Algorithmica (2015) 71:581–604
DOI 10.1007/s00453-014-9923-3

Pure Nash Equilibria in Graphical Games and Treewidth

Antonis Thomas · Jan van Leeuwen

Received: 21 November 2013 / Accepted: 24 July 2014 / Published online: 22 August 2014

Abstract We treat PNE-GG, the problem of deciding the existence of a Pure Nash Equilibrium in a graphical game, and the role of treewidth in this problem. PNE-GG is known to be NP-complete in general, but polynomially solvable for graphical games of bounded treewidth. We prove that PNE-GG is $W[1]$-Hard when parameterized by treewidth. On the other hand, we give a dynamic programming approach that solves the problem in $O^*(\alpha^w)$ time, where α is the cardinality of the largest strategy set and w is the treewidth of the input graph (and O^* hides polynomial factors). This proves that PNE-GG is in FPT for the combined parameter (α, w). Moreover, we prove that there is no algorithm that solves PNE-GG in $O^*((\alpha - \epsilon)^w)$ time for any $\epsilon > 0$, unless the Strong Exponential Time Hypothesis fails. Our lower bounds implicitly assume that $\alpha \geq 3$; we show that for $\alpha = 2$ the problem can be solved in polynomial time. Finally, we discuss the implication for computing pure Nash equilibria in graphical games (PNE-GG) of $O(\log n)$ treewidth, the existence of polynomial kernels for PNE-GG parameterized by treewidth, and the construction of a sample and maximum-payoff pure Nash equilibrium.

Keywords Nash equilibria · Graphical games · Parameterized complexity · Treewidth

A. Thomas (✉)
Institute of Theoretical Computer Science, ETH Zurich, 8092 Zurich, Switzerland
e-mail: athomas@inf.ethz.ch

J. van Leeuwen
Department of Information and Computing Sciences, Utrecht University,
3584 CC Utrecht, The Netherlands
e-mail: J.vanLeeuwen1@uu.nl

1 Introduction

The computation of Nash equilibria in finite games is a fundamental class of problems in algorithmic game theory. Several recent breakthroughs have settled the complexity of computing (mixed) Nash equilibria for any number of players ≥ 2 [6,7]. These equilibria are guaranteed to exist, but their suitability as models of behavior and rationality can be debated as they involve deliberate randomization by players and are based on the assumption that all players are risk neutral [8]. In this paper we consider *pure Nash equilibria* (PNE), which are more intuitive even though they do not exist in all games.

We especially focus on the computational aspects of pure Nash equilibria in so-called *graphical games*, introduced by Kearns et al. [25]. A graphical game consists of a graph and a collection of matrices—one for each player. A player is represented by a vertex in the input graph and her payoff is determined entirely by her action and that of her neighbors. Graphical games are a prime example of a class of games that are 'succinct represented' and computationally meaningful [30].

Let PNE-GG denote the problem of deciding the existence of a pure Nash equilibrium in a graphical game. PNE-GG was proved to be NP-complete by Gottlob et al. [17], even in the restricted case of neighborhoods of size at most 3 and at most 3 strategies per player. On the other hand, they proved that the problem is tractable when the underlying graph has bounded treewidth. Treewidth is a widely studied structural parameter that measures the extent to which a graph deviates from a basic tree structure (cf. Sect. 2).

In this paper we give an extensive analysis of the computational complexity of PNE-GG as it depends on the treewidth of the underlying graph. We will especially position PNE-GG in the class of problems that are *fixed parameter tractable*, using treewidth as the parameter. Various related questions will be addressed. This paper extends the first presentation of our results in [35].

Our results can be summarized as follows. As a first main result we prove that PNE-GG is $W[1]$-hard for the parameter treewidth (Sect. 4). Next, we develop a tedious dynamic programming method to compute pure Nash equilibria in games of bounded treewidth (Sect. 5). Our algorithm runs in $O^*(\alpha^w)$ time, where α is the size of the largest strategy set and w is the treewidth of the input graph (and O^* hides polynomial factors). It follows that PNE-GG is in FPT for the combined parameter (α, w), hence notably so when the cardinality of the strategy sets is bounded. Subsequently, we discuss the implications of our algorithm for games of $O(\log n)$ treewidth and we argue that PNE-GG for graphs of bounded treewidth and constant strategy sets does not admit a polynomial kernel unless $NP \subseteq coNP/poly$. In addition, we show how our algorithm is able to efficiently construct a sample or the maximum-payoff equilibrium (Sect. 7).

Finally, we note that the time bound of our algorithm is most likely tight. We prove that any improvement of the running time to a bound of the sort $O^*((\alpha - \epsilon)^w)$, for some $\epsilon > 0$, would cause the Strong Exponential Time Hypothesis to fail (Sect. 6). This is considered unlikely. Note that our lower bounds (from both Sects. 4 and 6) implicitly assume that $\alpha \geq 3$, while the proof of the NP-completeness of PNE-GG has the same assumption. We justify this by showing that for $\alpha = 2$, PNE-GG can be solved in polynomial time (Sect. 3).

1.1 Related Work

Gottlob et al. [17] proved PNE-GG to be polynomial on graphs of bounded treewidth. They proved this by mapping a graphical game to a constraint satisfaction problem (CSP) while maintaining pure Nash equilibria as solutions of the resulting instance. The time complexity of their procedure is $O(||\mathcal{G}||^{w+1} \cdot \log ||\mathcal{G}||)$, thus exponential in w, where $||\mathcal{G}||$ is the size of the description of the game instance and w the treewidth of the underlying graph. Marx showed that the algorithm for solving CSP is essentially optimal under the Exponential Time Hypothesis [28] and thus, faster algorithms are not expected using this approach.

A different approach was provided by Daskalakis and Papadimitriou [8]. They attacked the problem by providing a reduction from graphical games to Markov random fields. This yields a unified proof to the previously known tractable cases with time complexity $O(n \cdot |M_p|^{w+1}) = O(n \cdot \alpha^{\Delta \cdot (w+1)})$, where n is the number of players, p is the player with the largest neighborhood (of size Δ) and M_p its local game matrix (cf. Sect. 2). Their result implies that the class of games with $O(\log n)$ treewidth and bounded Δ and α, is tractable. We show that the dependence on Δ can be greatly simplified, removing it from the exponent in the running time and thus obtaining tractability regardless of the value of Δ.

Furthermore, Jiang and Leyton-Brown [22] provide an algorithm for another class of succinctly represented games, namely *action graph games*, that is polynomial for symmetric action graph games of bounded treewidth. It is known that any graphical game can be mapped to a non-symmetric action graph game [23]. For bounded cardinality strategy sets this mapping keeps the treewidth bounded. However, computing pure Nash equilibria for non-symmetric action-graph games is NP-complete even when the treewidth is 1 [9].

Greco and Scarcello build on [17] and provide a dynamic programming approach that decides, in polynomial time, the existence of constrained pure Nash equilibria for graphical games of bounded treewidth with a bounded number of constraints [18]. Their approach is based on a non-deterministic algorithm, implicitly provided in [25], that associates pure with approximate mixed equilibria. A quite different approach is provided by Jiang and Safari in [24]. The parameter they consider is the representational size of the graph, say $p = |V| + |E|$. They show that for every recursively enumerable class of graphical games with bounded treewidth, deciding p-PNE-GG is in FPT if and only if deciding PNE-GG is in P. Observe that none of the known methods implies fixed-parameter tractability of PNE-GG with treewidth of the input graph as the parameter.

Finally, there are recent papers that treat game-theoretic problems from the perspective of parameterized complexity as well. Estevill-Castro and Parsa [12] initiated the study of computing Nash equilibria of bimatrix games from the perspective of parametrized complexity theory. Define the support of a player as the collection of strategies which she plays with non-zero probability. Estevill-Castro and Parsa [12] proved that for bimatrix games, both the existence of a k-uniform Nash equilibrium (with strategies having either zero or uniform probabilities for each action) and finding a k-minimal Nash equilibrium (one of minimal support), are W[2]-hard, where the parameter k is the size of the support. In a subsequent study [13] they show that the existence of a k-uniform Nash equilibrium is in FPT for a restricted class of symmetric 0, 1-win-lose bimatrix games, where the parameter is the treewidth of the undirected graph that arises when one interprets the payoff matrix of a player as an adjacency matrix [13]. Supplementing these results, Hermelin et al. [19] show, among other things, that computing a Nash equilibrium of an ℓ-sparse bimatrix game is in FPT time for the combined parameter (ℓ, k) (where k is, again, the size of the support).

