The discrete strategy improvement algorithm for parity games and complexity measures for directed graphs

Felix Canavoi Erich Grädel Roman Rabinovich
RWTH Aachen University
Germany
{canavoi,graedel,rabinovich}@logic.rwth-aachen.de

For some time the discrete strategy improvement algorithm due to Jurdziński and Vöge had been considered as a candidate for solving parity games in polynomial time. However, it has recently been proved by Oliver Friedmann that the strategy improvement algorithm requires super-polynomially many iteration steps, for all popular local improvements rules, including switch-all (also with Fearnley's snare memorisation), switch-best, random-facet, random-edge, switch-half, least-recently-considered, and Zadeh's Pivoting rule.

We analyse the examples provided by Friedmann in terms of complexity measures for directed graphs such as treewidth, DAG-width, Kelly-width, entanglement, directed pathwidth, and cliquewidth. It is known that for every class of parity games on which one of these parameters is bounded, the winning regions can be efficiently computed. It turns out that with respect to almost all of these measures, the complexity of Friedmann's counterexamples is bounded, and indeed in most cases by very small numbers. This analysis strengthens in some sense Friedmann's results and shows that the discrete strategy improvement algorithm is even more limited than one might have thought. Not only does it require super-polynomial running time in the general case, where the problem of polynomial-time solvability is open, it even has super-polynomial lower time bounds on natural classes of parity games on which efficient algorithms are known.

1 Introduction

Parity games are a family of infinite two-player games on directed graphs. They are important for several reasons. Many classes of games arising in practical applications admit reductions to parity games (over larger game graphs). This is the case for games modelling reactive systems, with winning conditions specified in some temporal logic or in monadic second-order logic over infinite paths (S1S), for Muller games, but also for games with partial information appearing in the synthesis of distributed controllers. Further, parity games arise as the model-checking games for *fixed-point logics* such as the modal μ-calculus or LFP, the extension of first-order logic by least and greatest fixed-points. Conversely, winning regions of parity games (with a bounded number of priorities) are definable in both LFP and the μ-calculus. Parity games are also of crucial importance in the analysis of structural properties of fixed-point logics.

From an algorithmic point of view parity games are highly intriguing as well. It is an immediate consequence of the *positional determinacy* of parity games, that their winning regions can be decided in NP $\cap$ Co-NP. In fact, it was proved in [11] that the problem is in UP $\cap$ Co-UP, where UP denotes the class of NP-problems with unique witnesses. The best known deterministic algorithm has complexity $n^{O(\sqrt{n})}$ [13]. For parity games with a number d of priorities the progress measure lifting algorithm by Jurdziński [12] computes winning regions in time $O(dm \cdot (2n/(d/2))^{d/2}) = O(n^{d/2+O(1)})$, where m is the number of edges, giving a polynomial-time algorithm when d is bounded. The two approaches can be

M. Faella, A. Murano (Eds.): Games, Automata, Logics
and Formal Verification (GandALF 2012)
EPTCS 96, 2012, pp. 197–209, doi:10.4204/EPTCS.96.15

combined to achieve a worst-case running time of $O(n^{d/3+O(1)})$ for solving parity games with d priorities, with $d = \sqrt{n}$ (see [1, Chapter 3].

Although the question whether parity games are in general solvable in PTIME is still open, there are efficient algorithms that solve parity games in special cases, where the structural complexity of the underlying directed graphs, measured by numerical graph parameters, is low. These include parity games of bounded treewidth [15], bounded entanglement [3, 4], bounded DAG-width [2], bounded Kelly-width [10], or bounded cliquewidth [16].

One algorithm that, for a long time, had been considered as a candidate for solving parity games in polynomial time is the *discrete strategy improvement algorithm* by Jurdziński and Vöge [14]. The basic idea behind the algorithm is to take an arbitrary initial strategy for Player 0 and improve it step by step until an optimal strategy is found. The algorithm is parametrized by an *improvement rule*. Indeed, there are many possibilities to improve the current strategy at any iteration step, and the improvement rule determines the choice that is made. Popular improvement rules are switch-all, switch-best, random-facet, random-edge, switch-half and Zadeh's Pivoting rule. Although it is open whether there is an improvement rule that results in a polynomial worst-case runtime of the strategy improvement algorithm, Friedmann [9] was able to show that there are super-polynomial lower bounds for all popular improvement rules mentioned above. For each of these rules, Friedmann constructed a family of parity games on which the strategy improvement algorithm requires super-polynomial running time.

In this paper we analyse the examples provided by Friedmann in terms of complexity measures for directed graphs. It turns out that with respect to most of these measures, the complexity of Friedmann's counterexamples is bounded, and indeed in most cases by very small numbers. This analysis strengthens in some sense Friedmann's results and shows that the discrete strategy improvement algorithm is even more limited than one might have thought. Not only does it require super-polynomial running time in the general case, where the problem of polynomial-time solvability is open, it even has super-polynomial lower time bounds on natural classes of parity games on which efficient algorithms are known.

2 The strategy improvement algorithm

We assume that the reader is familiar with basic notions and terminology on parity games. We shall now briefly discuss the discrete strategy improvement algorithm and the different improvement rules that parametrize it. For the purpose of this paper a precise understanding of the algorithm is not needed. The idea of the strategy improvement algorithm is that one can compute an optimal strategy of a player by starting with an arbitrary intitial strategy and improve it step by step, depending on a discrete valuation of plays and strategies, and on a rule that governs the choices of local changes (switches) of the current strategy.

It is well-known that parity games are determined by positional strategies, i.e. strategies which at each position just select one of the outgoing edges, independent of the history of a play. The discrete valuation defined by Jurdziński and Vöge [14] measures how good a play is for Player 0 in a more refined way than just winning or losing. Given a current strategy one can then, at each position of Player 0, consider the possible local changes, i.e. the switches of the outgoing edges, and select a locally best posibility. Rules that describe how to combine such switches in one improvement step are called *improvement rules* and parametrise the algorithm.

The *switch-all* or *locally optimizing* rule [14] regards each vertex independently and performs the best possible switch for every vertex. In other words, for every vertex, it computes the best improvement of the strategy *at that vertex* assuming that the strategy remains unchanged at other vertices. However, the

switch is done simultaneously at each vertex. The *switch-best* or the *globally optimising rule* takes cross-effects of improving switches into account and applies in every iteration step a best possible combination of switches.

