is not surprising; consider the parameterized problem:

w-PNE-GG
Input: $\mathcal{G} = (G, \mathcal{M})$, $\mathcal{T}$ tree decomposition of G.
Parameter: w - the width of $\mathcal{T}$.
Question: Is there a PNE in $\mathcal{G}$?

In this section we will prove that deciding a pure Nash equilibrium is $W[1]$-Hard with regard to the parameter treewidth. For this, a reduction from the $W[1]$-Hard problem k-Multicolor Clique will be used. The hardness of the problem, which follows easily by reduction from k-Clique together with a general reduction technique was originally described in [18]. The statement of the problem is as follows:

k-Multicolor Clique
Input: A graph $G = (V, E)$ and a vertex coloring $c : V \to \{1, \ldots, k\}$.
Parameter: k - the number of colors.
Question: Does G contain a clique containing vertices of all k colors?

Before preceding to the reduction, let us introduce some useful notation. Let G be the input graph given along with k-coloring $c : V \to \{1, \ldots, k\}$. We let V_c denote the vertices colored c, i.e. $V_c = \{v \in V | c(v) = c\}$, and E_{c_i,c_j} be the set of edges such that if $(u, v) \in E_{c_i,c_j}$, then $\{c(u), c(v)\} = \{c_i, c_j\}$. In addition, observe that it can be assumed w.l.o.g. that the input coloring is *proper*, i.e. for any color c, $E_{c,c} = \emptyset$. Any such edge can be removed from G [18]. We can also assume that the color classes of G, and the edge sets between them, have uniform sizes, i.e $|V_c| = N$ for all c and $|E_{c_i,c_j}| = M$ for all $c_i < c_j$. A simple justification of this assumption is given in [17], by reducing the Multicolor Clique problem to itself: Let $\mathcal{S}_k$ be the set of permutations of $\{1, \ldots, k\}$. Given a k-colored graph and a permutation $\sigma \in \mathcal{S}_k$, with G_σ we denote the graph where the color class c is colored with $\sigma(c)$. Given an instance of Multicolor Clique, we take the union of $k!$ disjoint copies of G, one of each permutation of the set of colors, i.e. $G' = \bigcup_{\sigma \in \mathcal{S}_k} G_\sigma$. It is evident that G has a multicolor clique if and only if G' has one, while G' has uniform sizes both for the color classes and the edge sets between different color classes.



(a) Multicolor Clique (b) PNE-GG

Figure 5: An example of the reduction where the numbers correspond to different colors. In (b) the strategy sets are shown in curly brackets (omitting NA) and the auxiliary players are represented as black vertices.

Theorem 16. *w-PNE-GG is W*[1]*-Hard.*

Proof. Given an instance of Multicolor Clique, graph G with k-coloring c, we construct an instance $\mathcal{G} = (G' = (P, E'), \mathcal{M})$ of PNE-GG as follows: The players of $\mathcal{G}$ are separated in two distinct sets, the *colorful* P_c and the *auxiliary* P_a players, $P = P_c \cup P_a$. Every $c \in P_c$ is connected to all the other colorful players $c' \in P_c$, through an auxiliary vertex $a \in P_a$. In other words, G' is the graph that arises by taking a k-clique and adding one auxiliary player on each edge. It is obvious that the treewidth of G' is exactly k and thus the parameter is preserved.

The strategy sets are defined in the following manner: For a player $c \in P_c$, the possible actions are all the vertices of G that are colored c plus an extra action NA, that stands for non-adjacent. Formally, $St(c) = \{v \in V | c(v) = c\} \cup \{NA\}$. An auxiliary player $a \in P_a$ has only two possible actions, $St(p) = \{A, NA\}$, that stand for adjacent and non-adjacent respectively. Observe that G' is built such that all colorful vertices neighbor only with auxiliary vertices and each auxiliary vertex is neighbor to exactly 2 colorful ones. An example reduction can be found in Figure 5.

Let x be a global configuration. For an auxiliary player $a \in P_a$ let i, j be the two neighboring colorful players, i.e. $i, j \in \mathcal{N}(a)$. Then, the utility function u_a is such that:

(i) $u_a(x) = 1$ if a plays A and i, j play actions v, u such that $(v, u) \in E$ or at least one of i, j plays NA;

(ii) $u_a(x) = 1$ if a plays NA and i, j play actions v, u such that $(v, u) \notin E$ and neither of i, j plays NA;

(iii) $u_a(x) = 0$ in all other cases.

For each player $c \in P_c$, her utility function u_c is such that

(iv) $u_c(x) = 1$ if c plays an action in $St(c)\backslash\{NA\}$, and all of her neighbors play action A;

(v) $u_c(x) = 1$ if c plays NA and at least one of her neighbors plays NA;

(vi) $u_c(x) = 0$ in all other cases.

Intuitively, game $\mathcal{G}$ is built such that in a pure Nash equilibrium every colorful vertex $c \in P_c$ plays an action that corresponds to a vertex $v \in V$ with $c(v) = c$ and every auxiliary player chooses A, indicating that her neighbors are adjacent in G. If the neighbors of an auxiliary vertex a play actions that correspond to non-adjacent vertices, then a will play NA and her neighbors will have the incentive to play NA. Then a would have the incentive to play A and thus such a configuration cannot be a pure Nash equilibrium.

Claim 3. *G has a clique including all k colors if and only if $\mathcal{G}$ has a pure Nash equilibrium.*

Proof. Let $(v_1, \ldots, v_k)$ be a k-clique of G that contains all k colors. Consider the global strategy x where each player $c \in P_c$ plays the action that corresponds to vertex v_c (the vertex from the clique that is colored c) and each auxiliary vertex plays A. Observe that in this case all players receive payoff 1 which is the maximum they can receive and thus x is a pure Nash equilibrium.

To prove the opposite direction of the claim we will first argue that there is no pure Nash equilibrium of $\mathcal{G}$ where there is an auxiliary vertex that plays NA. Assume that x is a PNE and $\exists a \in P_a$, with neighbor $j \in \mathcal{N}(a)$, that plays NA. Then, j would have an incentive to play NA and get payoff 1 rather than an action in $St(j)\backslash\{NA\}$. Consequently, a would prefer A over NA which contradicts our assumption that x is a PNE.