2 Preliminaries

2.1 Graphical Games

In a graphical game with graph $G = (V, E)$ we have $n = |V|$ players and each player $p \in V$ has a finite set of strategies, each strategy $St(p)$ being a finite set of actions with $|St(p)| \geq 2$. The cardinality of the largest strategy set is denoted with $\alpha = \max_{p \in V} |St(p)|$. For a non-empty set of players $P \subseteq V$ a joint strategy or *configuration* $\mathcal{C}$ is a set containing exactly one strategy for each player $p \in P$. The set of all joint strategies of players in P is denoted as $St(P)$ and thus we write $\mathcal{C} \in St(P)$. For a player p, $\mathcal{C}_p$ denotes the strategy of player p with respect to configuration $\mathcal{C}$ and $\mathcal{C}_{-p}$ denotes the configuration resulting from removing the strategy suggested for p in $\mathcal{C}$. Additionally, for every $a_p \in St(p)$ and $\mathcal{C}_{-p} \in St(V \setminus \{p\})$ we denote by $(\mathcal{C}_{-p}; a_p)$ the configuration in which p plays a_p and every other player $p' \neq p$ plays according to $\mathcal{C}$. Abusing notation, we use $\mathcal{C} \cup \{a_p\}$ to denote the configuration resulting by adding strategy $a_p \in St(p)$ to configuration $\mathcal{C} \in St(P')$ where $p \notin P'$. A configuration $\mathcal{C}$ is termed *global* if it is over the set of all players ($\mathcal{C} \in St(V)$). The global configurations are the possible outcomes of the game. We define the neighborhood of player $p \in V$ as $\mathcal{N}(p) = \{u \in V | (p, u) \in E\}$.

Definition 1 ([25]) 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 joint strategy $\mathcal{C}$, the local game matrix $M_p \in \mathcal{M}$ specifies the payoff $M_p(\mathcal{C})$ for player $p \in V$, which depends only on the actions taken by p and the players in $\mathcal{N}(p)$.

Note that for graphical games on undirected graphs, players' interests are necessarily symmetric, i.e. for any pair of players p_1 and p_2, $p_1 \in \mathcal{N}(p_2)$ if and only if $p_2 \in \mathcal{N}(p_1)$. Let the size of the collection of matrices be $|\mathcal{M}| = \sum_{p \in V} |M_p|$.

Definition 2 The best response function of a player p is a function $\beta_p : St(\mathcal{N}(p)) \rightarrow 2^{St(p)}$, with respect to a (global) configuration $\mathcal{C}$, such that:

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

Intuitively, $\beta_p(\mathcal{C})$ is the set of strategies that maximize the payoff of player p when the players in p's neighborhood play according to (global) configuration $\mathcal{C}$.

Consequently, a pure Nash equilibrium (PNE for short) is a global configuration $\mathcal{C}$ such that for every player $p \in V, \mathcal{C}_p \in \beta_p(\mathcal{C}_{-p})$. Alternatively:

Definition 3 A global configuration $\mathcal{C}$ is a pure Nash equilibrium if for every player p and strategy $a_p \in St(p)$ we have $M_p(\mathcal{C}) \geq M_p(\mathcal{C}_{-p}; a_p)$.

2.2 Treewidth and Mixed Search Games

First we give the definition of a tree decomposition and of treewidth.

Definition 4 ([31]) A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i | i \in I\}, T = (I, F))$, where T is a tree and 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, $\mathtt{twd}(G)$, is the minimum width over all tree decompositions of G. If Definition 4 is restricted to paths instead of trees, we speak of the *pathwidth* of a graph G, denoted by $\mathtt{pwd}(G)$.

As an alternative way to characterize treewidth we use the notion of a *mixed search game*. The game is played on a graph G by placing tokens, called searchers, on the vertices of the graph and moving them along the edges to other vertices. The edges of the graph are initially contaminated by a gas. They get cleared if either (i) a searcher goes through the edge or (ii) two searchers are simultaneously placed at the two endpoints of that edge. In addition, a cleared edge gets re-contaminated if there is a path from a contaminated edge to the cleared edge with no searchers on its vertices. A search strategy consists of the operations of adding/removing searchers to the graph vertices and of sliding searchers across edges to neighboring vertices. If after the termination of the strategy all edges are cleared then we have a *winning* strategy. The *mixed search number* of a graph $G, \mathtt{msn}(G)$ is the minimum number of searchers needed for a winning strategy. Our interest here lies in bounding the treewidth by describing a mixed search strategy. It is known that $\mathtt{pwd}(G) \leq \mathtt{msn}(G) \leq \mathtt{pwd}(G)+1$ [34]. It also holds that $\mathtt{twd}(G) \leq \mathtt{pwd}(G)$ which gives us mixed search games as a tool to bound the treewidth of a graph (cf. proof of Lemma 8).

2.3 Complexity

We give the definitions of some basic concepts from the theory of parameterized complexity [10,16,29].

Definition 5 ([10,16,29]) 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 parameters we consider here are nonnegative integers, hence we write $L \in \Sigma^* \times \mathbb{N}$ from now on. For $(x, k) \in L$, the two dimensions of parameterized complexity are the input size n, $n = |(x, k)|$, and the parameter value k.

Definition 6 ([10,16,29]) 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 parameterized problems of the form (x, k), that are solvable in time $f(k) \cdot n^{O(1)}$, is denoted as FPT. In order to prove hardness for parameterized problems we also need a reducibility concept.

Definition 7 ([10,16,29]) Let (Q, k) and (Q', k') be parameterized problems over the alphabets Σ and Σ'. An *fpt-reduction* from (Q, k) to (Q', k') is a mapping $R : \Sigma^* \rightarrow (\Sigma')^*$ such that:

– $\forall x \in \Sigma^*$ we have $(x \in Q \Leftrightarrow R(x) \in Q')$;
– R is computable in FPT time (with respect to k);
– there exists a computable function $g : \mathbb{N} \rightarrow \mathbb{N}$ such that for all $x \in \Sigma^*$, if k is the parameter of x and k' is the parameter of $R(x)$, then $k' \leq g(x)$.

Fixed-parameter intractability beyond FPT is captured in the W*-hierarchy* (cf. [10,16,29]). A parameterized problem is $W[1]$-hard if WEIGHTED 3SAT is reducible to it by an fpt-reduction. WEIGHTED 3SAT is the problem of deciding whether a 3-CNF formula has a satisfying assignment of Hamming weight k, with k as the parameter. The Hamming weight of an assignment is the number of literals set to true. While it is known that $FPT \subseteq W[1]$, it is currently open whether $FPT \subset W[1]$.

It is known that every parameterized problem in FPT has a *kernel* [10], i.e. every instance of the problem can be reduced by a polynomial-time parameterized reduction to an equivalent instance whose size only depends on the parameter. This gives a potentially very powerful way of preprocessing a problem, reducing instances in polynomial time to a smaller kernel which can be solved by an algorithm that we can afford to be inherently exponential. Bounding the size of kernels of problems in FPT has been the subject of extensive research. We refer to Kratsch [26] for an excellent introduction to the relevant notions and for many recent results.

Finally, we describe the (Strong) Exponential Time Hypothesis, (S)ETH, originally conjectured by Impalliagzo et al. [21]. Consider k-SAT, a version of the SAT problem where each clause of the given formula ϕ has at most k literals. Let $s_k = \inf\{\delta \mid k\text{-SAT}$ is solvable in time $2^{\delta n}\}$. Then, the ETH asserts that $s_3 > 0$. In [21] it was shown that the ETH is robust in the sense that $s_3 > 0$ if and only if there is $k \geq 3$ such that $s_k > 0$. Moreover, in [20], it was shown that the sequence $s_3, s_4, \ldots$ increases infinitely often. Finally, the SETH asserts that $\lim_{k\rightarrow\infty} s_k = 1$.

3 Graphical Games with 2 Strategies

Here we prove that PNE-GG can be solved in polynomial time when each player has exactly two strategies available (2PNE-GG). For this we present a reduction from

2PNE-GG to UTVIP. The latter stands for Unit Two Variable Integer Programming. This is a restricted version of Integer Programming that is solvable in polynomial time. The problem statement is as follows: Let $A \in \mathbb{Z}^{m\times n}$ such that $\sum_{j=1}^{n} |a_{ij}| \leq 2, \forall i = 1, 2, \ldots m$, and let $b \in \mathbb{Z}^m$. Is there $x \in \mathbb{Z}^n$ such that $Ax \leq b$?

The restriction on A means that for every inequality there are at most two non-zero coefficients with values in $\{-1, 1\}$ or at most one non-zero coefficient with absolute value 2. UTVIP is known to be solvable in polynomial time by Schrijver (Proposition 6 in [32]). Later, it was shown to be solvable in $O(n \log n + m)$, where n is the number of variables and m the number of inequalities [33].

