Theoretical Computer Science 560 (2014) 235–250

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

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

Felix Canavoi, Erich Grädel, Roman Rabinovich [*,1]

RWTH Aachen University, Germany

ARTICLE INFO

Available online 9 January 2014

Keywords:
Parity games
Graph complexity measures
Treewidth

ABSTRACT

The problem whether winning regions and wining strategies for parity games can be computed in polynomial time is a major open problem in the field of infinite games, which is relevant for many applications in logic and formal verification. For some time the discrete strategy improvement algorithm due to Jurdziński and Vöge had been considered to be a candidate for solving parity games in polynomial time. However, it has recently been proved by Oliver Friedmann that this 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 time lower bounds on natural classes of parity games on which efficient algorithms are known.

1. Introduction

Parity games are a family of infinite games on directed graphs. They are played by two players, player 0 and player 1, whose moves consist in shifting a pebble from a vertex to a vertex along edges. Vertices have marks indicating the player to move, and priorities (colours) from a finite set of natural numbers. A player wins a finite play if it ends in a vertex that belongs to the opponent and has no outgoing edges. Otherwise the players construct an infinite play and thus an infinite sequence of colours. If the greatest infinitely often appearing colour is even, player 0 wins, otherwise player 1 wins.

Parity games are important for several reasons. Many classes of games arising in practical applications admit reductions to parity games (over larger game graphs). This is not only 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

* Corresponding author.
E-mail addresses: canavoi@logic.rwth-aachen.de (F. Canavoi), graedel@logic.rwth-aachen.de (E. Grädel), rabinovich@logic.rwth-aachen.de (R. Rabinovich).
[1] This work was partially supported by ESF, www.esf.org.

0304-3975/$ – see front matter
http://dx.doi.org/10.1016/j.tcs.2013.12.021

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 ∩ Co-NP. In fact, it was proved in [13] that the problem is in UP ∩ Co-UP, where UP denotes the class of NP-problems with unique witnesses. The best known deterministic algorithm has complexity $n^{O(\sqrt{n})}$ [15]. For parity games with a number d of priorities the progress measure lifting algorithm by Jurdziński [14] 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 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 [18], bounded entanglement [3,4], bounded DAG-width [2], bounded Kelly-width [12], or bounded cliquewidth [19].

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 [16]. 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 parametrised 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 [8] 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 parametrise 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 initial 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 [16] 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 possibility. 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 optimising* rule [16] 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 *snare memorisation* in [7]. 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 [16,21]. 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 [8,10,9]. 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 (minus one in some cases). Hence, on simple graphs, few cops suffice to win whereas 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 are 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 $|C| \leqslant k$ 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 $|C'| \leqslant k$. 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 partial function $\sigma : 2^V \times V \to 2^V$ which prescribes, for every cop position (C, v), the next placement $\sigma(C, v)$. Similarly, a (positional) strategy for the robber is a function $\rho : 2^V \times 2^V \times V \to V$ which maps a robber position (C, C', v) to a cop position (C', w) with $w \in \mathrm{Reach}_{\mathcal{G}-(C\cap C')}(v) \setminus C'$. 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. The winning condition for the cops is a reachability condition (in both the treewidth and the pathwidth games). Indeed, the cops lose if the robber reaches a non-monotone position, regardless whether they capture him later or not. Hence, we can assume without loss of generality that a play stops in a non-monotone position and the robber wins. Then the winning condition for the cops is precisely to reach a position in which the robber is captured. It follows that the game are determined and that positional strategies suffice for both players.

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.

Example 1. We consider narrow long grids. Let $\mathcal{G}_{mn} = (V, E)$ be an undirected $(m \times n)$-grid with additional shortcuts from left to right, i.e. $V = \{1, \ldots, m\} \times \{1, \ldots, n\}$ and $E = \{((i, j), (k, l)) \mid |i - k| + 1$ and $j = l$, or $i = k$ and $|j - l| = 1\} \cup \{((i, j), (k, l)) \mid j < k\}$. Then $\mathrm{dagw}(\mathcal{G}_{m,n}) = m + 1$. Indeed, $m + 1$ cops win in the same way as on the undirected grid: they occupy the leftmost column $\{i, 1 \mid 1 \leqslant i \leqslant m\}$, place the remaining cop on $(1, 2)$, then move the cop from $(1, 1)$ to $(2, 2)$, then the cop from $(2, 1)$ to $(3, 2)$ and so on until they occupy the whole column $\{(i, 2)\}$. In this way they expel the robber from every column. Note that the additional edges do not help the robber, as they induce no additional cycles. On the other hand, it is well known that the treewidth of the grid even without the additional edges is at least n (which corresponds to $n + 1$ cops).

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 [12]. 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 $R \cup \cdots$ means that the robber may still be on a previous vertex if no cop is about to occupy it. Respecting the rules for the moves, Kelly-width is defined analogously to DAG-width.

Directed pathwidth The 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 the treewidth, the directed pathwidth $\mathrm{dpw}(\mathcal{G})$ of $\mathcal{G}$ is the minimal number k such that the cops have a winning strategy in $\mathrm{dpw}_{k+1}(\mathcal{G})$ where $\mathrm{dpw}_{k+1}(\mathcal{G})$ is the directed pathwidth game with $k + 1$ cops on $\mathcal{G}$.

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 vertex 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 into 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 $\mathrm{entG}_k(\mathcal{G})$.

Example 2. We consider narrow long grids. Let $\mathcal{G}_{mn}$ be an $(m \times n)$-grid where $m < n + 3$. We can see that, for $n \geqslant 10$, $\mathrm{ent}(\mathcal{G}_{2,n}) = 4$. Four cops have the following winning strategy. Two of them place themselves on $(0, \lfloor \frac{n}{2} \rfloor)$ and $(1, \lfloor \frac{n}{2} \rfloor)$ and build a *wall*. We call the other cops *chasers*. Due to the symmetry, assume without loss of generality that the robber hides in the right part of the grid (with larger second coordinate). The goal of the cops is to shift the wall to the right of two cops on $(0, \lfloor \frac{n}{2} \rfloor)$ and $(1, \lfloor \frac{n}{2} \rfloor)$ such that the robber remains to the right of it. The new wall will consist of the chasers. When it is constructed, the cops from the old wall become free from guarding the left part of the grid and the roles of the cops change: the chasers guard the robber and the cops from the old wall become chasers.

The shift of the wall is done as follows. One of the chasers follows the robber until he moves vertically, i.e. from (i, j) to $(1 - i, j)$, for some $i \in \{0, 1\}$ and $j \in \{\lfloor \frac{n}{2} \rfloor, \ldots, n - 1\}$. Then the second chaser goes to $(1 - i, j)$. If the robber now makes a move to the right, the wall is shifted. Otherwise, the chaser from (i, j) follows the robber. Both chasers continue to follow the robber in a leap-frogging manner to the left until he moves vertically. That happens at the latest when the current wall is reached. Then the rightmost chaser follows him. Again, if he goes to the right, the wall is shifted. He can go further to the left followed by the cop, but this process can continue at most until the robber hits the wall. So finally he moves vertically and then to the right.