Now, let x be a global configuration and a pure Nash equilibrium of $\mathcal{G}$. From the previous paragraph, every $a \in P_a$ plays A and thus every

$c \in P_c$ plays an action in $St(j)\backslash\{NA\}$. Consider the set of vertices $K = (v_1, \ldots, v_k)$ where each v_c corresponds to the strategy of player $c \in P_c$. Since each auxiliary vertex plays A, it means that all vertices in K are pairwise connected to each other and therefore form a clique. In addition, they all belong to a different color class because of the construction of $\mathcal{G}$. Therefore, K is a multicolored k-clique of G. □

To conclude our proof we need to show that the reduction takes at most time of the form $f(k) \cdot p(|G|, k)$ for some computable function f and polynomial $p(X)$. The time of the construction is dominated by the computation of the matrix collection $\mathcal{M}$, whose size is the summation of the sizes of the individual matrices

$$|\mathcal{M}| = \sum_{c \in P_c} |M_c| + \sum_{a \in P_a} |M_a|$$

As mentioned earlier, we assume that the color classes of the MULTICOLOR CLIQUE instance have uniform size N and thus $N = \frac{n}{k}$ and for $c \in P_c$, $|St(c)| = N + 1$. In addition, observe that $|P_c| = k$ and that $|P_a| = \frac{k(k-1)}{2}$ since we have one auxiliary vertex for each edge of the k-clique. Then the above summation can be rewritten as

$$\begin{aligned} & k \cdot ((N+1) \cdot 2^{k-1}) + \frac{k(k-1)}{2} 2 \cdot (N+1)^2 \leq \\ & 2^{k-1} \cdot (n+k) + k^2 \cdot (N+1)^2 \leq \\ & 2^k \cdot n + 4n^2 \end{aligned}$$

Therefore, the time we need for the whole reduction is at most $f(k) \cdot p(|G|)$ which concludes our proof.

□

We conclude that w-PNE-GG does not admit a Fixed-Parameter-Tractable algorithm, unless $FPT = W[1]$. Nevertheless, in the next chapter we will demonstrate an algorithm that becomes FPT for games with a bounded number of available strategies per player.

7 A Fixed-Parameter-Tractable algorithm

Following, we will present in detail a new approach for computing pure Nash equilibria of graphical games. First, we describe a linear-time algorithm for

instances where the input graph is a tree. Then we generalize this algorithm to a tree decomposition. For the sake of simplicity, our approach is analyzed in terms of a nice tree decomposition. We argue that our algorithm runs in time $O(\alpha^w \cdot w \cdot |\mathcal{M}|)$ which is fixed-parameter tractable for graphical game instances with bounded cardinality strategy sets. Subsequently, we argue that w-PNE-GG does not admit a polynomial kernel, even when the number of available strategies is bounded. In, addition we exploit techniques from [12] to show that our algorithm is polynomial for graphs of $O(\log n)$ treewidth. We improve on the known result by dropping the bounded neighborhood assumption. Finally, we argue that the output of our algorithm constitutes a succinct description of the set of all PNE and we describe how to construct a sample, or maximum-payoff, PNE using a linear-time algorithm.

7.1 Tree Algorithm

In this section we present a simple algorithm that answers the question of existence of a PNE of a graphical game, when the input graph is a tree. The idea is that every vertex is able to compute the best response(s) for each configuration of its children, while ignoring its parent. Then, visiting the vertices in a bottom-up manner the parent will be taken into account in the subsequent step. Given a graphical game $(T, \mathcal{M})$, where $T = (V, E)$ is a tree, we proceed as follows. First, we take an arbitrary vertex and consider it as the root r of the tree. For each $v \in V$, by T_v we denote the subtree with v as root and by p_v the parent of v. Moreover, for each vertex $v \in V$ and action $i \in St(v)$: $A_i(v)$ denotes what is the maximum number of players in PNE in T_v when v plays action i; $B_i(v)$ denotes the strategy p_v has to play to achieve PNE state for $A_i(v)+1$ nodes in the tree $T_v \cup \{p_v\}$. Namely, $B_i(v)$ contains the set of strategies that p_v has to play such that for $C \in St(\mathcal{N}(v))$ with $C_{p_v} \in B_i(v)$ and $A_i(C) = |T_v|$, $i \in \beta_v(C)$. Consequently, a dynamic programming approach is used to fill in the tables, which count to $\alpha \cdot |St(v)|$ for each player $v \in V$. An A_i table has only one value and a B_i table for player $v \in V$ has at most $|St(p_v)|$ values.

For each vertex/player $v \in V$ we perform a single column-by-column scan of M_v to indicate $\beta_v(C) \subseteq St(v)$, the set of best response strategies for

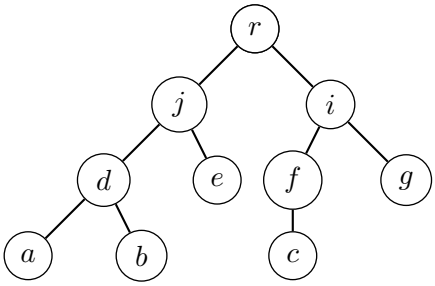


Figure 6: An illustration of a tree graph is given, in order to help the reader visualize the algorithm.

each configuration $C \in St(\mathcal{N}(v))$. Now, we can compute the tables A, B in a bottom-up order, starting with the leaves of G. Initially tables $A_i(v) = 0$ and $B_i(v) = \emptyset$, $\forall v \in V$. We distinguish three different cases:

v is a leaf: For each action $i \in St(v)$, if $\exists C \in St(\mathcal{N}(v))$ such that $i \in \beta_v(C)$ we set $A_i(v) = 1$ and $B_i(v) = C_{p_v}$ Otherwise the tables preserve the default values (as above). The corresponding B tables will contain the action(s) of p_v which result in both v, p_v being in PNE from v's perspective.

v has children $x_1, x_2, \ldots, x_m$:

1. Iterate through the matrix M_v row by row: $\forall i \in St(v)$ and for every configuration $C \in St(\mathcal{N}(v))$ such that $i \in \beta_v(C)$ compute $\sum_{u \in \mathcal{N}(v)} A_{C_u}(u)$.
2. For every $C \in St(\mathcal{N}(v))$ such that

$$\sum_{u \in \mathcal{N}(v)} A_{C_u}(u) = |T_v| - 1 \tag{1}$$

   if also

$$i \in B_{C_u}(u),\ \forall u \in \mathcal{N}(v) \backslash \{p_v\} \tag{2}$$

   then set $A_i(v) = |T_v|$ and otherwise $A_i(v) = |T_v| - 1$.
3. For each configuration $C \in St(\mathcal{N}(v))$ satisfying conditions (4.1), (4.2) put strategy C_{p_v} in $B_i(v)$.

v is the root: Perform steps $1, 2$ as above and set $B_i(v) = \emptyset$.

Then, game $(T, \mathcal{M})$ has a PNE if and only if , for root r, $\exists i \in St(r)$ such that $A_i(r) = |V|$. Note that at any point if we find a player $v \in V$ such that $\forall i \in St(v)$, $A_i(v) < |T_v|$ we can stop the execution of the algorithm and return NO. In the proposed procedure for each player $v \in V$ we perform two iterations through the matrix M_v, one at the initialization step to compute β_v and one at step 1 of the procedure above. The values of tables A and B are computed once and are used once by the parent of that vertex at the computations of step (4.1),(4.2) above. Observe that evaluating condition (4.1) needs only to be performed once for each configuration found in matrix M_v. To evaluate condition (4.2) we are only interested in the existence of a single action in the B tables of all children of v. Since configurations are order lexicographically in each matrix M_v, each of the tables A_i, B_i of the descendants of v needs to be looked up once. Thus, the total time needed to compute the answer is bounded by $O(|\mathcal{M}|)$.