Theorem 1 *2PNE-GG is solvable in polynomial time.*

Proof We give a polynomial time reduction from 2PNE-GG to UTVIP. Let $\mathcal{G} = (G, \mathcal{M})$ be an instance of 2PNE-GG. Consider $\{0, 1\}$ as the strategy set for each player of the game. We take a variable x_i for each of the n players of $\mathcal{G}$ and two inequalities for each variable: $0 \leq x_i \leq 1, \forall i = 1, 2, \ldots, n$.

Now let (i, j) be an edge of G. We take at most two inequalities per edge, for each of the players i, j. If s is a strategy $\{0, 1\}$ with $\bar{s}$ we mean the opposite strategy. In addition, we introduce the notation $\beta_i^{\mathcal{C}_j=s}$ to simplify the presentation ($\beta_i^{\mathcal{C}_j=s}$ is defined if and only if i, j are neighbors in G):

$$\beta_i^{\mathcal{C}_j=s} = \bigcup_{\{\mathcal{C}|\mathcal{C}\in St(\mathcal{N}(i)) \text{ and } \mathcal{C}_j=s\}} \beta_i(\mathcal{C})$$

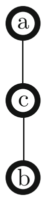
We perform a case distinction which we present from i's perspective. First consider the trivial cases, when i is indifferent to what j plays:

1. $\forall s \in \{0, 1\}, \beta_i^{\mathcal{C}_j=s} = \{0, 1\}$. Then, we take no inequality.
2. $\forall s \in \{0, 1\}, \beta_i^{\mathcal{C}_j=s} = k \in \{0, 1\}$. Then, we take the equality $x_i = k$.

Subsequently, we consider the more involved cases which are being considered only if case 2 above does not apply:

3. $\exists s \in \{0, 1\}$ such that $s = \beta_i^{\mathcal{C}_j=s}$ (coordination). Then, we take the inequalities $l_{ij} \leq x_i - x_j \leq u_{ij}$, where $l_{ij} \in \{-1, 0\}$ and $u_{ij} \in \{0, 1\}$.
   If $s = 0$, then $u_{ij} = 0$; $l_{ij} = 0$ if $\bar{s} = \beta_i^{\mathcal{C}_j=\bar{s}}$ and $l_{ij} = -1$ if $\beta_i^{\mathcal{C}_j=\bar{s}} = \{0, 1\}$.
   If $s = 1$, then $l_{ij} = 0$; $u_{ij} = 0$ if $\bar{s} = \beta_i^{\mathcal{C}_j=\bar{s}}$ and $u_{ij} = 1$ if $\beta_i^{\mathcal{C}_j=\bar{s}} = \{0, 1\}$.
4. $\exists s \in \{0, 1\}$ such that $\bar{s} = \beta_i^{\mathcal{C}_j=s}$ (anti-coordination). Then, we take the inequalities $l_{ij} \leq x_i + x_j \leq u_{ij}$, where $l_{ij} \in \{0, 1\}$ and $u_{ij} \in \{1, 2\}$.
   If $s = 0$, then $l_{ij} = 1$; $u_{ij} = 1$ if $s = \beta_i^{\mathcal{C}_j=\bar{s}}$ and $u_{ij} = 2$ if $\beta_i^{\mathcal{C}_j=\bar{s}} = \{0, 1\}$.
   If $s = 1$, then $u_{ij} = 1$; $l_{ij} = 1$ if $s = \beta_i^{\mathcal{C}_j=\bar{s}}$ and $l_{ij} = 0$ if $\beta_i^{\mathcal{C}_j=\bar{s}} = \{0, 1\}$.

In total, we take at most 4 inequalities per edge of G and exactly 2 inequalities per player/vertex of G. It follows from the reduction that the resulting UTVIP instance has a solution that satisfies all the constraints if and only if $\mathcal{G}$ admits a PNE. In addition, if a solution $x \in \mathbb{Z}^n$ exists it holds that x_i is a PNE strategy for player i. □



$$M_a \quad \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & +1 & +1 \\ 1 & -1 & +1 \end{array} \qquad -1 \le x_a - x_c \le 0$$

$$M_b \quad \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & -1 & +1 \\ 1 & +1 & -1 \end{array} \qquad 1 \le x_b + x_c \le 1$$

Fig. 1 An example of the reduction described in the proof of Theorem 1

Finally, we include a simple example in Fig. 1 to illustrate the proof presented above. The game consists of only three players, a, b, c, connected in a path. The matrices for the players a, b are included in the middle column. At the right column, we present the inequalities formed for the edges (a, c) and (b, c) from player a's and player b's perspective. The former is a coordination (falls in case 3): for $s = 0$ we have that $\beta_a^{\mathcal{C}_c=0} = \{0\}$ and $\beta_a^{\mathcal{C}_c=1} = \{0, 1\}$. The latter is an anti-coordination (falls in case 4): for $s = 0$ we have that $\beta_a^{\mathcal{C}_c=0} = \{1\}$ and $\beta_a^{\mathcal{C}_c=1} = \{0\}$.

4 W[1]-Hardness

Treewidth plays an important role in the study of pure Nash equilibria for graphical games (cf. [17]). However, 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 following parameterized problem:

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

We will prove that w-PNE-GG is $W[1]$-hard. For this, a reduction from the $W[1]$-hard problem k-MULTICOLOR CLIQUE will be used. The input of this problem is a graph $G = (V, E)$ and a vertex coloring $c : V \to \{1, \dots, k\}$, k is the parameter and the question is whether G contains a clique with vertices of all k colors. Hardness follows easily by reduction from k-CLIQUE [15].

Before proceeding to the reduction, we introduce some useful notation. Let G be the input graph, and $c : V \to \{1, \dots, k\}$ a k-coloring of G. We let V_a denote the vertices colored a, i.e. $V_a = \{v \in V | c(v) = a\}$, and we let E_{c_i,c_j} be the set of edges $(u, v) \in E$ such that $\{c(u), c(v)\} = \{c_i, c_j\}$. Observe that w.l.o.g we can assume that the input coloring is *proper*, i.e. for any color c, $E_{c,c} = \emptyset$, as any such edge can be removed from G [15]. W.l.o.g. 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 $c_i < c_j$ [14].

Theorem 2 *w-PNE-GG is $W[1]$-hard.*

Proof Given an instance of MULTICOLOR CLIQUE, graph $G = (V, E)$ with k-coloring c, we construct an instance $\mathcal{G} = (G' = (P, E'), \mathcal{M})$ of PNE-GG as follows: The

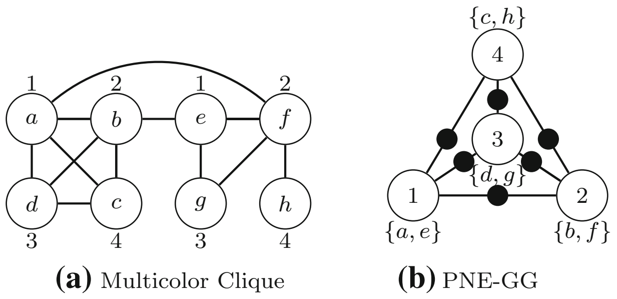


Fig. 2 An example of the reduction, where the numbers correspond to different colors. In (**b**) the strategy sets are shown in *curly brackets* (omitting NA) and the auxiliary players are represented as *black vertices*

players of $\mathcal{G}$ are separated in two distinct sets, the *colorful* P_c and the *auxiliary* P_a players, $P = P_c \cup P_a$. Each colorful player corresponds to a distinct color and each auxiliary player to a distinct pair of colors. Every $c \in P_c$ is connected to all the other colorful players $c' \in P_c$, through an auxiliary vertex $a \in P_a$. Thus, G' arises by taking a k-clique and adding one auxiliary player on each edge. By construction, 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 strategies are all the vertices of G that are colored c plus an extra NA strategy, 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 strategies, $St(p) = \{A, NA\}$, 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 Fig. 2.

The payoffs of $\mathcal{G}$ are as follows. For each player $p \in P$ we describe a payoff function[1] $u_p : St(P) \to \{0, 1\}$. Let $\mathcal{C}$ 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:

1. $u_a(\mathcal{C}) = 1$ if a plays A and i, j play v, u such that $(v, u) \in E$ or at least one of i, j plays NA;
2. $u_a(\mathcal{C}) = 1$ if a plays NA and i, j play v, u such that $(v, u) \notin E$ and neither of i, j plays NA;
3. $u_a(\mathcal{C}) = 0$ in all other cases.