The *random-edge* rule applies a single improving switch at some vertex chosen randomly and the improvement rule *switch-half* applies an improving switch at every vertex with probability $1/2$. The *random-facet* rule chooses randomly an edge e leaving a Player 0 vertex and computes recursively a winning strategy σ^* on the graph without e. If taking e is not an improvement, σ^* is optimal, otherwise σ^* switched to e is the new initial strategy. The *least-entered* rule switches at a vertex at that the least number of switches has been performed so far. Cunningham's *least-recently-considered* or *round-robin* rule fixes an initial ordering on all Player 0 vertices first, and then selects the next vertex to switch at in a round-robin manner. Fearnley introduced in [6] *snare memorisation*. It can be seen as an extension of a basic improvement rule by a snare rule that memorises certain structures of a game to avoid reoccurring patterns.

All local improvement rules discussed here can be computed in polynomial time [14, 17]. Hence the running time of the algorithm on a game depends primarily on the number of improvement steps. In a series of papers and in his dissertation, Friedmann has constructed, for each of the above mentioned improvement rules, a class of parity games on which the strategy improvement algorithm requires super-polynomially many iteration steps, with respect to the size of the game [9, 8, 7]. We shall analyse these games in terms of certain complexity measures for directed graphs which we describe in the next section.

3 Complexity measures for directed graphs

For most of the complexity measures we shall work with a characterisation in terms of so-called *graph searching games*, that allow us a more intuitive point of view on the measures and give us an easier way to analyse the graphs in question. A graph searching game is played on a graph by a team of cops and a robber. In any position, the robber is on a vertex of the graph and each cop either also occupies a vertex or is outside of the graph. The robber can move between vertices along cop-free paths in the graph, i.e. paths whose vertices are not occupied by cops. The moves of the cops have typically no restrictions. The aim of the cops is to capture the robber, i.e. to force him in a position where he has no legal moves. Precise rules of moves characterise a complexity measure of the graph. The value of the measure is the minimal number of cops needed to capture the robber (-1 in some cases). Hence, on simple graphs, few cops suffice to win wheras complex graphs demand many cops to capture the robber.

In that way, treewidth, DAG-width, Kelly-width, directed pathwidth and entanglement can be described. Another measure that we shall consider is cliquewidth, for which no characterisation by games is known. Recall that common definitions of such measures are usually given by means of appropriate decompositions of the graph into small parts that are connected in a simple way: as a directed path for directed pathwidth, as a tree for treewidth, as a DAG for DAG-width and Kelly-width, or as a parse tree for cliquewidth. The maximal size of a part in a decomposition corresponds to the minimal number of cops needed to capture the robber on the graph (except for entanglement, for which no corresponding decomposition is known). Such decompositions can be used to provide efficient algorithms for problems that are difficult (e.g. NP-complete) in general, on graph classes where the values of the respective measure is bounded. In particular, this is the case for parity games. In a series of papers it has been shown that parity games can be solved in PTIME on graph classes with bounded treewidth, directed pathwidth, DAG-width, Kelly-width, entanglement or cliquewidth. In the following, we define all complexity measures discussed above by their characterisations in terms of graph searching games except cliquewidth,

for which we give an inductive definition.

Treewidth. Treewidth is a classical measure of cyclicity on undirected graphs. It measures how close a graph is to being a tree. The treewidth game $\mathrm{twG}_k(\mathcal{G})$ is played on an undirected graph $\mathcal{G} = (V, E)$ by a team of k cops and a robber, whereby k is a parameter of the game. Initially, there are no cops on the graph and the robber chooses an arbitrary vertex and occupies it in the first move. The players move alternating. A cop position is a tuple (C, v) where $C \subseteq V$ with $|C| \leq k$ is the set of vertices occupied by cops (if $k > |C|$, the remaining cops are considered to be outside of $\mathcal{G}$) and $v \notin C$ is the vertex occupied by the robber. The cops can move to a position (C, C', v) with $|C'| \leq k$. Intuitively, they announce their next placement C' and take cops from $C' \setminus C$ away from $\mathcal{G}$. The robber positions are of the form (C, C', v). The robber can run along paths on the graph whose vertices are not occupied by cops, i.e. the next (cop) position may be (C', w) where $w \in \mathrm{Reach}_{\mathcal{G}-(C\cap C')}(v) \setminus C'$, i.e. w is reachable from v in $\mathcal{G} - (C \cap C')$. Thus the cops are placed on the vertices they announced in their previous move; furthermore, only those cops prevent the robber to run who are both in the previous and in the next placements. However, the robber is not permitted to go to a vertex which will be occupied by a cop in the next position.

The robber is captured in a position (C, C', v) if he has no legal move: all neighbours of v are in $C \cap C'$ and a cop is about to occupy his vertex, i.e. $v \in C'$. A play is monotone if the robber can never reach a vertex that has already been unavailable for him. It suffices to demand that in any move, the robber is not able to reach a vertex that has just been left by a cop. Formally, a play is monotone if, for every cop move $(C, v) \to (C, C', v)$ in the play, we have $\mathrm{Reach}_{\mathcal{G}-(C\cap C')}(v) \cap (C \setminus C') = \emptyset$.

The cops win a monotone play if it ends in a position in that the robber is captured. Infinite or non-monotone plays are won by the robber. A (positional) strategy for the cops is a function $\sigma\colon 2^V \times V \to 2^V$ which prescribes, for every cop position (C, v), the next placement $\sigma(C, v)$. Similarly, a (positional) strategy for the robber is a function $\rho\colon 2^V \times 2^V \to V$ which maps a robber position (C, C', v) to a cop position (C', w) with $w \in \mathrm{Reach}_{\mathcal{G}-(C\cap C')}(v)$. A play π is consistent with a strategy σ (or ρ) if every move in the play is made according to σ (or ρ). A strategy for a player is winning if he wins every play consistent with that strategy. As the winning conditions for both players are Boolean combinations of reachability and safety, it suffices to consider only positional strategies.

The minimal number k such that the cops have a winning strategy in $\mathrm{twG}_{k+1}(\mathcal{G})$ is the *treewidth* $\mathrm{tw}(\mathcal{G})$ of $\mathcal{G}$. If $\mathcal{G} = (V, E)$ is a directed graph then $\mathrm{tw}(\mathcal{G})$ is $\mathrm{tw}(\overline{\mathcal{G}})$ where $\overline{\mathcal{G}} = (V, \overline{E})$ and $\overline{E}$ is the symmetric closure of E.