The fact that the robber wins on $\mathcal{G}_{2,10}$ against 3 cops can be proven by inspecting all possible finite play prefixes that have no repetitions of positions.

Cliquewidth 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 $\gamma : V \to C$ 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 $|C| \leqslant k$. 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 [2–4,12,18,19].

Theorem 3. *Let $\mathcal{C}$ 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 for both players in parity games on graphs from $\mathcal{C}$ are computable in polynomial time.*

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

Theorem 4. *For a graph $\mathcal{G}$, we have* $\mathrm{dagw}(\mathcal{G}) \leqslant \mathrm{dpw}(\mathcal{G}) + 1$ *and* $\mathrm{Kw}(\mathcal{G}) \leqslant \mathrm{dpw}(\mathcal{G}) + 1$.

4. Friedmann's counterexamples

We now describe and analyse the graphs that underlie Friedmann's counterexample games for different rules. Note that the information about the priorities of the vertices and about which player they belong to is irrelevant for the analysis of the complexity of the underlying graphs. For most of the rules, these graphs have a rather similar structure, which implies similar proofs for out statements. We give detailed presentations only for the examples for the switch-all rule, Zadeh's Pivoting rule, and the random-edge rule. Our analysis of the examples for all rules we consider is summarised in Table 3.

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 be defined as follows. The set of vertices is

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

The set of edges and the graph $\mathcal{G}_3$ are given in Fig. 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 $j \leqslant n$ guarantees the algorithm to swap the least significant bit and the subgraph induced by a_j and t_j, for $j \leqslant 2n$ ensures that the other bits to change are swapped as well, see [8] for details.

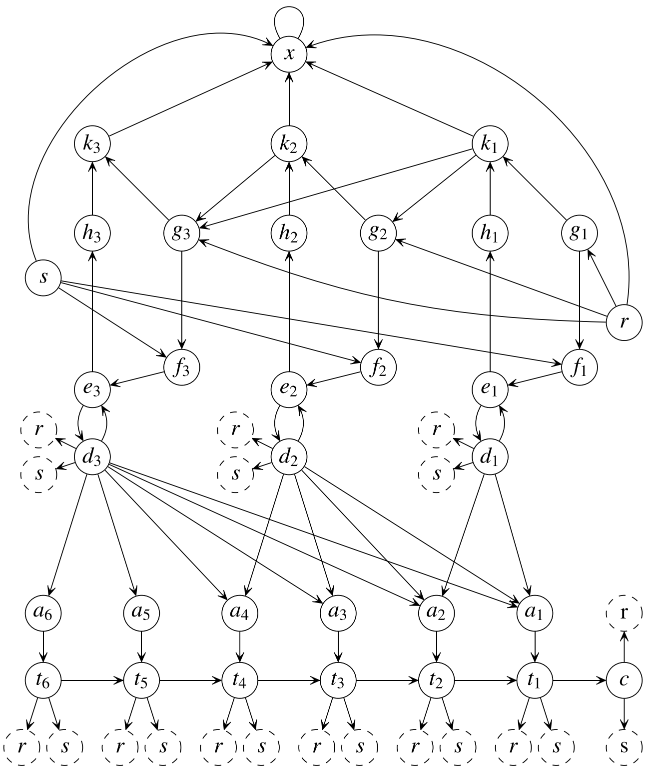
Friedmann showed in [8] that, for every $n > 0$, there is a parity game of size $O(n^2)$ with underlying graph $\mathcal{G}_n$ such that the strategy improvement algorithm with the switch-all rule requires at least 2^n improvement steps on that game.

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 3, 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 neighbours, 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 1. *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 $d_1, \dots, d_n$ to be connected to the vertices $A := \{a_j \colon j \leqslant k\}$. The $k - 1$ vertices d_i, $i = \lceil \frac{k}{2} \rceil + 1, \dots, \lceil \frac{k}{2} \rceil + k - 1$ 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 $B := \{d_i \mid i = \lceil \frac{k}{2} \rceil, \dots, \lceil \frac{k}{2} \rceil + k - 1\}$. It follows that $\overline{\mathcal{G}}[A \cup B]$ is isomorphic to $\mathcal{K}_{k,k}$. $\square$

Corollary 1. *For every $k > 0$, there is some $n > 0$ such that* $\mathrm{tw}(\mathcal{G}_n) > k$.



(a) The graph $\mathcal{G}_3$

Vertex	Successors	Vertex	Successors	Vertex	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	$\{x\}$				

(b) The edge relation of $\mathcal{G}_n$

Fig. 1. The graph $\mathcal{G}_n$ for the switch-all rule.

Remark 5. Although the treewidth of graphs $\mathcal{G}_n$ 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 4.4 that for the random-edge rule, Friedmann's counterexample class has bounded treewidth. In fact, that class requires super-polynomial time also for the switch-all rule, see [8] for details.

Now we prove that the 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* $\mathrm{dpw}(\mathcal{G}_n) \leqslant 3$.

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 4, we get the following corollary.

Corollary 2. *For all $n > 0$, we have* $\mathrm{dagw}(\mathcal{G}_n) \leqslant 4$ *and* $\mathrm{Kw}(\mathcal{G}_n) \leqslant 4$.

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 2. *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 7. *For all $n > 0$, we have* $\mathrm{ent}(\mathcal{G}_n) \leqslant 3$.

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 2, 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 $i = 1, \dots, n$ 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 8. *For all $n > 0$, we have* $\mathrm{cw}(\mathcal{G}_n) \leqslant 10$.

Proof. We consider graph $\mathcal{G}_n$ as consisting of layers $\mathcal{L}_i$, for $i \in \{1, \dots, n\}$ 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 $i = 1, 2 \dots, n$ 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 in Fig. 2 (connections from t_i to r and s are not shown).

- For $j \in \{1, \dots, 2i - 1\}$, all t_j, and, for $j \in \{1, \dots, i\}$, all d_j, e_j, h_j, and f_j have colour *Done*.
- t_{2i} has colour T.
- For $j \in \{1, \dots, 2i\}$, all a_j, have colour A.
- For $j \in \{1, \dots, i\}$, 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 in Fig. 2.

Now we construct the subgraph $\mathcal{C}_{i+1}$ induced by d_{i+1}, e_{i+1}, h_{i+1}, g_{i+1}, k_{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 in Fig. 2).

