DAG-Width and Parity Games

Dietmar Berwanger[1], Anuj Dawar[2], Paul Hunter[3], and Stephan Kreutzer[3]

[1] LaBRI, Université de Bordeaux 1, dwb@labri.fr
[2] University of Cambridge Computer Laboratory, anuj.dawar@cl.cam.ac.uk
[3] Logic and Discrete Systems, Institute for Computer Science, Humboldt-University Berlin, {hunter, kreutzer}@informatik.hu-berlin.de

Abstract. Tree-width is a well-known metric on undirected graphs that measures how tree-like a graph is and gives a notion of graph decomposition that proves useful in algorithm development. Tree-width is characterised by a game known as the cops-and-robber game where a number of cops chase a robber on the graph. We consider the natural adaptation of this game to directed graphs and show that monotone strategies in the game yield a measure with an associated notion of graph decomposition that can be seen to describe how close a directed graph is to a directed acyclic graph (DAG). This promises to be useful in developing algorithms on directed graphs. In particular, we show that the problem of determining the winner of a parity game is solvable in polynomial time on graphs of bounded DAG-width. We also consider the relationship between DAG-width and other measures of such as entanglement and directed tree-width. One consequence we obtain is that certain NP-complete problems such as Hamiltonicity and disjoint paths are polynomial-time computable on graphs of bounded DAG-width.

1 Introduction

The groundbreaking work of Robertson and Seymour in their graph minor project has focused much attention on tree-decompositions of graphs and associated measures of graph connectivity such as tree-width [13]. Aside from their interest in graph structure theory, these notions have also proved very useful in the development of algorithms. The tree-width of a graph is a measure of how tree-like the graph is, and it is found that small tree-width allows for graph decompositions along which recursive algorithms can work. Many problems that are intractable in general can be solved efficiently on graphs of bounded tree-width. These include such classical NP-complete problems as finding a Hamiltonian cycle in a graph or detecting if a graph is three-colourable. Indeed, a general result of Courcelle [4] shows that any property definable in monadic second-order logic is solvable in linear time on graphs of fixed tree-width.

The idea of designing algorithms that work on tree-decompositions of the input has been generalised from graphs to other kinds of structures. Usually the tree-width of a structure is defined as that of the underlying connectivity (or Gaifman) graph. For instance, the tree-width of a directed graph is simply that of the undirected graph we get by forgetting the direction of edges, a process which leads to some loss of information. This loss may be significant if the algorithmic problems we are interested in are inherently directed. A good example is the problem of detecting Hamiltonian cycles. While

[0] 23rd International Symposium on Theoretical Aspects of Computer Science (STACS), 2006

we know that this can be solved easily on graphs with small tree-width, there are also directed graphs with very simple connectivity structure which have large tree-width. A directed acyclic graph (DAG) is a particularly simple structure, but we lose sight of this when we erase the direction on the edges and find the underlying undirected graph to be dense. Several proposals have been made (see [12, 8, 2, 14]) which extend notions of tree-decompositions and tree-width to directed graphs. In particular, Johnson et al. [8] introduce the notion of *directed tree-width* where directed acyclic graphs have width 0 and they show that Hamiltonicity can be solved for graphs of bounded directed tree-width in polynomial time. However, the definition and characterisations of this measure are somewhat unwieldy and they have not, so far, resulted in many further developments in algorithms.

We are especially interested in one particular problem on directed graphs, that of determining the winner of a *parity game*. This is an infinite two-player game played on a directed graph where the nodes are labelled by priorities. The players take turns pushing a token along edges of the graph. The winner is determined by the parity of the least priority occurring infinitely often in this infinite play. Parity games have proved useful in the development of model-checking algorithms used in the verification of concurrent systems. The modal μ-calculus, introduced in [10], is a widely used logic for the specification of such systems, encompassing a variety of modal and temporal logics. The problem of determining, given a system $\mathcal{A}$ and a formula φ of the μ-calculus, whether or not $\mathcal{A}$ satisfies φ can be turned into a parity game (see [6]). The exact complexity of solving parity games is an open problem that has received a large amount of attention. It is known [9] that the problem is in NP $\cap$ co-NP and no polynomial time algorithm is known. It follows from the general result of Courcelle [4] that there is a polynomial time algorithm that solves parity games on graphs of bounded tree-width. Obdržàlek [11] exhibited a particular such algorithm. He points out that the algorithm would not give good bounds, for instance, on directed acyclic graphs even though solving the games on such graphs is easy. He asks whether there is a structural property of directed graphs that would allow a fast algorithm on both bounded tree-width structures and on DAGs.

In this paper, we give just such a generalisation. We introduce a new measure of the connectivity of graphs that we call DAG-width[4]. It is intermediate between tree-width and directed tree-width, in that for any graph $\mathcal{G}$, the directed tree-width of $\mathcal{G}$ is no greater than its DAG-width which, in turn, is no greater than its tree-width. Thus, the class of structures of DAG-width $k + 1$ or less includes all structures of tree-width k and more (in particular, DAGs of arbitrarily high tree-width all have DAG-width 1).

The notion of DAG-width can be understood as a simple adaptation of the game of *cops and robber* (which characterises tree-width) to directed graphs. The game is played by two players, one of whom controls a set of k cops attempting to catch a robber controlled by the other player. The cop player can move any set of cops to any nodes on the graph, while the robber can move along any path in the graph as long as there is no cop currently on the path. Such games have been extensively studied (see [15, 5, 7, 1, 2]). It is known [15] that the cop player has a winning strategy on an undirected

[4] We understand that Obdržàlek has defined a similar measure in a paper to appear at SODA'06. We have not yet had an opportunity to see that paper.

graph $\mathcal{G}$ using $k+1$ cops if, and only if, $\mathcal{G}$ has tree-width k. We consider the natural adaptation of this game to directed graphs, by constraining the robber to move along directed paths. We show that the class of directed graphs where there is a monotone (in a sense we make precise) strategy for k cops to win is characterised by its width in a decomposition that is a generalisation of tree-decompositions. We are then able to show that the problem of determining the winner of a parity game is solvable in polynomial time on the class of graphs of DAG-width k, for any fixed k.