DAG-width. The DAG-width game $\mathrm{dagwG}_k(\mathcal{G})$ [2] is played on a directed graph $\mathcal{G}$ in the same way as the treewidth game, but the edge relation of the graph is not symmetrised. Note that the meaning of the reachability relation Reach on directed graphs is, of course, different from the reachability relation on undirected graphs. In a DAG-width game, the robber is allowed to run only along *directed paths*. The DAG-width $\mathrm{dagw}(\mathcal{G})$ of a graph $\mathcal{G}$ is the minimal number k such that the cops have a winning strategy in the game $\mathrm{dagwG}_k(\mathcal{G})$. Note the difference to treewidth where the parameter of the game is defined by $k + 1$ in order to make forests have treewidth 1.

Kelly-width. The Kelly-width game $\mathrm{KwG}_k(\mathcal{G})$ is played on a directed graph $\mathcal{G}$ in the same way as the DAG-width game, but the robber is, first, invisible for the cops and, second, inert [10]. Invisibility means that a winning strategy for the cops must not depend on the robber vertex and the cops can make assumptions about it only from their own moves. Inertness of the robber means that the robber can change his vertex only if a cop has announced to occupy the robber vertex in the next position. Formally,

a cop position is a tuple (C,R) where C is as before and $R \subseteq V$ is disjoint with C. The cops can move to a robber position (C,C',R). The moves of the robber are determined by the current position, so, in fact, we have a one-player game: the next position is (C',R') where $R' = (R \cup \mathrm{Reach}_{\mathcal{G}-(C\cap C')}(R \cap C')) \setminus C'$. The term $\mathrm{Reach}_{\ldots}(R \cap C')$ describes the inertness of the robber and the term $R \cup \ldots$ means that the robber may still be on a previous vertex if no cop is about to occupy it. Kelly-width is defined analogously to DAG-width.

Directed pathwidth. The directed pathwidth game is played as the Kelly-width game, but the robber is not inert. Formally, the position following a robber position (C,C',R) is (C',R') where $R' = \mathrm{Reach}_{\mathcal{G}-(C\cap C')}(R \cap C') \setminus C'$. Similar to treewidth, directed pathwidth $\mathrm{dpw}(\mathcal{G})$ of $\mathcal{G}$ is the minimal number k such that the cops have a winning strategy in $\mathrm{dpw}_{k+1}(\mathcal{G})$.

Entanglement. The entanglement game $\mathrm{entG}_k(\mathcal{G})$ [3] is slightly different from the games defined above. First, the robber can move only along an edge rather than along a whole path. Second, he is *obliged* to leave his vertex, no matter if a cop is about to occupy it or not (thus no cops are needed on an acyclic graph). Third, the cops are restricted in their moves as well. In a cop position (C,v), one cop can go to the v, other cops must remain on their vertices. Another possibility for the cops is to stay idle. More formally, cop positions are of the form (C,v) and the cops can move to some position (C',v) where $C' = C$, or $C' = C \cup \{v\}$ (if a new cop comes in to the graph), or $C' = (C \cup \{v\}) \setminus \{w\}$ where $w \in C$ is distinct from v. From a position (C',v), the robber can move to a position (C',v') where $(v,v') \in E$ and $v' \notin C'$. Unlike all games above, in the entanglement game, the cops do not need to play monotonically, so they win all finite plays and the robber wins all infinite plays. Entanglement $\mathrm{ent}(\mathcal{G})$ of a graph $\mathcal{G}$ is the minimal number k such that the cops have a winning strategy in $\mathrm{entG}_k(\mathcal{G})$.

Cliquewidth was introduced in [5]. Let C be a finite set of labels. A C-labelled graph is a tuple $\mathcal{G} = (V,E,\gamma)$ where $\gamma\colon V \to C$ is a map that labels the vertices of $\mathcal{G}$ with colours from C. An a-port is a vertex with colour a. Let k be a positive natural number and let $|C| \leq k$. The class $\mathcal{C}_k$ of graphs of cliquewidth at most k is defined inductively by the following operations.

(1) For every $a \in C$, a single a-port without edges is in $\mathcal{C}_k$.

(2) If $\mathcal{G}_1 = (V_1,E_1,\gamma_1)$ and $\mathcal{G}_2 = (V_2,E_2,\gamma_2)$ are in $\mathcal{C}_k$ then the disjoint union $\mathcal{G}_1 \oplus \mathcal{G}_2 = (V,E,\gamma)$ of $\mathcal{G}_1$ and $\mathcal{G}_2$ is in $\mathcal{C}_k$ where $V = V_1 \dot\cup V_2$, $E = E_1 \dot\cup E_2$, and $\gamma(v) = \gamma_1(v)$ if $v \in V_1$ and $\gamma(v) = \gamma_2(v)$ if $v \in V_2$.

(3) If $\mathcal{G} = (V,E,\gamma)$ is in $\mathcal{C}_k$ then the graph $\mathcal{G}'$ obtained by recolouring every a-port to a b-port is in $\mathcal{C}_k$, i.e. $\mathcal{G}' = (V,E,\gamma')$ where $\gamma'(v) = \gamma'(v)$ if $\gamma(v) \neq a$ and $\gamma'(v) = b$ otherwise.

(4) If $\mathcal{G} = (V,E,\gamma)$ is in $\mathcal{C}_k$ then the graph $\mathcal{G}'$ obtained by connecting all a-ports to all b-ports is in $\mathcal{C}_k$, i.e. $\mathcal{G}' = (V,E',\gamma)$ where $E' = E \cup \{(v,w) \mid \gamma(v) = a \text{ and } \gamma(w) = b\}$.

The cliquewidth $\mathrm{cw}(\mathcal{G})$ of a graph $\mathcal{G} = (V,E,\gamma)$ is the least k such that the graph (V,E) is in $\mathcal{C}_k$.

The following theorem is a combination of results proved in [2, 3, 4, 10, 15, 16].

Theorem 1. *Let C be any class of finite graphs on which at least one of the following measures is bounded: treewidth, directed pathwidth, DAG-width, cliquewidth, Kelly-width, entanglement. Then the winning regions of parity games on C are computable in polynomial time.*

It follows directly from the definitions that DAG-width and Kelly-width are bounded in directed pathwidth.

Theorem 2. *For a graph $\mathcal{G}$, we have* $\mathrm{dagw}(\mathcal{G}) \leq \mathrm{dpw}(\mathcal{G}) + 1$ *and* $\mathrm{Kw}(\mathcal{G}) \leq \mathrm{dpw}(\mathcal{G}) + 1$.