Relabel $G \to$ *Done*. Build the disjoint union of $\mathcal{C}_{i+1}$ 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 $j \leqslant 2i$, dotted lines in the figure) and relabel $D \to$ *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 $\mathcal{Z}_n$. The set of vertices is

$$
\begin{aligned}
V_n := &\left\{b_{i,0}^0, b_{i,0}^1, b_{i,1}^0, b_{i,1}^1, d_i^0, d_i^1, h_i^0, h_i^1, c_i^0, c_i^1, A_i^0, A_i^1 \;\middle|\; 1 \leqslant i \leqslant n\right\} \\
&\cup \{k_i \mid 1 \leqslant i \leqslant n+1\} \cup \{t, s\}.
\end{aligned}
$$

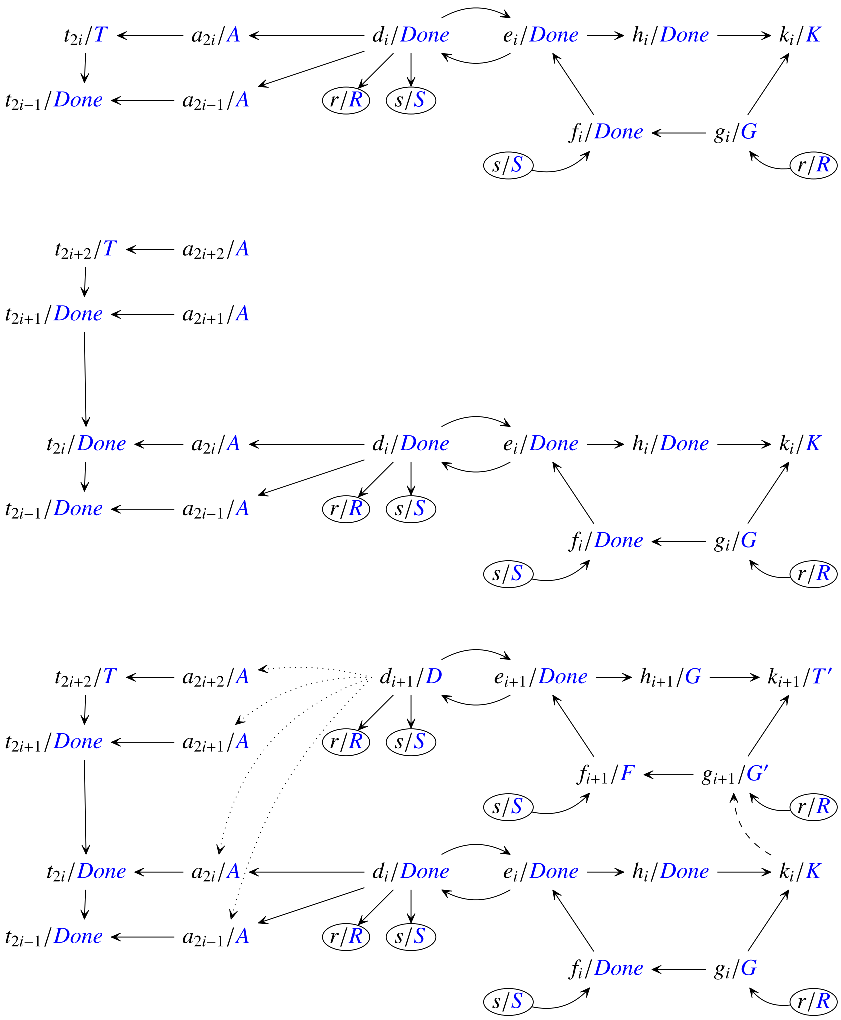


Fig. 2. Construction of $\mathcal{G}_n$ with ten colours.

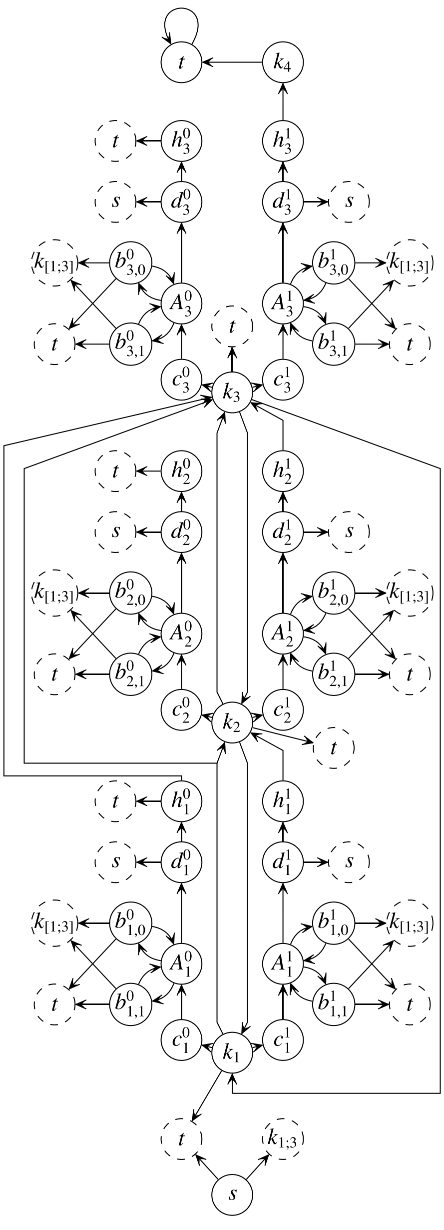
Similar to $\mathcal{G}_n$ the graph $\mathcal{Z}_n$ can be decomposed into n layers, see Fig. 3a for graph $\mathcal{Z}_3$ and Fig. 3b for the edge relation of $\mathcal{Z}_n$. In Fig. 3b we denote for a sequence of vertices $k_1, \ldots, k_n$ and two natural numbers l, m, $l \leqslant m$, by $k_{[l,m]}$ the set of vertices $\{k_l, \ldots, k_m\}$. 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 algorithm 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, $j \in \{0, 1\}$, 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 is 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 9. *For all* $n > 0$, *we have* $\mathrm{cw}(\mathcal{Z}_n) \leqslant 9$.

Proof. The proof is very similar to the proof of Theorem 8. 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 $i =$



(a) The graph $\mathcal{Z}_3$

Vertex	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

(b) The edge relation of $\mathcal{Z}_n$

Fig. 3. The graph $\mathcal{Z}_n$ for Zadeh's least-entered rule.

$1, 2, \ldots, n$ 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 8, 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 con-

Table 1
The edge relation of $\mathcal{H}_n$ for the switch-best rule.