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

4. $u_c(\mathcal{C}) = 1$ if c plays a strategy in $St(c) \setminus \{NA\}$, and all of her neighbors play A;
5. $u_c(\mathcal{C}) = 1$ if c plays NA and at least one of her neighbors plays NA;
6. $u_c(\mathcal{C}) = 0$ in all other cases.

In the following paragraphs we will show that G has a clique including all k colors if and only if $\mathcal{G}$ has a pure Nash equilibrium. Let $(v_1, \ldots, v_k)$ be a k-clique of G that

[1] Here we give the payoffs as a utility function over the global configurations. Of course, the only relevant part of the global configuration is the one referring to $\mathcal{N}(p)$ for a player $p \in P$. The only reason we use $St(P)$ as the domain of the function is to simplify the presentation. The same convention will be used in the proof of Theorem 4.

contains all k colors. Consider the global strategy $\mathcal{C}$ where each player $c \in P_c$ plays the strategy 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 $\mathcal{C}$ is a PNE and $\exists a \in P_a$ that plays NA, with neighbor $j \in \mathcal{N}(a)$. Then, j would have an incentive to play NA and get payoff 1 rather than a strategy in $St(j)\backslash\{NA\}$. Consequently, a would prefer A over NA which contradicts our assumption that $\mathcal{C}$ is a PNE.

Now, let $\mathcal{C}$ 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 a strategy in $St(j)\backslash\{NA\}$. Consider the set of vertices $K = (v_1, \dots, v_k)$ 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 $|P_c| = k$, that each player $c \in P_c$ has $k - 1$ auxiliary neighbors with 2 available strategies each, 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

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

The 2^{k-1} term corresponds to the number of possible configurations over the neighborhood of a colorful player, i.e. $|St(\mathcal{N}(c))| = 2^{k-1}$, for all $c \in P_c$. 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 $FPT = W[1]$. It is worth to mention that our construction proves hardness for a more general parameter, namely the number of vertices (which is $k + \binom{k}{2}$). Nevertheless, in the next section we will demonstrate an algorithm that becomes FPT for games with a bounded number of available strategies per player.

5 Towards Tractability

When the input graph of a graphical game is a tree, a relatively simple algorithm can answer the question of existence of a PNE in time linear in the input. The idea is

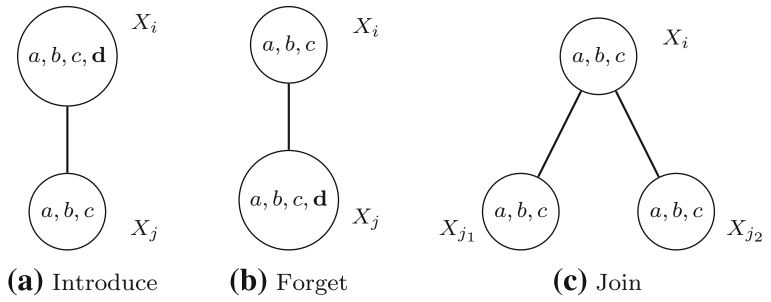


Fig. 3 An illustration of the three non-trivial type of nodes found in a nice tree decomposition

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 a subsequent step. The details of the algorithm and the proof of the result below are omitted as they are subsumed by the algorithm that runs on tree decompositions.

Proposition 1 *Given a graphical game* $(T, \mathcal{M})$*, where* T *is a tree, one can compute a PNE in time* $O(|\mathcal{M}|)$.

The idea of the tree algorithm will now be generalized to tree decompositions; the problem under consideration is w-PNE-GG as defined in the previous section. The intuition is to go through all possible configurations for each bag of the tree, which count to α^{w+1}. Then we put together this information on the tree decomposition in polynomial time. The analysis we provide, 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 $X_j = X_i \cup \{v\}$.

An illustration of the three non-trivial node types can be found in Fig. 3. It is known that if a graph $G = (V, E)$ has a tree decomposition with width at most w, then it also has a *nice* tree decomposition of width at most w and $O(w|V|)$ nodes. A given tree decomposition can be turned into a nice one in linear time [5].

5.1 A Dynamic Programming Approach

Suppose we are given an instance of a graphical game; 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 w.l.o.g. 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 equaling 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 $\mathcal{C} \in St(X_i)$. Therefore, when the treewidth is w, table A_i contains at most $\alpha^{|X_i|} \leq \alpha^{w+1}$ values. Given configuration $\mathcal{C} \in St(X_i)$, the table value $A_i(\mathcal{C})$ corresponds to the (maximum) number of players in best-response in G_i w.r.t. $\mathcal{C}$. Thus, $A_i(\mathcal{C}) = |V_i|$ if and only if $\exists \mathcal{C}' \in St(V_i)$ such that $\mathcal{C}' \supseteq \mathcal{C}$ and $\forall p \in V_i, \mathcal{C}'_p \in \beta_p(\mathcal{C}'_{-p})$. 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.

In addition, we have a 0, 1-table F_p for each $p \in V$ that has the same number of entries as matrix M_p. Initially, F_p has the value 1 at all entries. For the sake of simplicity, we will assume that $|F_p| = |M_p|$ (same description size) for all $p \in V$. Essentially, F_p is where we mark which joint strategies are allowed at PNE over the neighborhood of p, with respect to the neighbors of p that have been forgotten (through a forget node). It follows that F tables will be updated at forget nodes—when a player is forgotten, her neighbors will update their F tables. At introduce nodes the F table will be examined—when a player is introduced, her neighbors will check their F tables for joint strategies that are allowed with regards to their forgotten neighbors. Formally: Let $i \in \mathcal{T}$ be a forget node of the tree decomposition. Then, after i is examined and corresponding F tables updated, we have that: For every $p \in X_i$ and $\mathcal{C} \in St(\mathcal{N}(p) \cup \{p\})$, $F_p(\mathcal{C})$ has the value 0 if and only if $\exists u \in \mathcal{N}(p) \cap V_i$ such that $\mathcal{C}_u \notin \beta_u(\mathcal{C}_{-u})$. We repeat that the F tables are *only* modified at forget nodes and their initial state is to have the value 1 at all entries.

The algorithm presented here will use the best response function in a more generalized fashion than Definition 2: the input might be a configuration over a subset of the neighborhood of the player under consideration. Let $p \in V$, P' be subset of the neighborhood of p, $P' \subset \mathcal{N}(p)$, and $\mathcal{C}$ a configuration over the players in P', $\mathcal{C} \in St(P')$. In this case, $\beta_p(\mathcal{C})$ contains $a_p \in St(p)$ if and only if $\exists \mathcal{C}' \in St(\mathcal{N}(p))$ such that $\mathcal{C}' \supset \mathcal{C}$ and $a_p \in \beta_p(\mathcal{C}')$. Similarly, let $\mathcal{C} \in St(P' \cup \{p\})$. Then, by $F_p(\mathcal{C})$ we mean all entries $F_p(\mathcal{C}')$ such that $\mathcal{C}' \supset \mathcal{C}$. Also, if the configuration given as input includes strategies for players that are not in $\mathcal{N}(p)$, these strategies are ignored. First we briefly argue about the complexity of the best response function and then we provide a case analysis based on the type of the node under examination.

The complexity of the best response function. Let $\mathcal{G} = (G, \mathcal{M})$ be a graphical game and $p \in V$ a player of the game. In addition, let $P' = \mathcal{N}(p) \subseteq V$ and $\mathcal{C} \in St(P')$. To compute $\beta_p(\mathcal{C})$ we need to perform $|St(p)|$ steps; that is, to examine each row of the column corresponding to $\mathcal{C}$ in order to find the strategies of p that constitute the best responses to $\mathcal{C}$. Now, let $\mathcal{C}' \subset \mathcal{C}$, so that $\mathcal{C}'$ is a configuration for only a subset of the neighbors of p. In this case $\beta_p(\mathcal{C})$ contains $a_p \in St(p)$ if and only if $\exists \mathcal{C}' \in St(\mathcal{N}(p))$ such that $a_p \in \beta_p(\mathcal{C}')$. The worst case is when $\mathcal{C}'$ contains information only for one player v (thus, $\mathcal{C}' = \mathcal{C}'_v$). Then, computing $\beta_p(\mathcal{C}')$ takes $\frac{1}{|St(v)|} \cdot |M_p|$ computational steps, since $|St(\mathcal{N}(p))| = |St(v)| \cdot |St(\mathcal{N}(p)/\{v\})|$.

5.1.1 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 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 $\mathcal{C}$ over the vertices of X_i (in this case $St(X_i) = St(p)$) we set $A_i(\mathcal{C}) = 1$.