4 Friedmann's counterexamples

We now describe and analyze the graphs that underlie Friedmann's counterexample games. For most of the rules, these graphs have a rather similar structure, and for reasons of space we give a detailed pesentation just for the examples for the switch-all rule and Zadeh's Pivoting rule. Our analysis of the examples for the other rules will be summarized in a table, proofs will be given in the full version of this paper.

4.1 The Switch-All Rule

For $n \in \mathbb{N} \setminus \{0\}$, the graph $\mathcal{G}_n = (V_n, E_n)$ underlying Friedmann's games against the switch-all rule can defined as follows. The set of vertices is

$$V_n := \{x, s, c, r\} \cup \{t_i, a_i : 1 \leq i \leq 2n\} \cup \{d_i, e_i, g_i, k_i, f_i, h_i : 1 \leq i \leq n\}.$$

The set of edges and the graph $\mathcal{G}_3$ are given in Figure 1. The graph $\mathcal{G}_n$ consists of cycle gadgets induced by $\{d_i, e_i\}$ each encoding a bit which is considered to be set if the current strategy of Player 0 is to move from d_i to e_i and unset otherwise. Intuitively, the strategy improvement algorithm with the switch-all rule starts from the state where all bits are unset and increases the bit counter by one in each round. The subgraph induced by all h_j, k_j, g_j, and f_j, for $j \leq n$ guarantees the algorithm to swap the least significant bit and the subgraph induced by a_j and t_j, for $j \leq 2n$ ensures that the other bits to change are swapped as well, see [9] for details.

Friedmann showed in [9] that, for every $n > 0$, there is a parity game of size $O(n^2)$ with underlying graph $\mathcal{G}_n$ such that the strategy improvement algorithm with the switch-all rule requires at least 2^n improvement steps on that game.

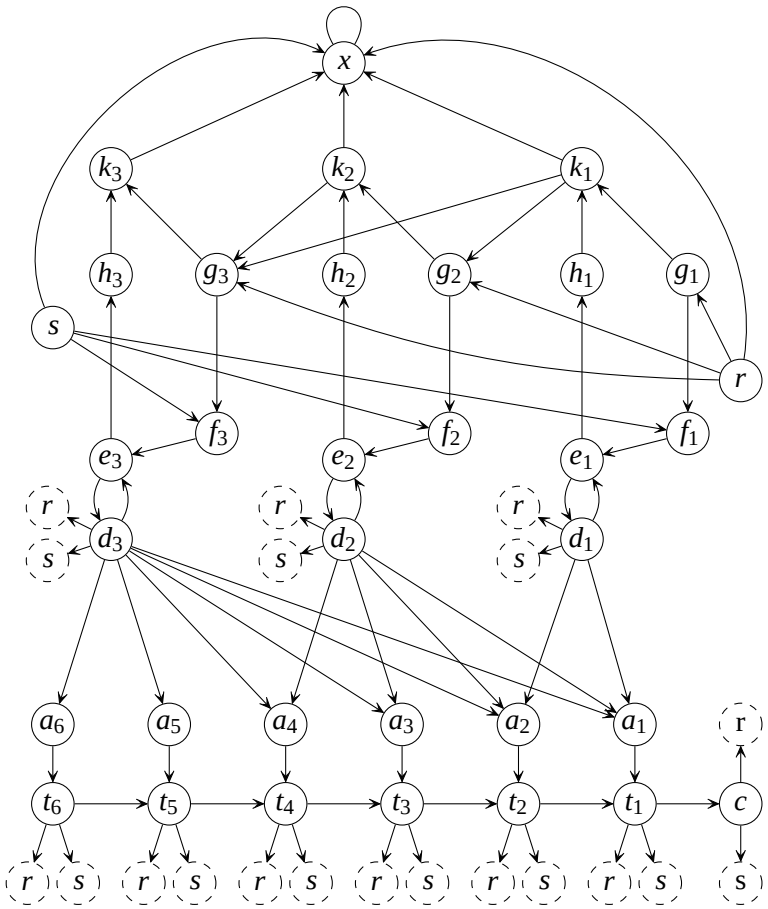
We shall now establish upper bounds for DAG-width, Kelly-width, directed pathwidth, entanglement, and cliquewidth of the graphs $\mathcal{G}_n$, which imply by Theorem 1, that Friedmann's games belong to natural classes of parity games that can be solved efficiently by other approaches than the strategy improvement algorithm. We start with an analysis of treewidth of $\mathcal{G}_n$ and show that it is unbounded on the class of graphs $\mathcal{G}_n$. Recall that treewidth of a directed graph $\mathcal{G} = (V, E)$ is defined by the treewidth of $\overline{\mathcal{G}} = (V, \overline{E})$ where $\overline{E}$ is the symmetrical closure of E. The reason for treewidth to be unbounded is that it contains arbitrarily large complete bipartite graphs $\mathcal{K}_{n,n}$ as subgraphs, whereby $\mathrm{tw}(\mathcal{K}_{n,n}) = n$. Indeed, every vertex has n direct successors, so if the robber is caught staying on a vertex v, all successors of v and v itself must be occupied by cops. The following lemma shows that we can find an arbitrary complete bipartite graph as a subgraph of a graph of the family $\overline{\mathcal{G}}_n$.

Lemma 3. *For every $k > 0$, there is some $n > 0$ such that $\overline{\mathcal{G}}_n$ has $\mathcal{K}_{k,k}$ as a subgraph.*

Proof. Choose $n := \lceil \frac{k}{2} \rceil + k - 1$. The vertex $d_{\lceil \frac{k}{2} \rceil}$ is the first of the vertices $d_1, \ldots, d_n$ to be connected to the vertices $A := \{a_j : j \leq k\}$. The $k-1$ vertices d_i, $i = \lceil \frac{k}{2} \rceil + 1, \ldots, \lceil \frac{k}{2} \rceil + k - 1$ are connected to each vertex of A as well. Neither the vertices of A are directly connected to one another, nor are the vertices of $B := \{d_i \mid i = \lceil \frac{k}{2} \rceil, \ldots, \lceil \frac{k}{2} \rceil + k - 1\}$. It follows that $\overline{\mathcal{G}}[A \cup B]$ is isomorphic to $\mathcal{K}_{k,k}$. □

Corollary 4. *For every $k > 0$, there is some $n > 0$ such that $\mathrm{tw}(\mathcal{G}_n) > k$.*

Remark 5. *Although the treewidth of $\mathcal{G}_n$ is unbounded, there is another class of graphs with bounded treewidth, such that the strategy improvement algorithm with switch-all rule requires super-polynomial time. We shall see in Section 4.3 that for the random-edge rule, Friedmann's counterexample class has*



Node	Successors
t_1	$\{s, r, c\}$
$t_{i>1}$	$\{s, r, t_{i-1}\}$
a_i	$\{t_i\}$
c	$\{s, r\}$
d_i	$\{s, r, e_i\} \cup \{a_j \mid j \leq 2i\}$
e_i	$\{d_i, h_i\}$
g_i	$\{f_i, k_i\}$
k_i	$\{x\} \cup \{g_j \mid i < j \leq n\}$
f_i	$\{e_i\}$
h_i	$\{k_i\}$
s	$\{x\} \cup \{f_j \mid j \leq n\}$
r	$\{x\} \cup \{g_j \mid j \leq n\}$
x	$\{x\}$

Figure 1: The graph $\mathcal{G}_3$ and the edge relation of $\mathcal{G}_n$ for the counterexample to the switch-all rule.

bounded treewidth. In fact, that class requires super-polynomial time also for the switch-all rule, see [9] for details.