Vertex	Successors	Vertex	Successors
t_1	$\{s, r, c\}$	y_i	$\{f_i, k_i\}$
$t_{i>1}$	$\{s, r, t_{i-1}\}$	g_i	$\{y_i, k_i\}$
a_i	$\{t_i\}$	k_i	$\{x\} \cup \{g_j \mid i < j \leqslant n\}$
c	$\{r\}$	f_i	$\{e_i\}$
d_i^1	$\{s, c, d_i^2\} \cup \{a_{3j+3} \mid j \leqslant 2i-2\}$	h_i	$\{k_i\}$
d_i^2	$\{d_i^3\} \cup \{a_{3j+2} \mid j \leqslant 2i-2\}$	s	$\{x\} \cup \{f_j \mid j \leqslant n\}$
d_i^3	$\{e_i\} \cup \{a_{3j+1} \mid j \leqslant 2i-1\}$	r	$\{x\} \cup \{g_j \mid j \leqslant n\}$
e_i	$\{d_i^1, h_i\}$	x	$\{x\}$

necting H-ports and H_r-ports to K'-ports, and relabelling $H_l \to H$ and $H_r \to Done$. Extend the clique consisting of the vertices $k_1, \ldots, k_i$ by connecting $K \to K'$ and $K' \to K$. Thus the connections 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^0_{i+1}, A^0_{i+1}, $b^0_{i+1,1}$, $b^0_{i+1,0}$, d^0_{i+1} and h^0_{i+1} labelled with colours C, $Done$, B, B, D and H_l, respectively, and connect them as needed. Repeat this procedure for vertices c^1_{i+1}, A^1_{i+1}, $b^1_{i+1,1}$, $b^1_{i+1,0}$, d^1_{i+1} and h^1_{i+1} with the difference that h^1_{i+1} obtains colour H_r. Build the disjoint union of these two subgraphs and the already constructed graph. Connect k_{i+1} to c^0_{i+1} and c^1_{i+1} by $K' \to C$. Relabel $K' \to K$ and $C \to Done$. 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 $D \to Done$. 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. The switch-best rule

Switch-best is a deterministic memoryless improvement rule that computes in every step of the iteration the best improvement that is currently possible. We denote the graphs in the family of counterexamples by $\mathcal{H}_n$, for $n \geqslant 1$. The lower bound construction for discrete strategy improvement with the switch-best rule is almost the same as for the switch-all rule. The differences in graphs $\mathcal{H}_n$ to $\mathcal{G}_n$ are the following.

- The cycles (d_i, e_i) are substituted by cycles $(d_i^1, d_i^2, d_i^3, e_i)$ where d_i^1 is also connected to the nodes s, c and a_{3j+3}, for $j \leqslant 2i-2$, d_i^2 to a_{3j+2}, for $j \leqslant 2i-2$, and d_i^3 to a_{3j+1}, for $j \leqslant 2i-1$.
- Every edge (g_i, f_i) is subdivided by an additional vertex y_i with an outgoing edge (y_i, k_i).
- Vertex c has only r as a direct successor.

Formally, the graph $\mathcal{H}_n = (V_n, E_n)$ is given by the set of vertices

$$V_n := \{x, s, c, r\} \cup \{a_i, t_i \colon 0 < i \leqslant 6n-2\} \cup \left\{d_i^1, d_i^2, d_i^3, e_i, f_i, h_i, g_i, y_i, k_i \colon 0 < i \leqslant n\right\}$$

and the edge set shown in Table 1. One can show that the graph $\mathcal{H}_n$ has size $O(n^2)$ [8].

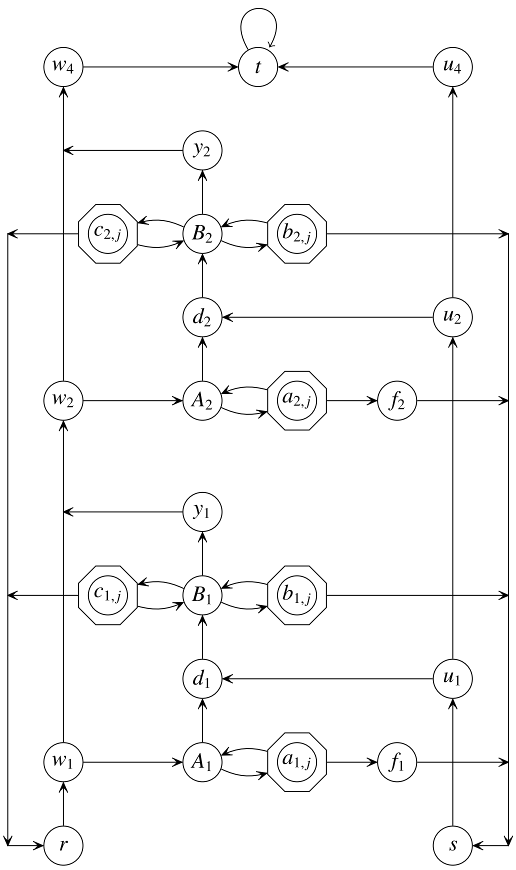
The values of the complexity measures for $\mathcal{H}_n$ we consider are very similar to those for $\mathcal{G}_n$ and so are the corresponding proofs.

The treewidth of $\mathcal{H}_n$ is not bounded by any finite constant, because it contains arbitrary complete bipartite graphs as the class $\mathcal{G}_n$. For the directed pathwidth, consider the following monotone winning strategy for four cops, which is the same for the DAG-width game and for the Kelly-width game. Two cops are placed on r and s and remain there for the rest of the play. Then, for $i = 1, 2, \ldots$, one cop is placed on e_i and the last cop visits d_i^1, d_i^2, d_i^3, a_i, h_i, g_i, y_i, k_i, f_i in that order. After visiting the n-th layer, the cops visit $t_n, t_{n-1}, \ldots, t_1$ and c and then x. It is clear that the strategy is winning and monotone.

In the entanglement game, two cops occupy r and s if the robber ever visits them and the remaining graph has entanglement one by Lemma 2. The construction of $\mathcal{H}_n$ for the cliquewidth is also very similar to the one of $\mathcal{G}_n$. We build the graph up layer by layer. However, we need four additional colours: two to distinguish three types of nodes a_i, $0 < i \leqslant 6n-2$, and two more to distinguish the nodes d_i^1, d_i^2 and d_i^3 from each other.

Thus we obtain the following theorem.

Theorem 10. *For all $n \geqslant 1$, we have* $\mathrm{tw}(\mathcal{H}_n) = \infty$, $\mathrm{dpw}(\mathcal{H}_n) = \mathrm{ent}(\mathcal{H}_n) = 3$, $\mathrm{dagw}(\mathcal{H}_n) = \mathrm{Kw}(\mathcal{H}_n) = 4$ *and* $\mathrm{cw}(\mathcal{H}_n) \leqslant 14$.



(a) The graph $\mathcal{G}_\zeta$ with two levels for the random-edge and switch-half rules. The meaning of octagons is explained in Figure 5

Vertex	Successors
t	t
r	w_1
s	u_1
w_i	w_{i+1}, A_i
u_i	u_{i+1}, d_i
w_{n+1}	t
u_{n+1}	t
A_i	$d_i, a_{i,h}$
$a_{i,1}$	A_i, f_i
$a_{i,j+1}$	$a_{i,j}, f_i$
B_i	$y_i, b_{i,l_i}, c_{i,g}$
$b_{i,1}$	B_i, s
$b_{i,j+1}$	$b_{i,j}, s$
$c_{i,1}$	B_i, r
$c_{i,j+1}$	$c_{i,j}, r$
f_i	s
d_i	B_i
y_i	w_{i+1}

(b) The edge relation of the graph $\mathcal{G}_\zeta$

Fig. 4. The graph $\mathcal{G}_\zeta$ for the random-edge and switch-half rules.