Theorem 17. *Given a graphical game $(T, \mathcal{M})$, where T is a tree, we can compute a PNE in time $O(|\mathcal{M}|)$.*

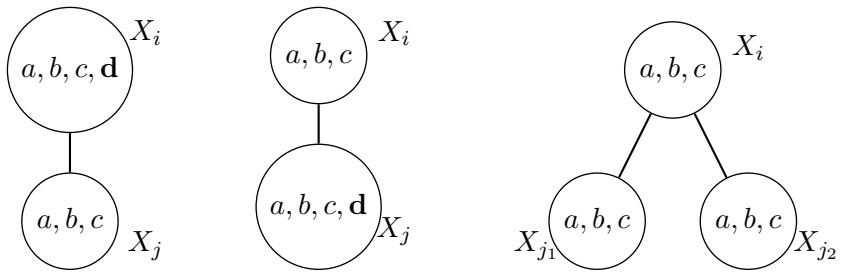
7.2 From Tree to Treewidth

In this section we generalize the tree algorithm to tree decompositions. For this, we will follow an approach native in the field of parameterized complexity and thoroughly described in [5]. The treewidth of the input graph will be treated as the parameter of the problem; thus the problem under consideration is w-PNE-GG as defined in the previous section. By Theorem 7, the unparameterized version of this problem is NP-Complete.

Our goal is to obtain a novel algorithm with improved time complexity upper bounds by attacking the combinatorics of the problem directly. The general idea is to go through all possible configurations for each bag of the tree, which count to α^w. Then, put together this information on the tree decomposition in polynomial time. The analysis we provide, for the sake of simplicity, is based on a *nice tree decomposition*. In such a decomposition, one node in $\mathcal{T}$ is considered to be the root and each node $i \in I$ is one of the following four types:

- Leaf: node i is a leaf of $\mathcal{T}$ and $|X_i| = 1$;
- Join: node i has exactly two children, say j_1, j_2 and $X_i = X_{j_1} = X_{j_2}$;
- Introduce: node i has exactly one child, say j, and $\exists v \in V$ with $X_i = X_j \cup \{v\}$;
- Forget: node i has exactly one child, say j, and $\exists v \in V$ with $X_j = X_i \cup \{v\}$.

It is known that if a graph G has a tree decomposition with width at most w, then it also has a nice tree decomposition of width at most w and $O(n)$ tree nodes. Such a decomposition can be built in linear time, given a not-nice one [30]. The approach presented below, consists of a dynamic programming algorithm which is executed on the tree decomposition. A table is computed for each node of the tree decomposition; for the decision problem the answer lies in the table of the root of the tree.



(a) Introduce (b) Forget (c) Join

Figure 7: An illustration of the different node types of a nice tree decomposition.

7.3 A dynamic programming approach

Suppose we are given an instance of the problem described above; a graph $G = (V, E)$, a collection of matrices $\mathcal{M}$ -one matrix M_p for each node $p \in V$- and a tree decomposition $\mathcal{T}$. We assume that the tree decomposition $(\{X_i | i \in I\}, \mathcal{T} = (I, F))$ is nice. Each node $i \in I$ is associated to a graph $G_i = (V_i, E_i)$. V_i is the union of all bags X_j, with $j = i$ or a descendant of i

in $\mathcal{T}$, and $E_i = E \cap (V_i \times V_i)$. In other words, G_i is the subgraph of G induced by V_i. A table A_i is to be computed for each node $i \in I$ and contains an integer value for each possible configuration $C \in St(X_i)$. Therefore, when the treewidth is w, table A_i contains at most $\alpha^{|X_i|} \leq \alpha^w$ values. Given configuration $C \in St(X_i)$, the value of the table $A_i(C)$ corresponds to the maximum number of players in PNE in G_i, such that every player $p \in X_i$ plays according to C. Note that the strategy for the players in $V_i - X_i$ is not explicitly mentioned at this point (where the algorithm is treating bag X_i) but has been treated at an earlier time of the execution of the algorithm. Table A_i is computed for all nodes $i \in I$ in bottom-up order; for each non-leaf node we use the tables of its children to compute table A_i. We perform a case analysis based on the type of the node under examination.

Leaf nodes.

Suppose node i is a leaf of $\mathcal{T}$ with $X_i = \{p\}$. Then, table A_i has only $|St(p)|$ entries. The value 1 will be attributed to these entries since we assume that a single player can be in PNE, no matter what strategy it follows, when there is no other player to compete with. Hence, for each configuration C over the vertices of X_i (in this case $St(X_i) = St(p)$) we set $A_i(C) = 1$.

Introduce nodes.

Suppose i is an introduce node of $\mathcal{T}$ with child j and that $X_i = X_j \cup \{p\}$. It is known that there is no vertex $u \in V_j - X_j$ such that $\{p, u\} \in E$ [5]. Hence, G_i is formed from G_j by adding p and zero or more edges from p to vertices in X_j.

Lemma 2. *Let $C \in St(X_j)$. If $\forall u \in X_j$, $\{p, u\} \notin E$, then $A_i(C \cup \{a_p\}) = A_j(C) + 1$ for all actions $a_p \in St(p)$.*

In the case above, p is not connected to any vertex in G_i and therefore we consider it as in PNE. For the other case we have to be more elaborate. Assume that there is $u \in X_j$ such that $\{p, u\} \in E$. We use Algorithm 2 which proceeds in the following manner: Given a strategy a_p of player $p \in X_i - X_j$, for each player $u \in X_j$ that is adjacent to p it checks if u is in PNE with

respect to configuration $C \cup \{a_p\}$. In the positive case it adds player u to the set P_i. In the end of the iteration if all players in $\mathcal{N}(p) \cap X_j$ are also in P_i it means that all players in G_i connected to p are in PNE with respect to the current configuration. Hence, for $A_i(C \cup \{a_p\})$ we take the value $A_j(C) + 1$.

Algorithm 2 IntroNode

Input: $C \in St(X_j)$, $p \in X_i - X_j$, $a_p \in \beta_p(C)$
Output: $A_i(C \cup \{a_v\})$

1: Initiate set $P_i = \emptyset$
2: **for** $u \in \mathcal{N}(p) \cap X_j$ **do**
3: **if** $C_u \in \beta_u(C \cup \{a_p\})$ **then**
4: $P_i \leftarrow P_i \cup \{u\}$
5: **end if**
6: **end for**
7: **if** $P_i = \mathcal{N}(p) \cap X_j$ **then**
8: $A_i(C \cup \{a_p\}) \leftarrow A_j(C) + 1$
9: **else**
10: $A_i(C \cup \{a_p\}) \leftarrow A_j(C)$
11: **end if**

Lemma 3. *Given an introduce node $i \in \mathcal{T}$ with child j such that $p \in X_i - X_j$, we can compute $A_i(C)$ for all configurations C in at most*

$$\alpha^{|X_i|} \cdot (|M_v| + \sum_{u \in \mathcal{N}(p) \cap X_j} |M_u|)$$

computational steps.