Now we prove that directed pathwidth of graphs $\mathcal{G}_n$ is bounded, which leads to boundedness of DAG-width and Kelly-width.

Theorem 6. *For all $n > 0$, we have* $\mathrm{dpw}(\mathcal{G}_n) \leq 3$.

Proof. We describe a monotone winning strategy for 4 cops in the directed pathwidth game. First, r and s are occupied by two cops who will stay there until the robber is caught. In the next round, the two other cops expel the robber from all vertices d_i, e_i, f_i, g_i, h_i, and k_i (if he is there). For $i = 1, \ldots, n$, starting with $i = 1$ the cops place a cop on e_i and then visit with the last remaining cop vertices d_i, f_i, g_i, h_i, and k_i in that order.

The robber may be on vertex x, or in the part of the graph induced by a_i, t_i and c, for $i \in \{1, \ldots, 2n\}$. The cop from k_n (one of those not on r or s) visits x and then a_n, t_n, a_{n-1}, t_{n-1}, $\ldots$, a_1, t_1 in that order and finally c. Obviously, the described strategy for 4 cops is monotone and guarantees that the robber is captured. □

By Theorem 2, we get the following corollary.

Corollary 7. *For all $n > 0$, we have* $\mathrm{dagw}(\mathcal{G}_n) \leq 4$ *and* $\mathrm{Kw}(\mathcal{G}_n) \leq 4$.

We modify the strategy from the proof of Theorem 6 to obtain a winning strategy for 3 cops in the entanglement game. That is necessary, as in the latter, the cops are not permitted to be placed on a vertex which is not occupied by the robber. We first need a lemma from [3].

Lemma 8. *The entanglement of a graph is one if, and only if, it is not acyclic and each of its strongly connected components contains a vertex whose removal makes the component acyclic.*

Theorem 9. *For all $n > 0$, we have* $\mathrm{ent}(\mathcal{G}_n) \leq 3$.

Proof. Let $\mathcal{G}_n^{r,s}$ be the graph which is obtained from $\mathcal{G}_n$ by deleting vertices r and s and all adjacent edges, i.e. $\mathcal{G}_n^{r,s} = \mathcal{G}_n[V_n \setminus \{r,s\}]$. The only strongly connected components of $\mathcal{G}_n^{r,s}$ where the robber can remain are the one induced by x and those induced by $\{d_i, e_i\}$. All other components are singletons without self-loops, so the robber can stay there only for one move. Each of the components induced by x or by $\{d_i, e_i\}$ have a vertex whose removal makes the component acyclic. By Lemma 8, there is a strategy σ for one cop to catch the robber on $\mathcal{G}_n^{r,s}$. Thus it suffices to prove that the cops can occupy r and s. They use one cop who moves according to σ until the robber is captured or visits r or s. Assume by the symmetry of argumentation, that the robber visits r. A second cop follows him to r and remains there until the end of the play. Then the first cop plays according to σ again. As r is occupied by a cop, the robber is either captured, or visits s. Then the last cop follows him to s. Finally, the first cop plays according to σ for the last time and the robber loses the play. □

We show that the cliquewidth of $\mathcal{G}_n$ is bounded as well. Graph $\mathcal{G}_n$ can be decomposed into n layers, the i-th layer is induced by vertices g_i, f_i, e_i, d_i, h_i, k_i, t_{2i}, t_{2i-1}, a_{2i}, and a_{2i-1}. We construct $\mathcal{G}_n$ inductively over $i = 1, \ldots, n$ connecting the new layer to the previous ones. Simultaneously, we connect r, and s to the i-th layer. Then vertex x is connected to the graph.

Theorem 10. *For all $n > 0$, we have* $\mathrm{cw}(\mathcal{G}_n) \leq 10$.

Proof. We consider graph $\mathcal{G}_n$ as consisting of layers $\mathcal{L}_i$, for $i \in \{1, \ldots, n\}$ where each $\mathcal{L}_i$ is induced by vertices d_i, e_i, f_i, h_i, k_i, g_i, a_{2i}, a_{2i-1}, t_{2i}, and t_{2i-1}. The layers are produced for $i = 1, 2 \ldots, n$ by induction on i and connected to the previous layers. Level 1 is constructed in the same way as further layers (up to vertex c, which is easy to produce), so we do not describe the base case explicitly. Assume, all layers from $\mathcal{L}_1$ to $\mathcal{L}_i$ are constructed with following labelling, which is an invariant that holds after a layer is constructed, see the first picture on Figure 2 (connections from t_i to r and s are not shown).

- For $j \in \{1, \ldots, 2i-1\}$, all t_j, and, for $j \in \{1, \ldots, i\}$, all d_j, e_j, h_j, and f_j have colour *Done*.
- t_{2i} has colour T.
- For $j \in \{1, \ldots, 2i\}$, all a_j, have colour A.
- For $j \in \{1, \ldots, i\}$, all k_j have colour K, and all g_j, have colour G .
- r has colour R and s has colour S.

We construct layer $i+1$ satisfying the invariant. First, create vertex a_{2i+1} with colour A and vertex t_{2i+1} with colour T' and connect $A \to T'$. Then take the union of the previous layers and $\{a_{2i+1}, t_{2i+1}\}$ and connect $T' \to T$, $T' \to R$ and $T' \to S$. Relabel $T' \to T$. Then repeat the same procedure with a_{2i+2} and t_{2i+2} instead of a_{2i+1} and t_{2i+1}, see the second picture on Figure 2.