4.4. The random-edge and switch-half rules

The counterexample graphs for the random-edge and the switch-half rules are parametrised by a tuple of natural numbers $\zeta = (n, (l_i)_{0 \leqslant i \leqslant n}, h, g)$. The game graph $\mathcal{G}_\zeta$ is composed of n isomorphic levels. Level i consists of vertices

$$\{w_i, x_i, u_i, d_i, y_i, A_i, B_i, a_{i,j}, b_{i,l}, c_{i,k}\},$$

for $1 \leqslant j \leqslant h$, $1 \leqslant l \leqslant l_i$ and $1 \leqslant k \leqslant g$.

For $1 \leqslant i \leqslant n-1$, the i-th level is directly connected to the $i+1$-th level. The n-th level is connected to an additional top level that contains the vertices w_{n+1}, u_{n+1} and t. Furthermore, the graph has the two distinguished vertices r connected to w_1 and s connected to u_1. From each level, there are edges that lead to r and s.

More formally, for a tuple $\zeta = (n, (l_i)_{0 \leqslant i \leqslant n}, h, g)$, with $n, l_i, h, g > 0$, we define the graph $\mathcal{G}_\zeta = (V_\zeta, E_\zeta)$ by

$$\begin{aligned} V_\zeta := {} & \{a_{i,j} \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant h\} \cup \{b_{i,l} \mid 1 \leqslant i \leqslant n, 1 \leqslant l \leqslant l_i\} \\ & \cup \{c_{i,k} \mid 1 \leqslant i \leqslant n, 1 \leqslant k \leqslant g\} \cup \{d_i, y_i, x_i \mid 1 \leqslant i \leqslant n\} \\ & \cup \{w_i, u_i \mid 1 \leqslant i \leqslant n+1\} \cup \{t, r, s\} \cup \{A_i, B_i \mid 1 \leqslant i \leqslant n\} \end{aligned}$$

and E_ζ as given in Fig. 4b. The graphs with two levels are illustrated in Fig. 4a. The octagons in the figure symbolise the cycle gadgets, their interpretation is shown in Fig. 5. The vertex x represents s or r, V_i represents A_i or B_i and $k_{i,j}$ represents $a_{i,j}, b_{i,l}$ or $c_{i,k}$.

Theorem 11. *For all* ζ, $\mathrm{tw}(\mathcal{G}_\zeta) \leqslant 5$, $\mathrm{dpw}(\mathcal{G}_\zeta) = \mathrm{ent}(\mathcal{G}_\zeta) = 3$, $\mathrm{dagw}(\mathcal{G}_\zeta) = \mathrm{Kw}(\mathcal{G}_\zeta) = 4$ *and* $\mathrm{cw}(\mathcal{G}_\zeta) \leqslant 9$.

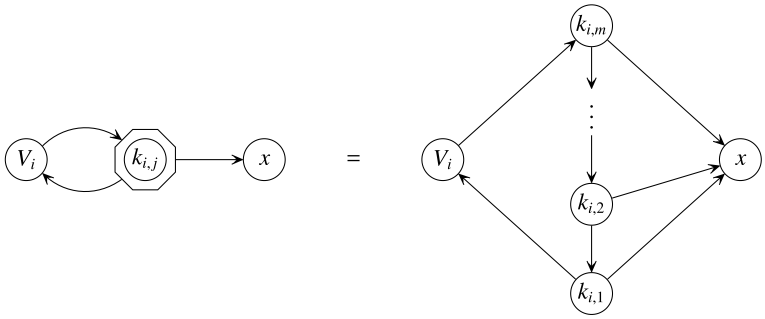


Fig. 5. Interpretation of the octagon.

Proof. *Treewidth*. Recall that we consider the undirected graph underlying $\mathcal{G}_\zeta$. We give a monotone winning strategy for 6 cops. At the beginning, two cops occupy r and s and remain there until the robber is captured. Then, for $i = 1, \dots, n-1$, the cops expel the robber from level i and simultaneously block paths to level $i-1$ (for $i > 1$). By induction on i, we show that the cops can expel the robber from level i such that, at the end of the round, they occupy r, s, w_i and u_i.

The proof of the induction base is the same as for induction step, so assume some $i \geqslant 1$ and if $i > 1$, assume that the cops occupy w_{i-1} and u_{i-1}. Thus, we have two more cops free. Place them on w_i and u_i and remove the cops from w_{i-1} and u_{i-1}. Then place a new cop on A_i. If the robber chooses the component induced by $a_{i,j}$ (for $1 \leqslant j \leqslant h$) and f_i, he is captured by cops from r, w_i and u_i, so assume that the robber goes to d_i. The last free cop occupies d_i, then the cops from A_i occupies B_i. If the robber chooses the component induced by $c_{i,l}$ or the component induced by $b_{i,k}$, he is captured there by the cops from w_i and from d_i, so assume that he goes to y_i. Then the cops from d_i go to y_i too and the induction is completed. Then the cops capture the robber on t.

For the other direction (that $\mathrm{tw}(\mathcal{G}_\zeta) \geqslant 6$), note that the graph consists, in essence, of the path $w_1, \dots, w_n$, the path $u_1, \dots, u_n$ and vertices r and s. Hereby, for each i, w_i is connected to u_i (via A_i and d_i), all w_i are connected to r and all u_i are connected to s. For $n \geqslant 3$, one needs 4 cops to capture the robber on the two paths [20]. Clearly, r and s must be occupied during all the play in order to ensure monotonicity.

Directed pathwidth. Note that all paths from level $i+1$ to level i lead through s or r. The strategy for 4 cops is as follows. Two cops always occupy r and s. For level i, one cop stays on A_i and the remaining one visits w_i $a_{i,h}, \dots, a_{i,1}$, f_i, d_i and B_i in tat order. Then the cops from A_i visits $c_{i,k}, \dots, c_{i,1}$, $b_{i,l_i}, \dots, b_{i,1}$ and h_i in that order. Proceeding in that way, the cops clear all levels and finally capture the robber on t.

For the other direction, note that two cops cannot clear any cycle alone. Let $n > 1$. Assume that there are only 3 cops. Let i be the number of the level where the first cop is placed. Then one needs at least two cops to block paths from another level to level i (via w_{i-1} and u_{i-1} or via r and s). Even if the two remaining cops are placed on w_i and u_i (i.e. in the level), one cop is not able to expel the robber from level i. The proofs for *DAG-width* and for *Kelly-width* are, essentially, the same.

Entanglement. The winning strategy for 3 cops is almost the same as for 4 cops in the directed pathwidth game. It uses the rule that the robber must change his vertex in every move. Note that once r and s are occupied, every strongly connected component of the remaining graph contains a vertex whose removal makes it acyclic. If there are only two cops and $n \geqslant 3$, the robber has the following winning strategy. Note that every level contains a cycle, so the robber can stay in a cop free level indefinitely. Furthermore, there are pairwise disjoint paths from each cycle to r and to s. As there are two cops, one cycle contains no cops. The robber can hold the invariant that he can always reach a cop free cycle. Assume that he is waiting in a cop free cycle. When a cop comes, r or s is not occupied by cops, so there is a cop free path to an other cop free cycle.