Proof. Before we start the procedure we compute the set $\mathcal{N}(p) \cap X_j$ in at most $|X_j|$ steps. This happens only once for each introduce node. In the case $\nexists u \in X_j$ such that $\{p, u\} \in E$ we compute the table value for each configuration in constant time and thus the total time needed is $O(\alpha^{|X_i|})$.

In the other case, we use Algorithm 2 for each configuration $C \in St(X_j)$. Computing $\beta_p(C)$ takes at most $\frac{|M_p|}{\min_{u \in \mathcal{N}(p) \cap X_j} |St(u)|}$ steps. The loop at lines 2-6 is through all vertices $u \in \mathcal{N}(p) \cap X_j$ and for each vertex u, $\beta_u(C \cup \{a_p\})$ is computed once at line 3 in at most $\frac{|M_u|}{|St(p)|}$ steps. The operation at line 7 takes one step. For each of the $\alpha^{|X_j|} = \alpha^{|X_i|-1}$, configurations Algorithm 2

has to be run at most α times (for each $a_p \in St(p)$). Note that for the computation of the best response function, we ignore the division since it does not improve our upper bounds. The lemma follows. □

The simple algorithm and the lemma above, show us how to compute the table for an introduce node using information found in the table of the child node. For the introduced vertex p, we have that the matrix M_p and at most other $w-1$ matrices are read once for each configuration. Note that adjacency is only checked once since it does not change for different entries.

Forget nodes.

Suppose i is a forget node of $\mathcal{T}$ with child j. In this case, G_i and G_j is the same graph but X_i and X_j differ by one vertex. Suppose this vertex is $p \in X_j - X_i$. To compute the tables of a forget node we use the procedure suggested by Lemma 4. For each of the $\alpha^{|X_i|}$ possible configurations we perform a number of α steps for a total of $O(\alpha^{|X_i|+1})$.

Lemma 4. *Let $C \in St(X_i)$, $A_i(C) = \max_{a_p \in St(p)} A_j(C \cup \{a_p\})$.*

Proof. Simply observe that in the case of a forget node the number of players in PNE in graph G_i (value A_i) is the same as the number of players in PNE in G_j. Thus, we only have to find which of the available strategies of p gives the maximum for A_j. □

Since X_i is a forget node we have $|X_i| < w$ and thus the time needed for this operation is $O(\alpha^w)$.

Join nodes.

Suppose i is a join node of $\mathcal{T}$ with children j_1 and j_2. Remember that $X_i = X_{j_1} = X_{j_2}$. Then, G_i can be interpreted as a union of G_{j_1} and G_{j_2}. The following lemma provides useful insight on this kind of nodes and will be significant in computing the table of a join node.

Lemma 5 ([5])**.** *Let i, j_1, j_2 as above. If $v \in V_{j_1}, w \in V_{j_2}$ and $v, w \notin X_i$, then $\{v, w\} \notin E$.*

What we need to capture here, is that a configuration C for the players of X_i may be part of a PNE for G_i if and only if it also is for both G_{j_1} and G_{j_2}. Given a configuration C the computation of $A_i(C)$ for a join node takes only constant time as described by Lemma 6. Therefore, the computation of the whole table for a join node takes place in time $O(\alpha^w)$.

Lemma 6. *Let $C \in St(X_i)$, $A_i(C) = A_{j_1}(C) + A_{j_2}(C) - |X_i|$.*

Proof. By adding the values of the two subtrees we add the vertices found in X_i two times and thus we have to subtract $|X_i|$ from the total sum. Because of Lemma 5, we know that vertices in X_i do not connect to vertices not in X_i. Assuming every vertex not in X_i is in PNE, all the vertices in X_i have to be in PNE in both subtrees in order to have PNE for the whole graph G_i. □

7.4 Putting the tables together

The algorithm proposed in this section is a simple bottom-up tree walk that finds partial configurations for each bag of the tree decomposition, that could be part of a global PNE configuration. Then, these configurations are synthesized together on every step of the tree walk and an answer can be achieved when the root of the tree decomposition is reached.

Lemma 7. *Graphical game $(G, \mathcal{M})$ has a PNE if and only if $\exists C \in St(X_r)$ such that $A_r(C) = |V|$.*

Proof. Observe that G_r is the input graph. Then $A_r(C) = |V|$ can be interpreted as all players in G are in PNE state under (global) configuration C. □

In addition, note that while computing the tables for each bag of the tree decomposition it is possible to stop before reaching the root when there is no PNE. During the bottom-up tree walk if there exists a bag X_i such that $\forall C \in St(X_i)$, $A_i(C) < |V_i|$ then we can stop the execution of the algorithm and reply NO. The tables of all bags of the tree decomposition have to be computed to verify a YES instance since any partial PNE configuration might be jeopardized by a newly introduced vertex.

The theorem below provides upper bounds to the computational complexity of the algorithm suggested in this section. Intuitively, for each player $p \in V$ that is introduced in $\mathcal{T}$ the matrix M_p is read and at most $w-1$ matrices that belong to its neighbors. Thus, for each node we iterate through at most w local matrices and since the nodes of $\mathcal{T}$ is $O(n)$ the summation over the local matrices can be bounded by the size of the matrix collection $\mathcal{M}$ given in the input description.

Theorem 18. *Given a graphical game $(G, \mathcal{M})$ and a tree decomposition $\mathcal{T}$ of width w, there is an algorithm that determines the existence of a PNE in $O(\alpha^w \cdot w \cdot |\mathcal{M}|)$ time.*

Proof. The time spent to compute the table of each non introduce node X_j is bounded by $O(\alpha^{|X_j|})$ as discussed in the previous section. Introduce nodes are computationally more expensive and therefore define the worst case complexity of the problem. Assuming every node $i \in \mathcal{T}$ is an introduce node[1] with child j and vertex $p \in X_i - X_j$ we derive the following upper bound:

$$\alpha^w \cdot \sum_{i \in \mathcal{T}} \left(|M_p| + \sum_{u \in \mathcal{N}(p) \cap X_j} |M_u| \right) \leq \alpha^w \cdot (|\mathcal{M}| + (w-1)|\mathcal{M}|) \qquad (3)$$

Since $\mathcal{T}$ is a nice tree decomposition it consists of $O(n)$ vertices and therefore the first summation is over $O(n)$ elements. The summation over all matrices $|M_v|$ gives $|\mathcal{M}|$. In addition, the second summation is over at most $w-1$ elements which in turn are upper bounded by $|\mathcal{M}|$ because of the first summation. The theorem follows. □

Furthermore, it is known from Theorem 9 that for a game $\mathcal{G}$, $hwd \leq w+1$, where hwd and w is the hypertree width and the treewidth of $\mathcal{G}$, respectively. This fundamental relationship between the two width notions was shown in [20] and exploitted in order to show tractability of games with bounded treewidth. Combining this result with Theorem 18, we obtain as a corollary the fixed-parameter tractability of computing a PNE for graphical games with bounded cardinality strategy sets when the parameter is hypertreewidth.