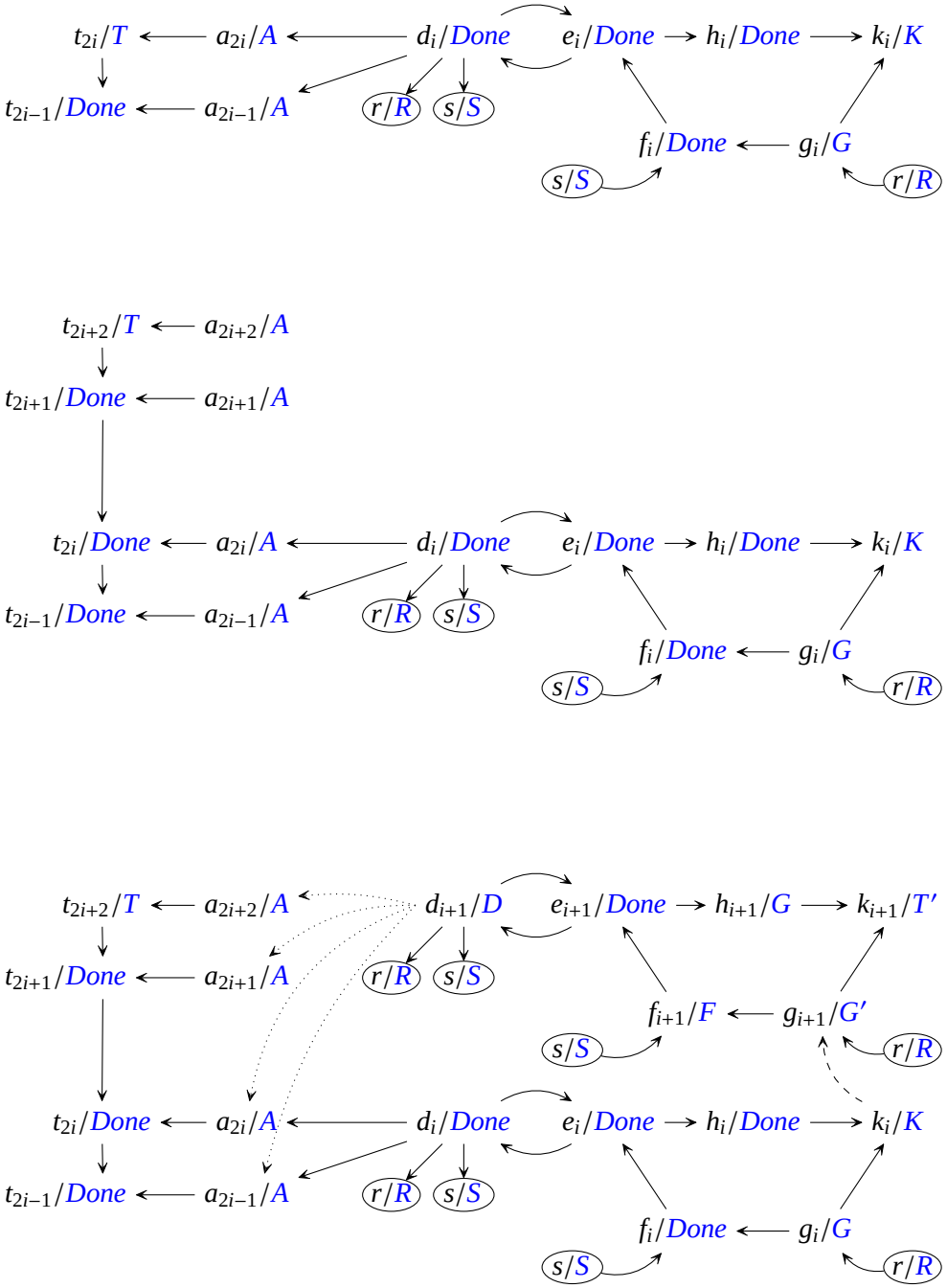
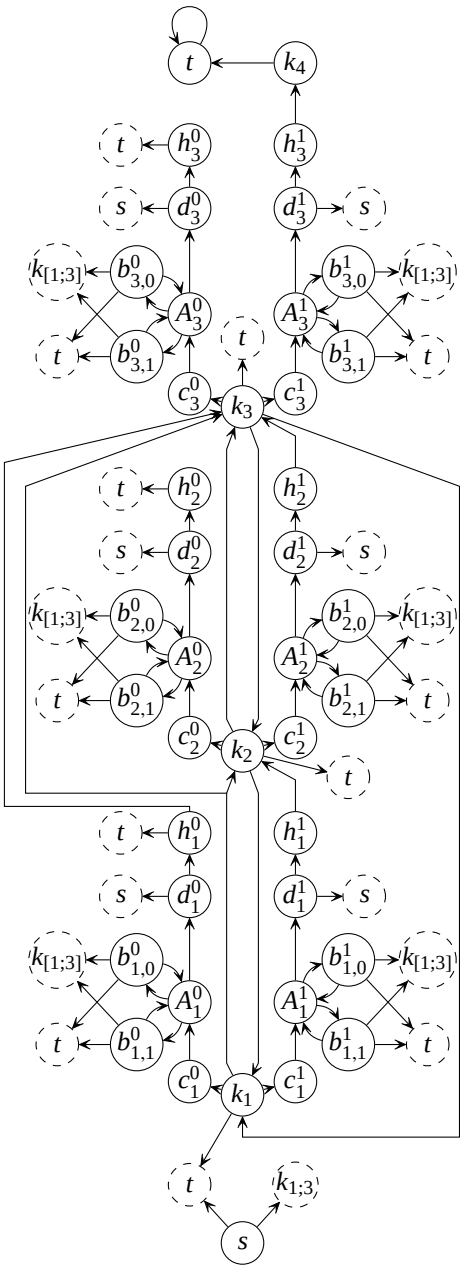


Figure 2: Construction of $\mathcal{G}_n$ with ten colours.



Node	Successors
d_i^j	h_i^j, s
A_i^j	$d_i^j, b_{i,0}^j, b_{i,1}^j$
$b_{i,*}^j$	$t, A_i^j, k_{[1;n]}$
t	t
s	$t, k_{[1;n]}$
k_{n+1}	t
k_i	$c_i^0 c_i^1, t, k_{[1;n]}$
h_i^0	$t, k_{[i+2;n]}$
h_i^1	k_{i+1}
c_i^j	A_i^j

Figure 3: The graph $\mathcal{Z}_3$ and the edge relation of $\mathcal{Z}_n$ for the counterexample to Zadeh's least-entered rule.