Cliquewidth. Vertices r and s have there own colours. We construct the graph by induction on the number of levels. Level 1 is constructed as all other levels. In general, assume that the graph is constructed up to level i such that r, s, w_i, u_i and y_i have unique colours and all other vertices have colour *Old*. Thus 3 colours are free for further use. We refer to a recolouring of a vertex to *Old* as to forgetting its colour. First, create vertices w_{i+1} and u_{i+1} with a new colour (1 free colour), connect w_i and y_i to w_{i+1} and u_i to u_{i+1} and forget the colours of w_i, y_i and u_i (4 colours are free). Construct A_{i+1} with a new colour (3 free colours) and the path $a_{i+1,k}, \dots, a_{i+1,1}$ connecting A_{i+1} to $a_{i_1,k}$ at the beginning and $a_{i+1,1}$ to A_{i+1} at the end using 3 more colours (0 free colours). After that, all $a_{i+1,j}$ have the same colour; one of their three colours is used to create f_{i+1} and all $a_{i+1,j}$ are connected to f_{i+1}, which is connected to s. Then the colours of all $a_{i_1,j}$ and f_{i+1} are forgotten (again 3 free colours).

Now, d_{i+1} is created with a new colour, A_{i+1} and u_{i+1} are connected to it and the colours of A_{i+1} and u_{i+1} are forgotten (4 free colours). Create B_{i+1} with a new colour, connect d_{i+1} to it and forget the colour of d_{i+1}. The octagons with B_{i+1} are constructed as the octagon with A_{i+1}, and y_{i+1} is created with a new colour and the colours of B_{i+1} and all $b_{i+1,j}$

and $c_{i+1,j}$ are forgotten. It is clear that now r, s, w_{i+1}, u_{i+1} and y_{i+1} have unique colours and all other vertices have colour *Old*, so the induction step is proved. □

4.5. The random-facet rule

The improvement rule *random-facet* is a randomised recursive optimisation rule. The rule maintains a set E_0 of edges that can be chosen by a player 0 strategy σ such that $\sigma(v) = w$ implies $(v, w) \in E_0$. Initially, an arbitrary strategy for player 0 is taken. Then the algorithm computes recursively which edges can be removed from E_0 as follows. If E_0 is deterministic (i.e. for every vertex v exists only one vertex w with $(v, w) \in E_0$), then σ is already optimal. Otherwise, let $e = (v, w)$ be some edge from E_0 with $w \neq \sigma(v)$ chosen uniformly at random. Then, recursively, an optimal strategy σ' in the game without the edge e is computed. If e is not an improving switch for σ', then σ' is an optimal strategy in the original game. Otherwise, the switch to e is performed. We refer to [17,8] for more details.

The counterexample graphs are parametrised by triples (n, s, r) of natural numbers. The vertex set of the graph $\mathcal{G}_{n,s,r} = (V_{n,s,r}, E_{n,s,r})$ is given by

$$\begin{aligned} V_{n,s,r} := {} & \left\{a_i^j \,\middle|\, 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant sr\right\} \cup \left\{b_i^j \,\middle|\, 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant sr\right\} \\ & \cup \{C_i \mid 1 \leqslant i \leqslant n\} \cup \left\{A_i^j \,\middle|\, 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant l\right\} \cup \{B_i \mid 1 \leqslant i \leqslant n\} \\ & \cup \{D_i \mid 1 \leqslant i \leqslant n\} \cup \{T\}. \end{aligned}$$

Fig. 6b defines the edge relation $E_{n,s,r}$. The graph $\mathcal{G}_{n,s,r}$ contains n isomorphic levels. The i-th level is the subgraph that is induced by the vertices a_i^j, b_i^j, c_i, A_i^j, B_i, D_i. Level i is connected to level $i+1$. Level n is connected to the vertex T.

Theorem 12. *For all $n > 1$, and all $s, r > 0$, we have that* $\operatorname{tw}(\mathcal{G}_{n,s,r}) = 3$, $\operatorname{dpw}(\mathcal{G}_{n,s,r}) = 1$, $\operatorname{dagw}(\mathcal{G}_{n,s,r}) = \operatorname{Kw}(\mathcal{G}_{n,s,r}) = 2$, $\operatorname{ent}(\mathcal{G}_{n,s,r}) = 1$, $\operatorname{cw}(\mathcal{G}_{n,s,r}) \leqslant 5$.

Proof. The winning strategy for 2 cops in the directed pathwidth, the DAG-width and the Kelly-width games is, for all layers $i = 1, 2, \ldots, n$ as follows. For $j = 1, 2, \ldots, s$, one cop occupies A_i^j and the other cop all a_i^k, for $r \cdot (j-1) + 1 \leqslant k \leqslant r \cdot j$, connected with the current $A_{i,j}$. Then one cop visits D_i and after that B_i and the other one all b_i^j. Finally, the robber is captured in T. As the graph contains cycles, one cop is not able to capture the robber. For the entanglement, the proof is similar. For the treewidth, note that two cops on C_i and b_{i+1}^1 isolate level i; the two remaining cops catch the robber in a level. On the other hand, it is easy to see that the robber can escape three cops. The proof for the cliquewidth is similar to the proofs for other graphs and we leave it as an exercise for the reader. □

4.6. The least-recently-considered rule

Cunningham's *least-recently-considered* or *round-robin* rule [6] is a deterministic, memorising improvement rule. It fixes an initial ordering on all edges first, and then selects the improving switches in a round-robin fashion. We define the graph $\mathcal{R}_n = (V_n, E_n)$ underlying the corresponding parity game. The set of vertices is given by

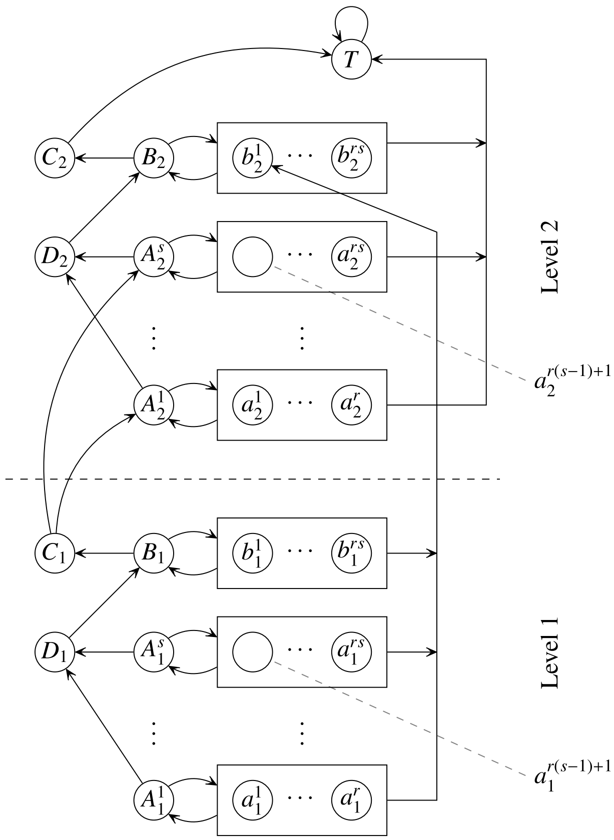
$$\begin{aligned} V_n := {} & \{b_{i,j} \mid 1 \leqslant j \leqslant i \leqslant n\} \cup \{y_i, d_i \mid 1 \leqslant i \leqslant n\} \cup \{u_i, w_i \mid 1 \leqslant i \leqslant n+1\} \\ & \cup \{t, s\} \cup \{B_i \mid 1 \leqslant i \leqslant n\} \end{aligned}$$

and the set of edges is shown in Fig. 7b. See also Fig. 7a for an illustration.