[1] Observe that in the case of a clique all the nodes of $\mathcal{T}$ but one are introduce nodes.

Corollary 1. *Given a graphical game $(G, \mathcal{M})$ and a tree decomposition $\mathcal{T}$ of width hwd, there is an algorithm that computes a succinct desciption of all PNE in $O(2^{hwd} \cdot hwd \cdot |\mathcal{M}|)$ time.*

Our algorithm improves significantly on the previous known bounds, since the base of the exponent is only the number of possible actions and not the whole description of the game. In addition, if we assume that the number of available strategies is bounded by a constant, our algorithm becomes fixed-parameter tractable. To the best of our knowledge, this is the first FPT algorithm for computing pure Nash equilibria. Summing up, for computing PNE of graphical games the time bound of Theorem 18 implies:

- A fixed-parameter algorithm when strategy sets are of bounded cardinality and the parameter is either treewidth or hypertreewidth.
- A polynomial algorithm with improved bounds for graphical games of bounded treewidth.
- A linear algorithm for games of bounded treewidth and bounded cardinality strategy sets.

Note that determining whether the treewidth of a given graph G is at most w, and if so, find a tree decomposition of width at most w is FPT when the parameter is the treewidth [5]. On the contrary, deciding whether the hypertreewidth of a given hypergraph H is at most hwd is $W[2]$-Hard and thus unlikely to be FPT [21].

7.5 Kernelization

Since the algorithm suggested in this section is fixed-parameter tractable for games with bounded cardinality strategy sets (α is a constant), the question of the existence of a kernelization algorithm arises. A polynomial kernel means that given an instance of w-PNE-GG, we would obtain in polynomial time an equivalent instance whose size is bounded polynomially to w. In this section we will argue that the existence of such a kernel is rather unlikely.

The proof of NP-Completeness for the unparameterized version of the problem, described by Theorem 7, consists of a reduction from 3-CNF-SAT

[20]. The resulting game is a played on the incidence graph of the SAT instance. Then, the treewidth of the resulting game is exactly the incidence treewidth of the SAT instance and this is a polynomial time and parameter transformation [6] from w-3-CNF-SAT, where w is the treewidth of the incidence graph, to w-PNE-GG. In addition, each player has only 3 available strategies and thus α is bounded. Since, w-SAT is fixed parameter tractable [39] and can be shown trivially and-compositional, the theorem below follows from the method described in [6].

Lemma 8. *SAT is and-compositional with respect to the treewidth of the incidence graph.*

Proof. Given m instances $(x_1, w), (x_2, w), \ldots, (x_m, w)$ of SAT we construct instance X to be $\bigwedge_{1 \leq i \leq m} x_i$. W.l.o.g. we assume that there is no variable that occurs in two different input instances (even if there is one can trivially rename). The treewidth the incidence graph of X is w since there is no clause vertex with an edge to a variable that does not belong to the same original instance. X is a YES instance if and only if all instances $x_1, x_2, \ldots, x_m$ are YES instances. □

Theorem 19. *w-PNE-GG does not admit a polynomial kernel, unless the and-distillation conjecture does not hold.*

It remains open if we can come to the same result by using or-compositionality, which is essentially stronger. If the or-distillation conjecture does not hold then $coNP \subseteq NP/poly$ [19] but no relevant result is known for and-compositionality.

7.6 $O(\log n)$-Treewidth

In their paper, Daskalakis and Papadimitriou, proved that deciding the existence of a PNE is in P for all classes of games with $O(\log n)$ treewidth, bounded number of strategies and bounded neighborhood size [12]. In this section we follow their approach and argue that our algorithm implies slight improvements on their results. First, we obtain a polynomial algorithm for graphical games of $O(\log n)$ treewidth and bounded number of strategies, dropping the bounded neighborhood size assumption.

Theorem 20. *Given a graphical game with $O(\log n)$ treewidth and bounded number of strategies, there is an algorithm that decides the existence of a PNE in time polynomial in the description of the game.*

Proof. Suppose that the treewidth of the input graphical game is $w = O(\log n)$; we use a modified version of the algorithm presented by Becker and Geiger in [1] as discussed in [12]. This algorithm runs in time $poly(n) \cdot 2^{4.67 \cdot k}$, when the input graph consists of n vertices, and either outputs a legitimate tree decomposition $\mathcal{T}$ of width $3.67w$ or outputs that the treewidth is larger than w; for $w = c \log n$, where c is a constant, the algorithm either returns $\mathcal{T}$ of width $3.67c \log n$ or outputs that the treewidth of G is larger than $c \log n$. Assuming the positive case, we have a tree decomposition $\mathcal{T}$ of the input graph of size $3.67c \log n$. When $\mathcal{T}$ is fed to our algorithm presented in the previous sections it results in an upper bound of

$$\alpha^{3.67 \cdot c \cdot \log n} \cdot 3.67 \cdot c \cdot \log n \cdot |\mathcal{M}| =$$
$$n^{3.67 \cdot c} \cdot 3.67 \cdot c \cdot \log n \cdot |\mathcal{M}|$$

computational steps, which is $poly(|\mathcal{M}|)$. □

Note that if the degree of the graph is bounded, then the description of the graphical game is polynomial in the number of players. Therefore, if we apply the additional assumption of bounded neighborhood we achieve an upper bound that is polynomial in n. Our bound improves on the time complexity of the algorithm presented in [12] by removing $\Delta = \max_{p \in V} |\mathcal{N}(p)|$ from the exponent.

Corollary 2. *Given a graphical game with $O(\log n)$ treewidth, bounded number of strategies and bounded neighborhood size, there is an algorithm that decides the existence of a PNE in time polynomial in the number of players.*

Proof. We follow the same rationale as above. Since $|\mathcal{M}| \leq n \cdot |M_\Delta| = n \cdot \alpha^\Delta$ we have that our algorithm runs in at most

$$n^{3.67 \cdot c} \cdot 3.67 \cdot c \cdot \log n \cdot n \cdot \alpha^\Delta$$

computational steps. This can be rewritten as $O(n^{3.67c+2})$. □

Finally, in [12] an algorithm is suggested for computing an ϵ-approximate mixed Nash equilibrium of a graphical game with $O(\log n)$ treewidth, bounded

neighborhood size and bounded number of strategies per player. The approach is based on a reduction, implicit in [28], of the problem of finding mixed Nash equilibria to the problem of finding pure Nash equilibria. Our approach improves on the time complexity, again, by removing Δ from the exponent. The algorithm implied is polynomial in n and $\frac{1}{\epsilon}$ (PTAS).