Now we construct the subgraph C_{i+1} induced by d_{i+1}, e_{i+1}, h_{i+1}, g_{i+1}, and f_{i+1} using colours *D*, *Done*, *G*, T', G', and *F*. Note that colours *Done* and T' are reused. Produce d_{i+1}, e_{i+1}, f_{i+1}, g_{i+1}, k_{i+1}, and h_{i+1} with labels *D*, *Done*, *F*, G', T', and *G*, respectively, and connect them as needed, also to *r* and to *s* (see the third picture on Figure 2).

Relabel $G \to Done$. Build the disjoint union of C_{i+1} and the already constructed graph. Connect $K \to G'$ (which connects all k_j, for $j < i+1$ to g_{i+1}; dashed line in the figure), and relabel $T' \to k$ and $G' \to G$.

Connect $D \to A$ (which connects d_{i+1} to all a_j, for $j \leq 2i$, dotted lines in the figure) and relabel $D \to \mathit{Done}$. This finishes the construction of layer L_{i+1}. Note that the properties from the invariant hold for L_{i+1}. Finally, produce vertex x with colour T' and connect all k_i to x and x to itself. It remains to count the colours. We used *Done*, T, T', A, F, G, G', K, R, and S, which makes ten colours.

□

4.2 Zadeh's least-entered rule

As a second example we discuss the counterexample of Friedmann against Zadeh's least-entered rule. The underlying game graphs are denoted $\mathcal{Z}_n$. As $\mathcal{G}_n$ the graph $\mathcal{Z}_n$ can be decomposed into n layers, see Figure 3 for graph $\mathcal{Z}_3$ and a list of edges of $\mathcal{Z}_n$. Vertices k_i, c_i^0, A_i^0, $b_{i,1}^0$, $b_{i,0}^0$, d_i^0, h_i^0, c_i^1, A_i^1, $b_{i,1}^1$, $b_{i,0}^1$, d_i^1, and h_i^1 induce the i-th layer. The subgraphs induced by c_i^0, A_i^0, $b_{i,1}^0$, $b_{i,0}^0$, d_i^0, h_i^0 and c_i^1, A_i^1, $b_{i,1}^1$, $b_{i,0}^1$, d_i^1, h_i^1 are isomorphic to each other.

A run of the strategy improvement alogrithm on $\mathcal{Z}_n$ simulates an n-bit counter with values from 0 to $2^n - 1$. The difference to the switch-all rule is that the least-entered rule chooses an improving edge that has been switched least often. Because the lower bits of an n-bit counter are switched more often, the higher bits would be switched before they should in order to catch up with the lower bits. This means that the n-bit counter would not go through all the steps from 0 to $2^n - 1$. Friedmann solved this problem by representing each bit i by two bits, i_0 and i_1. The associated structures in $\mathcal{Z}_n$ are the gadgets induced by $\{A_i^0, b_{i,1}^0, b_{i,0}^0\}$ and $\{A_i^1, b_{i,1}^1, b_{i,0}^1\}$ respectively. The bit i_j is considered to be set, if the current Player 0 strategy chooses both edges $(b_{i,0}^j, A_i^j)$ and $(b_{i,1}^j, A_i^j)$, and unset otherwise. In a run of the algorithm only one of the bits i_0 and i_1 is active and able to effect the rest of the counter at the time. The inactive bit can, in the meantime, switch back and forth from 0 to 1 in order to catch up with the rest of the counter without having an effect on it.

The counterexample contains a vertex k_i in each layer i such that all k_i induce an n-clique in the graph. This makes all values of measures that describe cyclicity (i.e., treewidth, directed pathwidth, DAG-width, and Kelly-width) unbounded on the class of the counterexample graphs, but cliquewidth of the graphs is still small.

Theorem 11. *For all $n > 0$, we have* $\mathrm{cw}(\mathcal{Z}_n) \leq 9$.

Proof. The proof is very similar to the proof of Theorem 10. We regard the graphs $\mathcal{Z}_n$ as consisting of layers $\mathcal{L}_i$ that are induced by vertices k_i, c_i^0, A_i^0, $b_{i,0}^j$, $b_{i,1}^j$, d_i^j, and h_i^j, for $i \in \{1, \ldots, n\}$ and $j \in \{0, 1\}$. The layers are constructed for $i = 1, 2, \ldots, n$ by induction on i and connected to the previous layers. In the induction step, we build a new layer and connect it to the previous ones. Finally, we add the vertex s and the top layer, that consists of t and k_{n+1}, and establish the connections to the other n layers.

As in the proof of Theorem 10, layer $\mathcal{L}_1$ is constructed in the same way as further layers. Assume, layers from $\mathcal{L}_1$ to $\mathcal{L}_i$ have been constructed with the following labelling, which is an invariant that holds after a new layer is constructed.

- For $j \in \{1, \ldots, i\}$ and $s, s' \in \{1, 2\}$, all k_j have colour K, all A_j^s and c_j^s have colour *Done*, all d_j^s have colour D, and all $b_{i,s'}^s$ have colour B.
- For $j \in \{1, \ldots, i-1\}$, all h_j^0 have colour H and all h_j^1 have colour *Done*.
- h_i^0 has colour H_l and h_i^1 has colour H_r.

We construct the layer $\mathcal{L}_{i+1}$ and connect it to the previous layers such that at the end of that process the invariant is true. First, produce the vertex k_{i+1} with colour K'. Connect the vertices $h_1^0, \ldots, h_{i-1}^0$

and the vertex h_i^1 to k_{i+1} by connecting H-ports and H_r-ports to K'-ports, and relabelling $H_l \to H$ and $H_r \to \mathit{Done}$. Extend the clique consisting of the vertices $k_1, \ldots, k_i$ by connecting $K \to K'$ and $K' \to K$. Thus the conncections between $\mathcal{L}_{i+1}$ and the previous layers have been established.

Next, we construct the rest of $\mathcal{L}_{i+1}$ using colours C, Done, B, D, H_l and H_r. Create vertices c_{i+1}^0, A_{i+1}^0, $b_{i+1,1}^0$, $b_{i+1,0}^0$, d_{i+1}^0 and h_{i+1}^0 labelled with colours C, Done, B, B, D and H_l, respectively, and connect them as needed. Repeat this procedure for vertices c_{i+1}^1, A_{i+1}^1, $b_{i+1,1}^1$, $b_{i+1,0}^1$, d_{i+1}^1 and h_{i+1}^1 with the difference that h_{i+1}^1 obtains colour H_r. Build the disjoint union of these two subgraphs and the already constructed graph. Connect k_{i+1} to c_{i+1}^0 and c_{i+1}^1 by $K' \to C$. Relabel $K' \to K$ and $C \to \mathit{Done}$. This finishes the construction of $\mathcal{L}_{i+1}$. Note that the invariant for the vertex labels is satisfied.