5.1.2 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 1. 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 1 *Let* $\mathcal{C} \in St(X_i)$*,* $A_i(\mathcal{C}) = \max_{a_p \in St(p)} A_j(\mathcal{C} \cup \{a_p\})$.

In the case of a forget node we additionally have to update the F_u table for each $u \in X_i \cap \mathcal{N}(p)$. While computing the maximizing values for the procedure suggested by Lemma 1 we encounter all the possible combinations of joint strategies over the players in X_j (the bag including p). For each $\mathcal{C} \in St(X_i)$ and $a_p \in St(p)$, if $a_p \notin \beta_p(\mathcal{C})$ then we set $F_u(\mathcal{C} \cup \{a_p\}) = 0$. Information about the preferences of forgotten players propagates this way.

For each $u \in X_j$ we read the table M_p (to conclude if $a_p \notin \beta_p(\mathcal{C})$) and table F_u once (to update it). Since X_i is a forget node we have $|X_i| \le w$. Thus, the time needed to compute the values of the table A_i and to update the tables F is $O(\alpha^w \cdot (|M_p| + \sum_{u \in \mathcal{N}(p) \cap X_j} |M_u|))$.

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

In the case above, p is not connected to any vertex in G_i. 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 1 which proceeds in the following manner: Given $\mathcal{C} \in St(X_j)$ and a best response for $p \in X_i - X_j, a_p \in \beta_p(\mathcal{C})$, for each player $u \in X_j \cap \mathcal{N}(p)$ it checks if u is in best-response with respect to configuration $\mathcal{C} \cup \{a_p\}$. If $\mathcal{C}_u \in \beta_u(\mathcal{C} \cup \{a_p\})$ it also checks that $F_u(\mathcal{C} \cup \{a_p\}) = 1$ and thus that the suggested joint strategy $\mathcal{C} \cup \{a_p\}$ is allowed from the perspective of u with regards to her forgotten neighbors. 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 best-response with respect to the current configuration. Hence, for $A_i(\mathcal{C} \cup \{a_p\})$ we take the value $A_j(\mathcal{C}) + 1$.

Algorithm 1 IntroNode

```
Input: $\mathcal{C} \in St(X_j)$, $p \in X_i - X_j$, $a_p \in \beta_p(\mathcal{C})$
Output: $A_i(\mathcal{C} \cup \{a_v\})$
 1: Initiate set $P_i = \emptyset$
 2: for $u \in \mathcal{N}(p) \cap X_j$ do
 3:     if $C_u \in \beta_u(\mathcal{C} \cup \{a_p\})$ and $F_u(\mathcal{C} \cup \{a_p\}) = 1$ 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:     $A_i(\mathcal{C} \cup \{a_p\}) \leftarrow A_j(\mathcal{C}) + 1$
 9: else
10:     $A_i(\mathcal{C} \cup \{a_p\}) \leftarrow A_j(\mathcal{C})$
11: end if
```

Lemma 3 *Given an introduce node $i \in \mathcal{T}$ with child $j \in \mathcal{T}$ such that $p \in X_i - X_j$, we can compute $A_i(\mathcal{C})$ for all configurations $\mathcal{C} \in St(X_i)$ in time $O(\alpha^{|X_i|} \cdot (|M_p| + \sum_{u \in \mathcal{N}(p) \cap X_j} |M_u|))$.*

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 1 for each configuration $\mathcal{C} \in St(X_j)$. Computing $\beta_p(\mathcal{C})$ takes at most $|M_p|$ steps. The loop at lines 2–6 is through all vertices $u \in \mathcal{N}(p) \cap X_j$ and for each vertex u, $\beta_u(\mathcal{C} \cup \{a_p\})$ is computed once and table F_u is checked once at line 3 in at most $2 \cdot |M_u|$ steps. The operation at line 7 takes one step. For each of the $\alpha^{|X_j|} = \alpha^{|X_i|-1}$ configurations Algorithm 1 has to be run at most α times (for each $a_p \in St(p)$). The lemma follows. □

The algorithm and lemma above show us how to compute the A_i 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 $|X_i| - 1$ matrices are read once for each configuration. Note that adjacency is only checked once since it does not change for different entries.

5.1.4 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}. What we need to capture here, is that a configuration $\mathcal{C}$ 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 $\mathcal{C}$ the computation of $A_i(\mathcal{C})$ for a join node takes only constant time as described by Lemma 4. Therefore, the computation of the whole table for a join node takes place in time $O(\alpha^{|X_i|})$.

Lemma 4 *Let $\mathcal{C} \in St(X_i)$, $A_i(\mathcal{C}) = A_{j_1}(\mathcal{C}) + A_{j_2}(\mathcal{C}) - |X_i|$.*

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. When $v \in V_{j_1}$, $w \in V_{j_2}$ and $v, w \notin X_i$, then $\{v, w\} \notin E$ [5]. 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. □

5.2 Combining the Tables

The algorithm proposed in the previous section is a bottom–up tree walk that finds partial configurations for each bag of the tree decomposition, that could be part of a 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 5 *Graphical game $(G, \mathcal{M})$ has a PNE if and only if $\exists \mathcal{C} \in St(X_r)$ such that $A_r(\mathcal{C}) = |V|$.*

In addition, note that during the bottom–up tree walk, if there exists a bag X_i such that $\forall \mathcal{C} \in St(X_i)$: $A_i(\mathcal{C}) < |V_i|$, 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. Upper time bounds of the algorithm suggested in this section are provided by the theorem below.

Theorem 3 *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 $O(\alpha^w \cdot n \cdot |\mathcal{M}|)$ time.*

Proof As discussed above, the computationally expensive nodes are the forget and introduce nodes. Both types have asymptotically the same upper bound. Thus, here we assume that every node $i \in \mathcal{T}$ is an introduce node[2] with child j and vertex $p \in X_i - X_j$. We derive the following upper bound:

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

The summation over all matrices $|M_p|$ gives $|\mathcal{M}|$. In addition, the second summation is over at most $n - 1$ elements which in turn are upper bounded by $|\mathcal{M}|$ because of the first summation. The theorem follows. □

Our algorithm improves significantly on the previous known bounds, since the base of the exponent is only the number of available strategies and not the whole

[2] Observe that in the case of a clique all the nodes of $\mathcal{T}$ but one are introduce nodes. The treewidth of an n-clique is $n - 1$.

game description. For example, if we assume that the number of available strategies is bounded by a constant, our algorithm becomes fixed-parameter tractable with respect to treewidth.

5.3 $O(\log n)$ Treewidth

Let us consider the implications that our algorithm has for games where the input graph has logarithmically-bounded treewidth. Daskalakis and Papadimitriou [8] 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. Our algorithm improves on their results in the following ways: First, it is polynomial for graphical games of $O(\log n)$ treewidth and bounded number of strategies, even without the bounded neighborhood size assumption. Second, if the size of the neighborhood is bounded we achieve an upper bound that is polynomial[3] in n. Our bound improves on the time complexity of the algorithms presented in [8] by removing $\Delta = \max_{p \in V} |\mathcal{N}(p)|$ from the exponent.

Corollary 1 *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. Moreover, if the size of the neighborhood is bounded the algorithm becomes polynomial in the number of players.*

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 [8]. 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

$$\begin{aligned} &\alpha^{3.67 \cdot c \cdot \log n} \cdot 3.67 \cdot c \cdot \log n \cdot |\mathcal{M}| \\ &= n^{3.67 \cdot c} \cdot 3.67 \cdot c \cdot \log n \cdot |\mathcal{M}| \end{aligned}$$

computational steps, which is $poly(|\mathcal{M}|)$. Since $|\mathcal{M}| \leq n \cdot |M_\Delta| = n \cdot \alpha^{\Delta}$, if the size of the neighborhood is bounded we have that the above time bound is $poly(n)$. □

5.4 Final Remarks

We give some further comments on the results we have obtained in this section. First, we consider our assumption—present in Sect. 5.1—that the tree decomposition is part

[3] Note that if the degree of the graph is bounded, then the description of the graphical game is polynomial in the number of players.

of the input. This assumption is justified by virtue of the celebrated treewidth algorithm of Bodlaender [2]. This algorithm computes a tree decomposition of a given graph in fixed-parameter tractable time, where the parameter is the treewidth of the graph. However, it is known to be of little practical use due to very large constants in the running time. Recently, Bodlaender et al. gave the first algorithm providing a constant approximation of treewidth which runs in time $2^{O(w)}n$, where w is the treewidth of the given graph and n the number of vertices [4]. Given a graph G and an integer k, this algorithm either outputs that the treewidth of G is larger than k or gives a tree decomposition of width at most $5k + 4$. Therefore, if we combine this algorithm with ours, the asymptotic running time of the latter does not change.