7.7 Constructing pure Nash equilibria

When a PNE exists all tables A_i, $\forall i \in \mathcal{T}$ are computed; we are able to use these tables to construct a configuration that corresponds to a PNE. Note that the collection of tables $\mathcal{A} = \{A_i | i \in \mathcal{T}\}$ contains information about *all* the PNE of the game instance. Since all PNE can be exponentially many to the input size, it should suffice to have a structure that is bounded polynomially to the input size. Consider the following definition.

Definition 17. *Given a game $\mathcal{G} = (G, \mathcal{M})$, let $\mathcal{S}_\mathcal{G}$ be the set of all PNE. A succinct description of $\mathcal{S}_\mathcal{G}$ is a string y such that $|y|$ is polynomial in $|\mathcal{G}|$ and $\mathcal{S}_\mathcal{G} = f(y)$ for some function f computable in time polynomial to $|\mathcal{G}| + |y|$.*

We notice that $\mathcal{A}$ constitutes a succinct description of all pure Nash equilibria ($|\mathcal{A}|$ is polynomial to the size of the game and the set of all PNE can be computed in time polynomial in $|\mathcal{G}| + |\mathcal{A}|$).

To construct a *sample* PNE, we perform a breadth-first iteration of the bags beginning from the root. The key observation is that for each player $p \in V$ the action(s) that are part of a PNE configuration, are the ones that maximize the table A_i of node $i \in I$ such that $p \in X_i$ and i has the minimum distance from the root. The iteration ends when all vertices $p \in V$ have been encountered at least once. For each bag $X_i \in \mathcal{T}$ we need the configurations $C \in St(X_i)$ such that $A_i(C) = |V_i|$. These configurations can be marked while computing the tables, so the algorithm does not have to iterate again through the α^w different elements. Then, a PNE is constructed by filling in a table S, with $|S| = |V|$, the action suggested by the configuration that maximizes the table A_i. While the algorithm proceeds lower, the table is expanded by using the configurations that match with the current state of the table. When this finishes, table S will consist of a configuration

corresponding to a PNE. Observe that in the above procedure we do not need to read any information from the players' matrices $M_p \in \mathcal{M}$.

Theorem 21. *Given a graphical game $(G, \mathcal{M})$ and a tree decomposition $\mathcal{T}$ of width w, there is an algorithm that constructs a PNE, if one exists, or answers NO otherwise, in $O(\alpha^w \cdot w \cdot |\mathcal{M}|)$ time. Moreover, the same algorithm computes a succinct description of all PNE.*

Proof. We will describe a simple algorithm that, given the collection of tables $\mathcal{A}$ after the execution of the algorithm, constructs a PNE in $O(n)$ time. For this, we use a table S, with $|S| = n$ to represent the solution of the problem. With S_p we denote the position of the table that is indexed by player p. Begin at the root r of $\mathcal{T}$ and choose an arbitrary configuration $C \in St(X_r)$ such that $A_r(C) = |V|$. For each player $p \in X_r$ we set $S_p = C_p$. Then we iterate through the vertices of the tree decomposition in a breadth first manner. This top-down iteration terminates when all vertices $p \in V$ have been visited at least once. Let X_j be the bag under consideration. Observe that X_j possibly contains an unvisited vertex only if its parent X_i is a forget node. Let this player be $p \in X_j - X_i$. We find configuration $C \in St(X_j)$ such that $\forall u \in X_i$, $C_u = S_u$ and $A_j(C) = A_i(C \backslash \{C_p\})$ (remember that $G_i = G_j$) and set $S_p = C_p$.

We assume that the maximizing configurations have been marked when computing the values of the tables A_i and thus the enumeration at the root does not need to go through all α^w possible configurations. The procedure at the children of forget nodes does not increase the time complexity: the configuration $C \in St(X_j)$ that intersects with S and contains additionally action a_p such that $A_j(C) = A_i(C \backslash \{C_p\})$ can be found in a constant number of steps (this can be arranged, e.g. by ordering the configurations that maximize A_j lexicographically while the algorithm is executed). Therefore the whole algorithm takes at most $O(n)$ steps. □

7.7.1 Max Payoff Pure Nash Equilibrium

Finally, we will consider the problem of computing a *max payoff* pure Nash equilibrium. That is, a PNE where each individual receives the maximum possible payoff. Consider the following parameterized problem:

w-MP-PNE-GG

Input: $\mathcal{G} = (G, \mathcal{M})$, $\mathcal{T}$ tree decomposition of G, integer k.
Parameter: w - the treewidth of G.
Question: Is there a PNE C in $\mathcal{G}$ s.t. $u_p(C) \geq k$, $\forall p \in P$?

It is shown here that this problem can also be solved by a fixed-parameter algorithm when the number of available strategies is bounded by a constant. This problem is only slightly harder than deciding the existence of a PNE; it is dealt with by keeping payoff information for each vertex. This is achieved by introducing a new family of tables that have one entry per player per node of the tree decomposition. A table contains the maximum payoff a player can receive in a PNE state. We denote this table as $B_{\langle i,p\rangle}$, where i is a node of $\mathcal{T}$ with $p \in X_i$. We assume that all payoffs are integer to avoid arithmetic precision difficulties that arise by treating real numbers. It is obvious that in the case of leaf nodes we do not need to have any B table.

Let X_i be an *introduce node*, with child X_j and player $p \in X_i - X_j$. Using the method presented in previous section we use Algorithm 2 for each configuration $C \in St(X_j)$, after we have computed $\beta_p(C)$. Then, for $C \in St(X_j)$ and $a_p \in \beta_p(C)$ we take $B_{\langle i,p\rangle}(C \cup \{a_p\})$ to be the maximum payoff for p with respect to C_p. This can be found during the computation of $\beta_p(C)$. For a vertex $u \neq p$ we take $B_{\langle i,u\rangle}(C \cup \{a_p\})$ to be the maximum payoff of u when she plays C_u with respect to $C \cup \{a_p\}$. This value can be found while $\beta_u(C \cup \{a_p\})$ is computed at line 3 of Algorithm 2. For any configuration such that $A_i(C \cup \{a_p\}) < |V_i|$ we take $B_{\langle i,u\rangle}(C \cup \{a_p\}) = -\infty$. Note that here arises the following difficulty: nodes in X_j should take into account their neighbors that are forgot, since probably not all strategies from these players can be taken into account while looking for the value of the maximum payoff. This will be tackled with at forget nodes.