The graph $\mathcal{R}_n$ consists of n very similar levels. The i-th level is the subgraph induced by the vertices $w_i, d_i, u_i, y_i, B_i, b_{i,1}, \ldots, b_{i,i}$. The difference between the levels is the length of the cycle $B_i, b_{i,1}, \ldots, b_{i,i}$, which grows from level to level by one. The i-th level is directly connected to the $(i+1)$-th level by several edges. Each level is connected to a distinguished vertex s, whose only successor is the vertex d_1 in the first level.

Theorem 13. *For all $n \geqslant 3$,* $\operatorname{tw}(\mathcal{R}_n) = \operatorname{dpw}(\mathcal{R}_n) = 3$, $\operatorname{dagw}(\mathcal{R}_n) = \operatorname{Kw}(\mathcal{R}_n) = 4$, $\operatorname{ent}(\mathcal{R}_n) = 4$, $\operatorname{cw}(\mathcal{R}_n) \leqslant 7$.

Proof. A winning strategy for 5 cops in the treewidth game is to block s and u_1, and to go from level n to level 1 finishing level i with a cop on w_i and a cop on u_i. The step from level i to level $i-1$ is to place a cop on u_{i-1}, then move the cop from u_i to d_{i-1}. If the robber is in $\{B_{i-1}, y_{i-1}, b_{i-1,1}, b_{i-1,2}\}$, move the cop from u_{i-1} to B_{i_1} and then either from d_{i-1} to y_{i-1} (if the robber is on y_{i-1}) or from d_{i-1} and w_i to $b_{i-1,1}$ and $b_{i-1,2}$ (if the robber is there). On the other hand, its is easy to see that on the minor induced by w_3, u_3, d_3, w_4 and u_4 (i.e. where w_3 is connected to u_3, d_3 and w_4; w_4 to d_3 and u_4; u_4 to u_3; u_3 to d_3) three cops are needed. As the degrees of s and u_1 are at least 7, they must be occupied during all the game. The statements about the other measures can be proven analogously to the previous cases. □



(a) The graph $\mathcal{G}_{n,s,r}$

Vertex	Successors
$a_i^{k\cdot r+j}$ (for $j < r$ and $i < n$)	A_i^k, b_{i+1}^1
$a_n^{k\cdot r+j}$ (for $j < r$)	A_n^k, T
b_i^j	B_i, b_{i+1}^1
b_n^j	B_n, T
T	T
C_i	$\{A_{i+1}^j \mid 1 \le j \le r\}$
C_n	T
A_i^k	$D_i, \{a_i^{k\cdot r+j} \mid j < r\}$
B_i	C_i, b_i^j
D_i	B_i

(b) The edge relation of $\mathcal{G}_{n,s,r}$

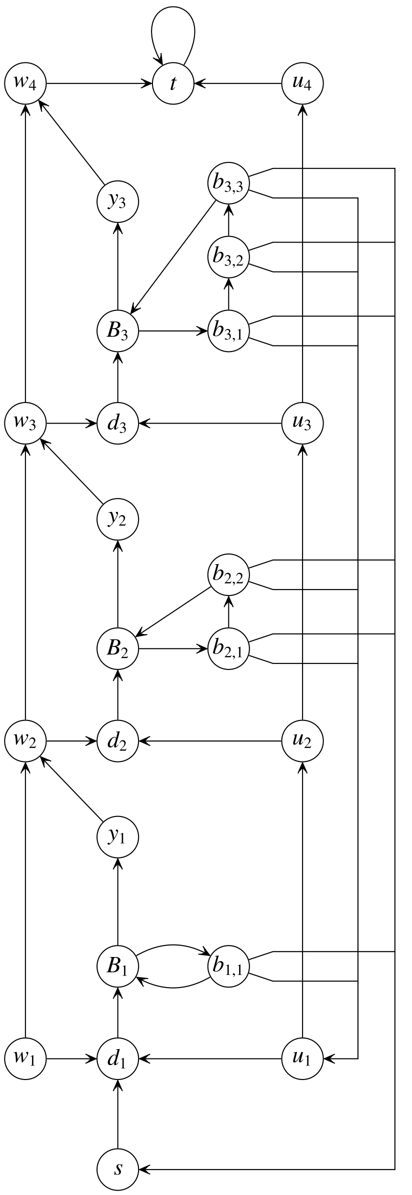
Fig. 6. The graph $\mathcal{G}_{3,s,r}$ for the random-facet rule.

4.7. Strategy improvement based on snare memorisation

The graph $\mathcal{U}_n = (V_n, E_n)$ underlying Friedmann's counterexample [9] for the snare memorisation technique of Fearnley [7] has vertex set

$$V_n := \{x, s, c, r\} \cup \{t_i, a_i \mid 1 \leqslant i \leqslant 3n\} \cup \{b_i, d_i, e_i, g_i, k_i, f_i, h_i \mid 1 \leqslant i \leqslant n\}$$
$$\cup \{q_i, m_i \mid 1 \leqslant i \leqslant n\} \cup \{u_{i,j}, v_{i,j}, w_{i,j} \mid 1 \leqslant i < j \leqslant n\}$$

and the set of edges as shown in Table 2. The graph $\mathcal{U}_n$ is similar to the graph $\mathcal{G}_n$, the counterexample graph for the switch-all rule. The basic structure is the same, but each level is extended in order to render the memorisation ineffective. The treewidth of the graphs $\mathcal{U}_n$ is not bounded, the other measures are the same for $\mathcal{U}_n$ and $\mathcal{G}_n$, and the same proofs apply.



(a) The lower bound game $\mathcal{R}_3$ for the least-recently-considered rule

Vertex	Successors
u_{n+1}	t
$u_{i\leq n}$	d_i, u_{i+1}
w_{n+1}	t
$w_{i\leq n}$	d_i, w_{i+1}
$b_{i,j}$	$b_{i,j-1}, s, u_1$
t	t
s	d_1
B_i	$b_{i,i}, y_i$
y_i	w_{i+1}
d_i	B_i

(b) The edge relation of $\mathcal{R}_n$

Fig. 7. The counterexamples for the least-recently-considered rule.