An immediate consequence of Theorem 3 is that PNE-GG is in FPT, for the combined parameter (α, w). It follows that (α, w)-PNE-GG is kernelizable (cf. Sect. 2.3) and it is natural to ask whether (α, w)-PNE-GG admits a *polynomial* kernel. A polynomial kernel means that given an instance of (α, w)-PNE-GG, we can obtain, in time polynomial in the size of the instance and the value of the parameter, an equivalent instance whose size and parameter value are both polynomially bounded in (α, w). We sketch how the method of AND-Distillation, described in [3], can be used to prove the non-existence of a polynomial kernel for our problem. In [3], AND-Distillation was formulated as a conjecture, but recently, Drucker proved that AND-Distillation holds unless $NP \subseteq coNP/poly$ [11]. It is easy to prove that (α, w)-PNE-GG is AND-Compositional by taking the disjoint union of m instances of the problem. Note that each instance is assumed to have a constant number of available strategies for every player (otherwise the problem is $W[1]$-hard, cf. Sect. 4). The resulting AND-Composition instance has the same number of strategies and, in addition, it has treewidth w and there exists a PNE if and only if all m instances have a PNE. This implies that (α, w)-PNE-GG does not admit a polynomial kernel, unless $NP \subseteq coNP/poly$. The relevant definitions can be found in [3].

6 $O^*(\alpha^w)$ is Probably Optimal

We now prove a lower bound of the following form: one cannot improve the $O^*(\alpha^w)$ exponential dependency on treewidth for solving PNE-GG unless SETH fails. The formal statement is presented below at Theorem 4. We prove this by giving a reduction from SAT to PNE-GG, in the spirit of [27]. Our reduction is inspired from the one to q-Coloring, described in Sect. 6 of [27]. Note that PNE-GG can be seen as a generalization of q-Coloring: Given a q-Coloring instance, one could argue that it is decidable by solving a PNE-GG instance on the same graph, where all players have q strategies and their payoffs are such that they want to pick a different strategy than their neighbors (anti-coordination game). However, in such a game the payoff matrices would have size exponential in the size of the neighborhood (even if we only allow 0,1 payoffs). Thus, we cannot directly use the aforementioned reduction for q-Coloring to prove Theorem 4 below. In this section we describe a new reduction suitable for PNE-GG.

Theorem 4 *If, for some $\alpha \geq 3$ and $\epsilon > 0$, PNE-GG can be solved in time $O^*((\alpha - \epsilon)^w)$, then SAT can be solved in time $O^*((2 - \delta)^n)$, for some $\delta > 0$.*

Let ϕ be the input SAT formula with n the number of variables and m the number of clauses. Parameter p is chosen w.r.t. α and ϵ in a way to be explained later. Given a SAT formula ϕ the reduction described below outputs a graphical game $\mathcal{G} = (G, \mathcal{M})$, $G = (V, E)$, which admits a PNE if and only if ϕ is satisfiable. First we describe the construction of the graph G:

1. Group the variables of ϕ into t groups $F_1, \ldots, F_t$ of size $\lfloor \log \alpha^p \rfloor$. An assignment of values to the variables of a group F_i is called a *group assignment*. The number of groups is $t = \lceil \frac{n}{\lfloor \log \alpha^p \rfloor} \rceil$.
2. For each group F_k we have a set V_k that consists of p paths, $\mu_1, \ldots, \mu_p$. Each such path contains m vertices and from now on these paths will be called m-chains (to distinguish from the other type of path to appear in the next steps of the construction). Each vertex of a set V_k is denoted as v_k^i where $i \in [m]$ and has $St(v_k^i) = [\alpha]$. With $V_k^i \subset V_k$ we mean the set which consists of the ith vertex of each m-chain in V_k, $V_k^i = \{v_k^i | v_k^i \in V_k\}$.
   Let $\mathcal{C}_k \in St(V_k)$ be a configuration where for all m-chains of V_k, $\mu \in V_k$, we have that $\forall v, u \in \mu, \mathcal{C}_k[v] = \mathcal{C}_k[u]$ (all vertices of an m-chain play the same strategy). Such configurations are called the *valid configurations* of V_k. We have $\alpha^p \geq 2^{|F_k|}$ possible valid configurations. Each group assignment of F_k corresponds to a unique valid configuration of V_k (note that there might be some valid configurations that do not correspond to any group assignment of F_k).
3. For each clause C_i of ϕ we take a gadget $\widehat{C_i}$ that consists of a a path P_i and two extra vertices. On P_i we take one vertex for each satisfying group assignment of C_i. The total number of group assignments are $t2^{\lfloor \log \alpha^p \rfloor} \leq t\alpha^p$ which is $O(n)$ since α, p are constants independent of the input ϕ. Each vertex of P_i is denoted as w_i^k and has $St(w_i^k) = \{\mathtt{na}_1, \mathtt{na}_2, \mathtt{active}\}$ (the first two strategies can be thought of as not-active). With w_i^k we mean a vertex of P_i that corresponds to a group assignment of F_k. The corresponding configuration of V_k is denoted as $\mathcal{S}(w_i^k)$. The two extra vertices are denoted as w_i^{start} and w_i^{end}; the former is connected to the leftmost vertex and the latter to the rightmost vertex of P_i and have $\{\mathtt{na}_1, \mathtt{na}_2\}$ as their strategy set.
4. The last step is to connect vertices of paths P_i to vertices of sets V_k. For each $w_i^k \in P_i$ and $v_k^i \in V_k^i$ we take an edge in G.

An illustration of the construction can be found in Fig. 4.

Secondly, we define the payoff matrices $\mathcal{M}$. In the resulting game we only have 0, 1 payoffs which we describe in terms of a payoff function $u_v : St(V) \to \{0, 1\}$, for each player $v \in V$. Let $\mathcal{C} \in St(V)$ be a global configuration.

For every gadget $\widehat{C_i}$ we have that: $u_{w_i^{start}}(\mathcal{C}) = 1$ when w_i^{start} plays $\mathtt{na}_1$ and 0 otherwise, regardless of the strategies of her neighbors; $u_{w_i^{end}}(\mathcal{C}) = 1$ when w_i^{end} plays $\mathtt{na}_l$ and 0 for the other available strategy. Here, l is 1 if $V(P_i)$ is even and 2 otherwise. The payoffs for the vertices $w_i^k \in P_i$ are such that they play an anti-coordination game with their neighbors on $\widehat{C_i}$. By that we mean that each vertex w_i^k wants to play a strategy different from the ones the vertices in $\mathcal{N}(w_i^k) \cap P_i$ play. Then, by a parity argument there is at least one vertex of P_i that plays $\mathtt{active}$ (also, one is enough).

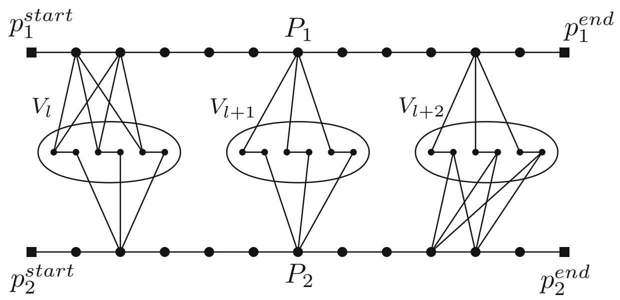


Fig. 4 Sample construction, derived from a formula with only two clauses. We depict the two paths P_1, P_2 and three sets of vertices V_l, V_{l+1}, V_{l+2} corresponding to groups F_l, F_{l+1}, F_{l+2}. We have C_1 being satisfied by two different assignments of F_l (similarly C_2 by two different assignments of F_{l+2}) while for the rest of the groups both clauses are satisfied by only one assignment

Following we describe the payoffs for vertices of a path P_i, $w_i^k \in P_i$.

- $u_{w_i^k}(\mathcal{C}) = 1$ if w_i^k plays `active`, none of the players in $\mathcal{N}(w_i^k) \cap P_i$ play `active` and vertices in $\mathcal{N}(w_i^k) \cap V_k$ play according to $\mathcal{S}(w_i^k)$;
- $u_{w_i^k}(\mathcal{C}) = 1$ if w_i^k plays $\mathtt{na}_l$ and none of the players in $\mathcal{N}(w_i^k) \cap P_i$ play $\mathtt{na}_l$;
- $u_{w_i^k}(\mathcal{C}) = 0$ in all other cases.