After all n layers have been built, relabel $H_l \to H$ and create vertex s with colour K' (which is reused). Connect $K' \to K$ and $D \to K'$. Relabel $D \to \mathit{Done}$. It remains to add vertices k_{n+1} and t to the graph. Create k_{n+1} and t with colours C and D. Connect $H_r \to C$, $C \to D$, $D \to D$, $K \to D$, $B \to K$, $B \to D$ and $K' \to D$. This produces the top layer induced by k_{i+1} and t and establishes the edges between the first n layers and the vertex s and the top layer. Note that we reused the colours D, C and K'. It remains to count the colours. We used the nine colours Done, K, K', C, B, D, H_r, H_l and H. Hence, our claim holds. □

4.3 Other rules

The graph complexity of the other counterexamples constructed by Friedmann is similar to the one of the graphs $\mathcal{G}_n$. All of them can be decomposed into layers which are connected in a simple way, and they can be treated by the same techniques as in Theorems 6, 9 and 10. The results are shown in Table 1. For the case of snare memorisation we did not find any bound for the cliquewidth, but all the other measures (except treewidth) have very small values on those graphs.

Rule / Measure	tree	directed path	DAG	Kelly	Entanglement	Clique
switch-all	∞	3	4	4	3	10
switch-best	∞	3	4	4	3	18
random-edge	8	3	4	4	3	12
random-facet	3	1	2	2	1	6
least-entered	∞	∞	∞	∞	∞	9
least-considered	7	3	4	4	4	7
snare memory	∞	3	4	4	4	?

Table 1: Upper bounds in different measures for the counterexample graph classes.

References

[1] Krzysztof Apt & Erich Grädel, editors (2011): *Lectures in Game Theory for Computer Scientists*. Cambridge University Press, doi:10.1017/CBO9780511973468.

[2] Dietmar Berwanger, Anuj Dawar, Paul Hunter, Stephan Kreutzer & Jan Obdržálek: *The DAG-Width of Directed Graphs*, doi:10.1016/j.jctb.2012.04.004. To appear.

[3] Dietmar Berwanger & Erich Grädel (2005): *Entanglement – A Measure for the Complexity of Directed Graphs with Applications to Logic and Games*. In: *Proceedings of LPAR 2004, Montevideo, Uruguay*, *LNCS* 3452, Springer, pp. 209–223, doi:10.1007/978-3-540-32275-7_15.

[4] Dietmar Berwanger, Erich Grädel, Łukasz Kaiser & Roman Rabinovich: *Entanglement and the Complexity of Directed Graphs*. *Theoretical Computer Science*. Available at http://logic.rwth-aachen.de/~rabinovich/entangle.pdf. To appear.

[5] Bruno Courcelle, Joost Engelfriet & Grzegorz Rozenberg (1993): *Handle-Rewriting Hypergraph Grammars*. *J. Comput. Syst. Sci.* 46(2), pp. 218–270, doi:10.1016/0022-0000(93)90004-G.

[6] John Fearnley (2010): *Non-oblivious Strategy Improvement*. In Edmund M. Clarke & Andrei Voronkov, editors: *LPAR (Dakar)*, *LNCS* 6355, Springer, pp. 212–230, doi:10.1007/978-3-642-17511-4_13.

[7] Oliver Friedmann: *A Subexponential Lower Bound for Policy Iteration Based on Snare Memorization*. Available at http://files.oliverfriedmann.de/papers/fearnley_lower_bound.pdf. Submitted for publication.

[8] Oliver Friedmann: *A Subexponential Lower Bound for the Least Recently Considered Rule for Solving Linear Programs and Games*. Available at http://files.oliverfriedmann.de/papers/least_recently_considered_lower_bound.pdf. Submitted for publication.

[9] Oliver Friedmann (2011): *Exponential Lower Bounds for Solving Infinitary Payoff Games and Linear Programs*. Ph.D. thesis, Ludwig-Maximilians-University Munich. Available at http://edoc.ub.uni-muenchen.de/13294.

[10] P. Hunter & S. Kreutzer (2008): *Digraph Measures: Kelly Decompositions, Games, and Orderings*. *Theor. Comput. Sci.* 399(3), pp. 206–219, doi:10.1016/j.tcs.2008.02.038.

[11] Marcin Jurdziński (1998): *Deciding the Winner in Parity Games is in UP ∩ co-UP*. *Inf. Process. Lett.* 68(3), pp. 119–124, doi:10.1016/S0020-0190(98)00150-1.

[12] Marcin Jurdziński (2000): *Small Progress Measures for Solving Parity Games*. In Horst Reichel & Sophie Tison, editors: *STACS*, *LNCS* 1770, Springer, pp. 290–301, doi:10.1007/3-540-46541-3_24.

[13] Marcin Jurdziński, Mike Paterson & Uri Zwick (2006): *A Deterministic Subexponential Algorithm for Solving Parity Games*. In: *SODA*, ACM Press, pp. 117–123, doi:10.1145/1109557.1109571. Available at http://dl.acm.org/citation.cfm?id=1109557.

[14] Marcin Jurdziński & Jens Vöge (2000): *A Discrete Strategy Improvement Algorithm for Solving Parity Games (Extended Abstract)*. In E. A. Emerson & A. P. Sistla, editors: *Computer Aided Verification, 12th International Conference, CAV 2000, Proceedings*, *LBCS* 1855, Springer, Chicago, IL, USA, pp. 202–215, doi:10.1007/10722167_18.

[15] Jan Obdržálek (2003): *Fast Mu-Calculus Model Checking when Tree-Width Is Bounded*. In Warren A. Hunt Jr. & Fabio Somenzi, editors: *CAV*, *LNCS* 2725, Springer, pp. 80–92, doi:10.1007/978-3-540-45069-6_7.

[16] Jan Obdržálek (2007): *Clique-Width and Parity Games*. In J. Duparc & T. A. Henzinger, editors: *CSL '07*, *LNCS* 4646, Springer, pp. 54–68, doi:10.1007/978-3-540-74915-8-8.

[17] Sven Schewe (2008): *An Optimal Strategy Improvement Algorithm for Solving Parity and Payoff Games*. In Michael Kaminski & Simone Martini, editors: *CSL*, *LNCS* 5213, Springer, pp. 369–384, doi:10.1007/978-3-540-87531-4_27.