Table 2
Edges of $\mathcal{U}_n$ (snare).

Vertex	Successors	Vertex	Successors
t_1	$\{s, r, c\}$	h_i	$\{k_i\}$
$t_{i>1}$	$\{s, r, t_{i-1}\}$	s	$\{x\} \cup \{f_j \mid j \leqslant n\}$
a_i	$\{t_i\}$	r	$\{x\} \cup \{g_j \mid j \leqslant n\}$
c	$\{s, r\}$	x	$\{x\}$
d_i	$\{s, r, u_{i,n}\} \cup \{a_j \mid j \leqslant 3i\}$	$u_{i,j}$	$\{v_{i,j}, w_{i,j}\}$
e_i	$\{b_i, h_i\}$	$v_{i,j}$	$\{u_{i,j-1}, m_j\}$
g_i	$\{f_i, k_i\}$	$w_{i,j}$	$\{u_{i,j-1}, q_j\}$
k_i	$\{x\} \cup \{g_j \mid i < j \leqslant n\}$	m_i	$\{e_i\}$
f_i	$\{e_i\}$	q_i	$\{h_i\}$
b_i	$\{s, r, d_i\} \cup \{a_j \mid j \leqslant 3i\}$		

Table 3
Upper bounds in different measures for the counterexample graph classes.

Rule	Measure					
	Tree	Directed path	DAG	Kelly	Entanglement	Clique
Switch-all	∞	3	4	4	3	$\leqslant 10$
Least-entered	∞	∞	∞	∞	∞	$\leqslant 9$
Switch-best	∞	3	4	4	3	$\leqslant 14$
Random-edge	6	3	4	4	3	$\leqslant 9$
Random-facet	3	1	2	2	1	$\leqslant 5$
Least-considered	4	3	4	4	4	$\leqslant 7$
Snare memory	∞	3	4	4	4	?

We conjecture that the cliquewidth of the graphs $\mathcal{U}_n$ is not bounded. The new elements (compared to $\mathcal{G}_n$) induce a grid-similar structure and it should be possible to show the unboundedness similarly to the case of square grids [11].

References

[1] K. Apt, E. Grädel (Eds.), Lectures in Game Theory for Computer Scientists, Cambridge University Press, 2011.
[2] D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, J. Obdržálek, The DAG-width of directed graphs, J. Comb. Theory, Ser. B 102 (4) (July 2012) 900–923, http://dx.doi.org/10.1016/j.jctb.2012.04.004.
[3] D. Berwanger, E. Grädel, Entanglement – a measure for the complexity of directed graphs with applications to logic and games, in: Proceedings of LPAR 2004, Montevideo, Uruguay, in: Lect. Notes Comput. Sci., vol. 3452, Springer-Verlag, 2005, pp. 209–223, http://www.logic.rwth-aachen.de/pub/dwb/BG_lpar04.pdf.
[4] D. Berwanger, E. Grädel, Ł. Kaiser, R. Rabinovich, Entanglement and the complexity of directed graphs, Theor. Comput. Sci. 463 (2012) 2–25, http://logic.rwth-aachen.de/~rabinovich/entangle.pdf.
[5] B. Courcelle, J. Engelfriet, G. Rozenberg, Handle-rewriting hypergraph grammars, J. Comput. Syst. Sci. 46 (2) (1993) 218–270.
[6] W.H. Cunningham, Theoretical Properties of the Network Simplex Method, Carleton Math. Ser., Department of Mathematics, Carleton University, 1977, http://books.google.de/books?id=vZY3HAAACAAJ.
[7] J. Fearnley, Non-oblivious strategy improvement, in: E.M. Clarke, A. Voronkov (Eds.), LPAR (Dakar), in: Lect. Notes Comput. Sci., vol. 6355, Springer, 2010, pp. 212–230.
[8] O. Friedmann, Exponential lower bounds for solving infinitary payoff games and linear programs, Ph.D. thesis, Ludwig-Maximilians-University Munich, 2011.
[9] O. Friedmann, A superpolynomial lower bound for strategy iteration based on snare memorization, Discrete Appl. Math. 161 (10–11) (July 2013) 1317–1337, http://dx.doi.org/10.1016/j.dam.2013.02.007.
[10] O. Friedmann, A subexponential lower bound for the Least Recently Considered rule for solving linear programs and games, in: The Annual Workshop of the ESF Networking Programme on Games for Design and Verification, GAMES'2012, Napoli, Italy, 2012.
[11] M.C. Golumbic, U. Rotics, On the clique-width of some perfect graph classes, Int. J. Found. Comput. Sci. 11 (3) (2000) 423–443.
[12] P. Hunter, S. Kreutzer, Digraph measures: Kelly decompositions, games, and orderings, Theor. Comput. Sci. 399 (3) (2008) 206–219.
[13] M. Jurdziński, Deciding the winner in parity games is in UP ∩ co-UP, Inf. Process. Lett. 68 (3) (1998) 119–124.
[14] M. Jurdziński, Small progress measures for solving parity games, in: H. Reichel, S. Tison (Eds.), STACS, in: Lect. Notes Comput. Sci., vol. 1770, Springer, 2000, pp. 290–301.
[15] M. Jurdziński, M. Paterson, U. Zwick, A deterministic subexponential algorithm for solving parity games, in: SODA, ACM Press, 2006, pp. 117–123.
[16] M. Jurdziński, J. Vöge, A discrete strategy improvement algorithm for solving parity games (Extended abstract), in: E.A. Emerson, A.P. Sistla (Eds.), Computer Aided Verification, 12th International Conference, CAV 2000, Proceedings, in: Lect. Notes Comput. Sci., vol. 1855, Springer-Verlag, Chicago, IL, USA, 2000, pp. 202–215.
[17] J. Matousek, M. Sharir, E. Welzl, A subexponential bound for linear programming, Algorithmica 16 (4/5) (1996) 498–516.
[18] J. Obdržálek, Fast mu-calculus model checking when tree-width is bounded, in: W.A. Hunt Jr., F. Somenzi (Eds.), CAV, in: Lect. Notes Comput. Sci., vol. 2725, Springer, 2003, pp. 80–92.
[19] J. Obdržálek, Clique-width and parity games, in: J. Duparc, T.A. Henzinger (Eds.), CSL'07, in: Lect. Notes Comput. Sci., vol. 4646, Springer, 2007, pp. 54–68.
[20] R. Rabinovich, Graph complexity measures and monotonicity, Ph.D. thesis, RWTH Aachen University, 2013, http://logic.rwth-aachen.de/~rabinovich/diss_rabinovich_07.12.2012.pdf.
[21] S. Schewe, An optimal strategy improvement algorithm for solving parity and payoff games, in: M. Kaminski, S. Martini (Eds.), CSL, in: Lect. Notes Comput. Sci., vol. 5213, Springer, 2008, pp. 369–384.