Let X_i be a *forget node*, with child X_j and player $p \in X_j - X_i$. Let $St^C(p) = \{a_p | a_p \in St(p) \text{ and } A_j(C \cup \{a_p\}) = |V_j|\}$. For $u \in X_i$ and $C \in St(X_i)$, we take $B_{\langle i,u\rangle}(C) = \max_{a_p \in St^C(p)} B_{\langle j,u\rangle}(C \cup \{a_p\})$. To tackle with the problem of remembering the best reply strategies of forgotten players we will use a simple coloring of the matrix M_u, using $|St(u)|$ colors, for

each player $u \in X_i$. When p is forgot, she colors the matrices of players in $\mathcal{N}(p) \cap X_j$ in the following way: Assume each player u of the game can color the cells of their matrices corresponding to the available configurations over their neighborhoods (headers of the columns) with $|St(u)|$ different colors. For configuration $C \in St(X_i)$ and a player $u \in X_i$, let color c_a correspond to $a \in St(u)$. Then, when p is forgot the columns corresponding to actions in $St^C(p)$ are colored c_a where $a = C_u$. This way, at introduce nodes, player u takes into account the possible actions of her forgotten neighbors before updating her maximum possible payoff. Note that more than one color is allowed per column.

Let X_i be a *join node*, with children X_{j_1} and X_{j_2}. For $p \in X_i$ we take $B_{\langle i,p\rangle}(C) = \min\{B_{\langle j_1,p\rangle}(C), B_{\langle j_2,p\rangle}(C)\}$. It is obvious that each player's maximum payoff is the minimum it may receive in each of the subgraphs G_{j_1}, G_{j_2}.

Time complexity. At introduce nodes, extra steps are needed to compute each table $B_{\langle i,u\rangle}$ for node X_i and player $u \in X_i$. These count to one step per player since the values can be computed while executing Algorithm 2 and thus the complexity at introduce nodes does not increase. The same argument holds for join nodes.

At forget nodes the algorithm might have to update the colors of the matrix for every player in X_i. In the worst case, this would take $\sum_{u \in X_i} |M_u|$. Therefore, forget nodes have a worst case complexity that is similar to that of introduce nodes. Even though more computational steps than the decision algorithm is needed in this case, the general worst-case complexity remains the same.

Finally, the construct algorithm can be modified to follow the maximizing B values for each player while computing a PNE. Before the execution of the algorithm the available actions for each player $p \in V$ should be ordered according to the value in the table $B_{\langle i,p\rangle}$ where i is either the root or the node where p is forgot (in other words, the last node in the bottom-up

iteration where p appears). Of course, this counts only for configurations $C \in St(X_i)$ such that $A_i(C) = |V_i|$.

Theorem 22. *Given a graphical game $(G, \mathcal{M})$ and a tree decomposition $\mathcal{T}$ of width w, there is an algorithm that constructs a maximum payoff PNE, if one exists, or answers NO otherwise, in $O(\alpha^w \cdot w \cdot |\mathcal{M}|)$ time.*

8 Conclusions

We provide a novel approach for computing pure Nash equilibria of a graphical game that is polynomial for games of bounded treewidth. Our method is based directly on the combinatorics of the problem, in contrast to previous results. When compared to the algorithm of [12], it improves the time bound by a removing a factor of $\max_{p \in P} |\mathcal{N}(p)|$ from the exponent. Even though we prove that the problem is $W[1]$-Hard in the general case, our algorithm is fixed-parameter tractable when the available strategies per player are bounded. In addition, we describe how to solve the problem in polynomial time for games of $O(\log n)$ treewidth, improving on the previous results, first by dropping the bounded neighborhood assumption and second by improving the time complexity in the bounded case. Finally, constructing a sample, or the maximum payoff, pure Nash equilibrium is also possible without additional computational effort.

8.1 Graphical vs action-graph games

After presenting our algorithm in Section 7, a fruitful criterion for comparison of the two succinct game representations arises. The relation of treewidth to the computation of pure Nash equilibria is different for each model and thus there appears to be an interesting *dichotomy*. From Theorem 2, AGGs are intractable in the general case, even if the treewidth of the input graph is 1. Note that in the proof of the theorem each player has only two (constant) available strategies. However, our results imply differently for graphical games.

In the light of Theorem 18, computing a PNE can be accomplished in linear time for classes of games with bounded treewidth and a bounded

number of available actions per player. This implies a linear algorithm for these restricted cases and thus graphical games of constant treewidth allow for efficient PNE computation. On the other hand, we remind that the computation of mixed Nash equilibria for both models is PPAD-Hard [11, 13]. The computational properties of these models rely on the way, and thus the size, of representation. In conclusion, one might say that graphical games are computationally more attractive than action-graph games (especially when the target solution is pure Nash equilibria).

8.2 Future directions

There are several interesting problems that remain open. We outline a few as an epilogue to this thesis.

- Is computing a PNE of a symmetric action-graph game fixed-parameter tractable (perhaps for restricted classes with some value bounded)? Would it be efficient to use an approach similar to the one used in this thesis?

- When there is no PNE for the input game, is our algorithm able to find the maximum subgraph that admits a PNE? If not, how is the computation of this subgraph possible?

- Could the time complexity of the suggested algorithm be improved, or is it tight?

- Does the fixed-parameter tractability of deciding a PNE help to attain FPT results for other problems by modeling them as PNE computations of graphical games? Note that in such a reduction the number of available strategies per player should be bounded. In addition, the max payoff PNE computation can also be used and might be more suitable for optimization problems. Of course having a proof that *social welfare optimizing* PNE is FPT would give more value to the explained procedure. In any case, such a methodology has to deal with a number of non-trivial matters, such that the size of the matrices of

a graphical game should be exponential only in treewidth, a difficulty we also dealt with for the proof of Theorem 16.

References

[1] Ann Becker and Dan Geiger. **A sufficiently fast algorithm for finding close to optimal clique trees**. *Artif. Intell.*, **125**(1-2):3–17, 2001. 32, 49

[2] Navin A. R. Bhat and Kevin Leyton-Brown. **Computing Nash Equilibria of Action-Graph Games**. In David Maxwell Chickering and Joseph Y. Halpern, editors, *UAI*, pages 35–42. AUAI Press, 2004. 3, 14

[3] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. **On problems without polynomial kernels**. *Journal of Computer and System Sciences*, **75**(8):423–434, 2009. 9

[4] Hans L. Bodlaender. **A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth**. *SIAM J. Comput.*, **25**(6):1305–1317, 1996. 31

[5] Hans L. Bodlaender and Arie M. C. A. Koster. **Combinatorial Optimization on Graphs of Bounded Treewidth**. *Comput. J.*, **51**(3):255–269, 2008. 40, 42, 44, 47

[6] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. **Kernel Bounds for Disjoint Cycles and Disjoint Paths**. In Amos Fiat and Peter Sanders, editors, *ESA*, **5757** of *Lecture Notes in Computer Science*, pages 635–646. Springer, 2009. 9, 48

[7] Felix Brandt, Felix A. Fischer, and Markus Holzer. **Symmetries and the complexity of pure Nash equilibrium**. *J. Comput. Syst. Sci.*, **75**(3):163–177, 2009. 11