Intuitively, a vertex $w_i^k \in P_i$ plays `active` while anti-coordinating with her neighbors on P_i and the vertices of in V_i^k play according to the corresponding configuration $\mathcal{S}(w_i^k)$. Then, C_i is satisfied by the group assignment to which w_i^k corresponds. It remains to define the payoffs for players of sets V_k. Let $v \in V_k \cap \mu$ be a player that plays action $s \in [\alpha]$ (and lies on m-chain μ). Her payoff is defined as:

- $u_v(\mathcal{C}) = 1$ if all of players in $\mathcal{N}(v) \cap \mu$ play s.
- $u_v(\mathcal{C}) = 0$ in all other cases.

It follows, that x is a PNE only if x is a valid configuration.

Lemma 6 *The construction takes polynomial time.*

Proof The number of vertices of each path are at most $t\alpha^p$ which is $O(n)$ and the number of vertices in sets V_i are at most mp which is $O(m)$. In addition, the degrees of all the vertices are bounded: Every vertex $w_i^k \in P_i$ has degree at most $p + 2$ which is constant; every vertex $v_k \in V_k$ has degree at most $\alpha^p + 2$ which is also constant. Since the degrees are bounded by constants, the size of the matrices are also constant.[4] □

Lemma 7 *ϕ is satisfiable if and only if $\mathcal{G}$ has a PNE.*

Proof A satisfying assignment to ϕ corresponds to a group assignment for each group F_k, which in turn corresponds to a valid configuration $\mathcal{C} \in St(V_k)$. Let $\mathcal{C}' \subset \mathcal{C}$ be

[4] The actual size of the matrices are $|M_{w_i^k}| \leq 3^3\alpha^p$ and $|M_{v_k}| \leq \alpha^3 3^{\alpha^p}$.

such that $\mathcal{C}' \in St(V_k^i)$. Then, there exists a vertex $w_i^k \in P_i$, such that $\mathcal{C}' = \mathcal{S}(w_i^k)$. We let w_i^k play `active` . Note that a path P_i might have more than one vertices that play `active` if the satisfying assignment corresponds to more than one group assignments (for different groups) that satisfy C_i. The rest of the vertices on P_i play alternatingly $\mathtt{na}_1, \mathtt{na}_2$ in order to anti-coordinate. Finally, all gadgets $\widehat{C_i}$ are pairwise disjoint; therefore, we can repeat the same procedure for each of them until we end up with a global configuration $\mathcal{C} \in St(V)$. $\mathcal{C}$ is a PNE because $\forall v \in V, u_v(\mathcal{C}) = 1$ and 1 is the maximum payoff that v could receive.

For the opposite direction let $\mathcal{C} \in St(V)$ be a PNE. Then, $\mathcal{C}' \in St(V_k), \mathcal{C}' \subset \mathcal{C}$ is a valid configuration, $\forall k$. In addition, we know that there is at least one vertex $w_i^k \in P_i$ that plays `active` (because of the parity argument we discuss above) and the vertices in V_k^i play according to $\mathcal{S}(w_i^k)$—thus the vertices in V_k play $\mathcal{C}_k \in St(V_k)$, the valid configuration corresponding $\mathcal{S}(w_i^k)$. For each group F_k, we take the group assignment that corresponds to $\mathcal{C}_k$ and satisfies C_i. The resulting assignment satisfies each clause C_i and thus it satisfies ϕ. If the configuration $\mathcal{C}' \in St(V_k)$ does not correspond to a satisfying assignment for F_k then we set all the variables of F_k in $false$. This is safe as all clauses are satisfied by group assignments of other groups. □

Lemma 8 *The mixed search number of the resulting graph is at most* $pt + 1$.

Proof We describe a mixed search strategy where we use a distinguished *searcher* and a set of pt searchers that we call *blockers*. For every i we place a blocker on the first vertex of every m-chain, which are pt in total. Initially, we place the searcher at w_1^{start}. The strategy then develops as follows:

The searcher goes through all the vertices of path P_i (from w_i^{start} to w_i^{end}). At the same all the blockers are placed on the vertices of the sets $V_k^i, \forall k \in [t]$. During this all the edges of the path P_i get cleaned. Also the edges between P_i and the sets V_k get cleaned because at the time the searcher reaches a vertex with edges towards V_k the blockers are on the other side of these edges.

In the next round, the searcher moves to the next path P_{i+1} and the blockers move accordingly to the vertices of sets V_k^{i+1}. During this move no edge of the graph gets recontaminated because there is no path to a contaminated edge (any such path is blocked by a blocker). So after m rounds the whole graph will be cleaned: the searcher will clean all the paths, the blockers will clean all the m-chains and the combination of the two will clean the edges in between. □

Lemma 9 *If* α*-PNE-GG can be solved in* $O^*((\alpha - \epsilon)^w)$ *for some* $0 < \epsilon < 1$*, then SAT can solved in* $O^*((2 - \delta)^n)$ *for some* $0 < \delta < 1$.

Proof We have $O^*((\alpha - \epsilon)^w) = O^*(\alpha^{\lambda w})$, where $\lambda = \log_\alpha (\alpha - \epsilon) < 1$. Next, we describe how we choose the parameter p. We need p be to be sufficiently large such that $\frac{p+1}{p-1}\lambda = \delta' < 1$. Based on the construction described above, given a SAT formula ϕ we construct a graphical game $\mathcal{G}$. According to Lemma 7, we can decide satisfiability of ϕ by deciding the existence of a PNE for $\mathcal{G}$. W.r.t. Lemma 8 the treewidth of G is at most $pt + 2$. Thus, we can solve $\mathcal{G}$ in time $O^*(\alpha^{\lambda(pt+2)}) = O^*(\alpha^{\lambda pt})$. We have that:

$$\lambda pt = \lambda p \lceil \frac{n}{\lfloor p \log \alpha \rfloor} \rceil \leq \lambda(p+1) \frac{n}{(p-1)\log \alpha} = \delta' \frac{n}{\log \alpha} = \delta'' n$$

for some $\delta'' < 1$. Therefore, we can solve the SAT formula (by solving $\mathcal{G}$) in time $O^*(2^{\delta'' n})$ for some $\delta'' < 1$, which is equal to $O^*((2-\delta)^n)$ for for some $\delta > 0$. □

This concludes the proof of Theorem 4.

7 Constructing a PNE

In this section we discuss the construction of 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 (similar to the one used in [8]).

Definition 8 (Succinct Description) 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 PNE since $|\mathcal{A}|$ is polynomial to $\mathcal{G}$ and the set of all PNE can be computed in time polynomial in $|\mathcal{G}| + |\mathcal{A}|$. Subsequently, we prove that constructing a sample or the maximum-payoff PNE does not require additional computational effort.

Theorem 5 *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* $O(\alpha^w \cdot n \cdot |\mathcal{M}|)$ *time. Moreover, the same algorithm computes a succinct description of all PNE.*

Proof First we execute the algorithm of Sect. 5, which will either output NO or give us the tables $A_i, \forall i \in \mathcal{T}$. To construct a *sample* PNE 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 $\mathcal{C} \in St(X_r)$ such that $A_r(\mathcal{C}) = |V|$. For each player $p \in X_r$ we set $S_p = \mathcal{C}_p$. 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 $\mathcal{C} \in St(X_j)$ such that $\forall u \in X_i, \mathcal{C}_u = S_u$ and $A_j(\mathcal{C}) = A_i(\mathcal{C} \backslash \{\mathcal{C}_p\})$ (remember that $G_i = G_j$) and set $S_p = \mathcal{C}_p$. 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+1} possible configurations. The procedure at the children of forget nodes does not increase the time complexity: the configuration $\mathcal{C} \in St(X_j)$ that intersects with S and contains additionally strategy a_p such that $A_j(\mathcal{C}) = A_i(\mathcal{C}\backslash\{\mathcal{C}_p\})$ 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 top–down iteration takes at most $O(n)$ steps. □

Note that for the above computations no information from the payoff matrices is read. However, this cannot be the case if our interest lies in specific PNE. The following is a corollary of Theorems 3 and 5: Given game $\mathcal{G}$ and integer k, there is an algorithm that can decide the existence and construct a *maximum-payoff* PNE configuration $\mathcal{C}$ (i.e. such that $M_p(\mathcal{C}) \geq k, \forall p \in V$). In order to be able to do this we need to slightly modify the algorithm of Sect. 5. Previously, the algorithm allowed $A_i(\mathcal{C}) = V_i$ if and only if $\mathcal{C}$ was a PNE for $V_i \subseteq V$. Now we want it to allow $A_i(\mathcal{C}) = V_i$ if and only if $\mathcal{C}$ is a PNE for $V_i \subseteq V$ and $M_p(\mathcal{C}) \geq k, \forall p \in V_i$. Otherwise, it would not increase the value of $A_i(\mathcal{C})$. This alteration is necessary and sufficient to decide the existence of a maximum-payoff PNE. Then, with the altered A_i tables we could use the same top–down procedure as in the proof of Theorem 5 to construct a maximum-payoff PNE, given that one exists.

8 Conclusions

This article deals with the problem of deciding the existence of a pure Nash equilibrium for a graphical game. The methods we use are native to the field of parameterized complexity and our main contributions sum up to the following: PNE-GG is $W[1]$-hard when parameterized by treewidth; there is a $O^*(\alpha^w)$ algorithm that solves PNE-GG (in addition, we prove that PNE-GG is polynomial-time solvable for $\alpha = 2$); there is no $O^*((\alpha - \epsilon)^w)$ time algorithm, for some $\alpha \geq 3$ and $\epsilon > 0$, that solves PNE-GG unless the SETH fails. The previous algorithms for the problem, found in [17] and [8], are based on reduction to other problems. Our algorithm is based directly on the combinatorics of the problem and improves on the time bound of [8], which was previously the fastest algorithm, by removing a factor of $\Delta = \max_{p\in V} |\mathcal{N}(p)|$ from the exponent. This is an argument in favor of the power of the techniques used in parameterized complexity.

As an epilogue we outline a few open problems. Consider the following generalization of PNE-GG: given a graphical game $\mathcal{G}$ find the largest subset of players that are in PNE. Of course, if $\mathcal{G}$ admits a PNE then it is the set of all players. It this problem solvable in $O^*(\alpha^w)$ time? A result towards the opposite would be that this problem is hard for one of the classes of the W-hierarchy, even if α is bounded. In addition, it is not clear if our algorithms are capable of computing a social welfare optimizing PNE. That is, a PNE configuration $\mathcal{C}$ that maximizes $\sum_{p\in V} M_p(\mathcal{C})$. Finally, would the fact that w-PNE-GG is in FPT for fixed α be useful for proving FPT results for other problems by reducing those to PNE-GG (or to the maximum payoff version of PNE-GG)? Note that when designing such reductions one has to be very careful with regards to the matrix sizes and, of course, that α remains constant.

References

1. Becker, A., Geiger, D.: A sufficiently fast algorithm for finding close to optimal clique trees. Artif. Intell. **125**(1–2), 3–17 (2001)
2. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. **25**(6), 1305–1317 (1996)
3. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. J. Comput. Syst. Sci. **75**(8), 423–434 (2009)
4. Bodlaender, H.L., Drange, P.G., Dregi, M.S., Fomin, F.V., Lokshtanov, D., Pilipczuk, M.: An O($c^k n$) 5-approximation algorithm for treewidth. In: Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013). IEEE Computer Society, pp. 499–508 (2013)
5. Bodlaender, H.L., Koster, A.M.C.A.: Combinatorial optimization on graphs of bounded treewidth. Comput. J. **51**(3), 255–269 (2008)
6. Chen, X., Deng, X., Teng, S.-H.: Settling the complexity of computing two-player Nash equilibria. J. ACM **56**(3), 14:1–14:57 (2009)
7. Daskalakis, C., Goldberg, P.W., Papadimitriou, C.H.: The complexity of computing a Nash equilibrium. SIAM J. Comput. **39**(1), 195–259 (2009)
8. Daskalakis, C., Papadimitriou, C.H.: Computing pure Nash equilibria in graphical games via Markov random fields. In: Feigenbaum, J., Chuang, J.C.-I., Pennock, D.M. (eds.) ACM Conference on Electronic Commerce. ACM, pp. 91–99 (2006)
9. Daskalakis, C., Schoenebeck, G., Valiant, G., Valiant, P.: On the complexity of Nash equilibria of action-graph games. In: Mathieu, C. (ed.) Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009). SIAM, pp. 710–719(2009)
10. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Springer, London (2013)
11. Drucker, A.: New limits to classical and quantum instance compression. In: Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS 2012). IEEE Computer Society, pp. 609–618 (2012)
12. Estivill-Castro, V., Parsa, M.: Computing Nash equilibria gets harder: new results show hardness even for parameterized complexity. In Downey, R., Manyem, P. (eds.) Proceedings of the 15th Computing: The Australasian Theory Symposium (CATS 2009) volume 94 of Conferences in Research and Practice in Information Technology (CRPIT). Australian Computer Society, pp. 81–87 (2009)
13. Estivill-Castro, V., Parsa, M.: Single parameter FPT-algorithms for non-trivial games. In: Iliopoulos, C.S., Smyth, W.F. (eds.) Combinatorial Algorithms - 21st International Workshop (IWOCA 2010), Lecture Notes in Computer Science, vol. 6460, pp. 121–124, Springer, Berlin (2010)
14. Fellows, M.R., Fomin, F.V., Lokshtanov, D., Rosamond, F.A., Saurabh, S., Szeider, S., Thomassen, C.: On the complexity of some colorful problems parameterized by treewidth. Inf. Comput. **209**(2), 143–153 (2011)
15. Fellows, M.R., Hermelin, D., Rosamond, F.A., Vialette, S.: On the parameterized complexity of multiple-interval graph problems. Theoret. Comput. Sci. **410**(1), 53–61 (2009)
16. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, New York (2006)
17. Gottlob, G., Greco, G., Scarcello, F.: Pure Nash equilibria: hard and easy games. J. Artif. Intell. Res. **24**, 357–406 (2005)
18. Greco, G., Scarcello, F.: On the complexity of constrained Nash equilibria in graphical games. Theorert. Comput. Sci. **410**(38–40), 3901–3924 (2009)
19. Hermelin, D., Huang, C.-C., Kratsch, S., Wahlström, M.: Parameterized two-player Nash equilibrium. Algorithmica **65**(4), 802–816 (2013)
20. Impagliazzo, R., Paturi, R.: On the complexity of k-SAT. J. Comput. Syst. Sci. **62**(2), 367–375 (2001)
21. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. Syst. Sci. **63**(4), 512–530 (2001)
22. Jiang, A.X., Leyton-Brown, K.: Computing pure Nash equilibria in symmetric action graph games. In: Proceedings of the 22nd Conference on Artificial Intelligence (AAAI 2007). AAAI Press, Menlo Park, pp. 79–85 (2007)
23. Jiang, A.X., Leyton-Brown, K., Bhat, N.A.: Action-graph games. Games Econ. Behav. **71**(1), 141–173 (2011). Special Issue In Honor of John Nash
24. Jiang, A.X., Safari, M.: Pure Nash equilibria: Complete characterization of hard and easy graphical games. In: van der Hoek, W., Kaminka, G.A., Lespérance, Y., Luck, M., Sen, S. (eds.) Proceedings

of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010) International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), pp. 199–206 (2010)

25. Kearns, M.J., Littman, M.L., Singh, S.P.: Graphical models for game theory. In: Breese, J.S., Koller, D. (eds.) Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence (UAI 2001). Morgan Kaufmann, San Francisco, CA, pp. 253–260 (2001)
26. Kratsch, S.: Recent developments in kernelization: a survey. Bull. Eur. Assoc. Theoret. Comput. Sci. (EATCS) **113**, 58–97 (2014)
27. Lokshtanov, D., Marx, D., Saurabh, S.: Known algorithms on graphs on bounded treewidth are probably optimal. In: Randall, D. (ed.) Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011). SIAM, pp. 777–789 (2011)
28. Marx, D.: Can you beat treewidth? Theory Comput. **6**(1), 85–112 (2010)
29. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press, Oxford (2006)
30. Papadimitriou, C.H., Roughgarden, T.: Computing correlated equilibria in multi-player games. J. ACM **55**(14), 1–29 (2008)
31. Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. J. Algorithms **7**(3), 309–322 (1986)
32. Schrijver, A.: Disjoint homotopic paths and trees in a planar graph. Discret. Comput. Geom. **6**(1), 527–574 (1991)
33. Schutt, A., Stuckey, P.J.: Incremental satisfiability and implication for UTVPI constraints. INFORMS J. Comput. **22**(4), 514–527 (2010)
34. Takahashi, A., Ueno, S., Kajitani, Y.: Mixed searching and proper-path-width. Theoret. Comput. Sci. **137**(2), 253–268 (1995)
35. Thomas, A., van Leeuwen, J.: Treewidth and pure Nash equilibria. In: Gutin, G., Szeider, S. (eds.) Proceedings 8th International Symposium on Parameterized and Exact Computation (IPEC 2013) LNCS, vol. 8246, pp. 348–360. Springer, Berlin (2013)