[8] Xi Chen, Decheng Dai, Ye Du, and Shang-Hua Teng. **Settling the Complexity of Arrow-Debreu Equilibria in Markets with Additively Separable Utilities**. *Foundations of Computer Science, Annual IEEE Symposium on*, **0**:273–282, 2009. 3

[9] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. **Settling the complexity of computing two-player Nash equilibria**. *J. ACM*, **56**(3), 2009. 2

[10] Xi Chen and Shang-Hua Teng. **Spending Is Not Easier Than Trading: On the Computational Equivalence of Fisher and Arrow-Debreu Equilibria**. In Yingfei Dong, Ding-Zhu Du, and Oscar H. Ibarra, editors, *ISAAC*, **5878** of *Lecture Notes in Computer Science*, pages 647–656. Springer, 2009. 3

[11] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. **The Complexity of Computing a Nash Equilibrium**. *SIAM J. Comput.*, **39**(1):195–259, 2009. 2, 55

[12] Constantinos Daskalakis and Christos H. Papadimitriou. **Computing pure nash equilibria in graphical games via markov random fields**. In Joan Feigenbaum, John C.-I. Chuang, and David M. Pennock, editors, *ACM Conference on Electronic Commerce*, pages 91–99. ACM, 2006. 3, 4, 24, 30, 31, 32, 38, 48, 49, 54

[13] Constantinos Daskalakis, Grant Schoenebeck, Gregory Valiant, and Paul Valiant. **On the complexity of Nash equilibria of action-graph games**. In Claire Mathieu, editor, *SODA*, pages 710–719. SIAM, 2009. 2, 19, 20, 21, 55

[14] Nikhil R. Devanur, Christos H. Papadimitriou, Amin Saberi, and Vijay V. Vazirani. **Market equilibrium via a primal–dual algorithm for a convex program**. *J. ACM*, **55**(5), 2008. 3

[15] Kousha Etessami and Mihalis Yannakakis. **On the Complexity of Nash Equilibria and Other Fixed Points**. *SIAM J. Comput.*, **39**(6):2531–2597, 2010. 2, 3

[16] Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar. **The complexity of pure Nash equilibria**. In László Babai, editor, *STOC*, pages 604–612. ACM, 2004. 11

[17] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen. **On the complexity of some colorful problems parameterized by treewidth**. *Inf. Comput.*, **209**(2):143–153, 2011. 34

[18] Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. **On the parameterized complexity of multiple-interval graph problems**. *Theor. Comput. Sci.*, **410**(1):53–61, 2009. 34

[19] Lance Fortnow and Rahul Santhanam. **Infeasibility of instance compression and succinct PCPs for NP**. *J. Comput. Syst. Sci.*, **77**(1):91–106, 2011. 48

[20] Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. **Pure Nash Equilibria: Hard and Easy Games**. *J. Artif. Intell. Res. (JAIR)*, **24**:357–406, 2005. 3, 6, 12, 25, 27, 28, 29, 30, 33, 46, 48

[21] Georg Gottlob, Martin Grohe, Nysret Musliu, Marko Samer, and Francesco Scarcello. **Hypertree Decompositions: Structure, Algorithms, and Applications**. In Dieter Kratsch, editor, *WG*, **3787** of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2005. 7, 47

[22] Georg Gottlob, Nicola Leone, and Francesco Scarcello. **A comparison of structural CSP decomposition methods**. *Artif. Intell.*, **124**(2):243–282, 2000. 29

[23] Martin Grohe. **The complexity of homomorphism and constraint satisfaction problems seen from the other side**. *J. ACM*, **54**(1), 2007. 33

[24] Finn V. Jensen, Steffen L. Lauritzen, and Kristian G. Olesen. **Bayesian updating in causal probabilistic networks by local computations**. *Computational Statistics Quarterly*, **4**:269–282, 1990. 31

[25] Albert Xin Jiang and Kevin Leyton-Brown. **Computing Pure Nash Equilibria in Symmetric Action Graph Games**. In *AAAI*, pages 79–85. AAAI Press, 2007. 16, 17, 23, 24

[26] Albert Xin Jiang, Kevin Leyton-Brown, and Navin A.R. Bhat. **Action-Graph Games**. *Games and Economic Behavior*, **71**(1):141 – 173, 2011. Special Issue In Honor of John Nash. 16, 17

[27] Albert Xin Jiang and MohammadAli Safari. **Pure Nash equilibria: complete characterization of hard and easy graphical games**. In Wiebe van der Hoek, Gal A. Kaminka, Yves Lespérance, Michael Luck, and Sandip Sen, editors, *AAMAS*, pages 199–206. IFAAMAS, 2010. 33

[28] Michael J. Kearns, Michael L. Littman, and Satinder P. Singh. **Graphical Models for Game Theory**. In Jack S. Breese and Daphne Koller, editors, *UAI*, pages 253–260. Morgan Kaufmann, 2001. 3, 7, 11, 12, 32, 50

[29] Shiva Kintali, Laura J. Poplawski, Rajmohan Rajaraman, Ravi Sundaram, and Shang-Hua Teng. **Reducibility among Fractional Stability Problems**. *Foundations of Computer Science, Annual IEEE Symposium on*, **0**:283–292, 2009. 2

[30] Ton Kloks. *Treewidth, Computations and Approximations*, **842** of *Lecture Notes in Computer Science*. Springer, 1994. 41

[31] Steffen L. Lauritzen. *Graphical Models*. Oxford University Press, 1996. 30

[32] Dániel Marx. **Can You Beat Treewidth?** *Theory of Computing*, **6**(1):85–112, 2010. 29, 30

[33] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, USA, 2006. 8, 9

[34] Christos H. Papadimitriou. **Algorithms, games, and the internet**. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, STOC '01, pages 749–753, New York, NY, USA, 2001. ACM. 2

[35] Neil Robertson and Paul D. Seymour. **Graph Minors. II. Algorithmic Aspects of Tree-Width**. *J. Algorithms*, **7**(3):309–322, 1986. 7

[36] Robert W. Rosenthal. **A class of games possessing pure-strategy Nash equilibria**. *International Journal of Game Theory*, **2**(1):65–67, 12/01 1973. 11

[37] Tim Roughgarden. **Computing equilibria: a computational complexity perspective**. *Economic Theory*, **42**(1):193–236, 2010. 2

[38] Grant Schoenebeck and Salil P. Vadhan. **The computational complexity of nash equilibria in concisely represented games**. In Joan Feigenbaum, John C.-I. Chuang, and David M. Pennock, editors, *ACM Conference on Electronic Commerce*, pages 270–279. ACM, 2006. 11, 19, 25

[39] Stefan Szeider. **On Fixed-Parameter Tractable Parameterizations of SAT**. In Enrico Giunchiglia and Armando Tacchella, editors, *SAT*, **2919** of *Lecture Notes in Computer Science*, pages 188–202. Springer, 2003. 48