In Section 2, we introduce some notation. Section 3 introduces the cops and robber game, DAG-decompositions and DAG-width and shows the equivalence between the existence of monotone winning strategies and DAG-width. Also in Section 3 we discuss some algorithmic aspects of DAG-width. Section 4 relates DAG-width to other measures of graph connectivity, and Section 5 demonstrates a polynomial time algorithm for solving parity games on graphs with bounded DAG-width. All proofs appear in the full version of the paper.

2 Preliminaries

We first fix some notation used throughout the paper. All graphs used are finite, directed and simple unless otherwise stated.

We write ω for the set of finite ordinals, i.e. natural numbers. For every $n \in \omega$, we write $[n]$ for the set $\{1, \ldots, n\}$. For every set V and every $k \in \omega$, we write $[V]^k$ for the set of all k-element subsets of V, that is, $[V]^k := \{\{x_1, \ldots, x_k\} \subseteq V : x_i \neq x_j$ whenever $i \neq j\}$. We write $[V]^{\leq k}$ for the set of all $X \subseteq V$ with $|X| \leq k$.

Let $\mathcal{G}$ be a directed graph. We write $V^{\mathcal{G}}$ for the set of its vertices and $E^{\mathcal{G}}$ for the set of its edges. E^{op} denotes the set of edges that results from reversing the edges in $E \subseteq E^{\mathcal{G}}$, i.e. $E^{op} = \{(w, v) : (v, w) \in E\}$. The graph $\mathcal{G}^{op}$ is defined to be $(V^{\mathcal{G}}, (E^{\mathcal{G}})^{op})$.

A tree-decomposition of a graph $\mathcal{G}$ is a labelled tree $(\mathcal{T}, (X_t)_{t \in V^{\mathcal{T}}})$ where $X_t \subseteq V^{\mathcal{G}}$ for each vertex $t \in V^{\mathcal{T}}$, for each edge $(u, v) \in E^{\mathcal{G}}$ there is a $t \in V^{\mathcal{T}}$ such that $\{u, v\} \subseteq X_t$, and for each $v \in V^{\mathcal{G}}$, the set $\{t \in V^{\mathcal{T}} : v \in X_t\}$ forms a connected subtree of $\mathcal{T}$. The *width* of a tree-decomposition is the cardinality of the largest X_t minus one. The tree-width of $\mathcal{G}$ is the smallest k such that $\mathcal{G}$ has a tree-decomposition of width k.

Let $\mathcal{D} := (D, A)$ be a directed, acyclic graph (DAG). The partial order $\preceq_{\mathcal{D}}$ (or $\preceq_A$) on D is the reflexive, transitive closure of A. A *root* of a set $X \subseteq D$ is a $\preceq_{\mathcal{D}}$-minimal element of X, that is, $r \in X$ is a root of X if there is no $y \in X$ such that $y \preceq_{\mathcal{D}} r$. Analogously, a *leaf* of $X \subseteq D$ is a $\preceq_{\mathcal{D}}$-maximal element.

3 Games, Strategies and Decompositions

This section contains the graph theoretical part of this paper. We define DAG-width and its relation to graph searching games. As mentioned in the introduction, the notion of tree-width has a natural characterisation in terms of a cops and robber game. Directed tree-width has also been characterised in terms of such games [8], but these games appear to be less intuitive. In this paper, we consider the straightforward extension of the cops and robber game to directed graphs. We show that these games give a characterisation of the graph connectivity measure that we call DAG-width and introduce in Section 3.2. We comment on algorithmic properties in Section 3.3.

3.1 Cops and Robber Games

The Game. The Cops and Robber game on a digraph is a game where k cops try to catch a robber who may run along paths in the digraph. While the robber is confined to moving along paths in the graph, the cops may move to any vertex at any time. A formal definition follows.

Definition 3.1 (Cops and Robber Game)**.** Given a graph $\mathcal{G} := (V, E)$, the *k-cops and robber* game on $\mathcal{G}$ is played between two players, the *cop* and the *robber* player, as follows:

- At the beginning, the cop player chooses $X_0 \in [V]^{\leq k}$, and the robber player chooses a vertex r_0 of $V \setminus X_0$, giving position (X_0, r_0).
- From position (X_i, r_i), the cop player chooses $X_{i+1} \in [V]^{\leq k}$, and the robber player chooses a vertex r_{i+1} of $V \setminus X_{i+1}$ such that there is a path from r_i to r_{i+1} which does not pass through a vertex in $X_i \cap X_{i+1}$. If no such vertex exists then the robber player loses.

A *play* in the game is a (finite or infinite) sequence $\pi := (X_0, r_0)(X_1, r_1)\ldots$ of positions such that the transition from (X_i, r_i) to (X_{i+1}, r_{r+1}) is a valid move by the rules above and such that the play is finite if, and only if, $r_n \in X_n$ for the final position (X_n, r_n). A play is winning for the robber player if it is infinite.

As always when dealing with games we are less interested in a single play in the game as in strategies that allow a player to win every play in the game. Winning strategies for the cop player play a crucial role throughout this paper. We therefore give a precise definition of this notion.

Definition 3.2. Let $\mathcal{G} := (V, E)$ be a directed graph. A *(k-cop) strategy* for the cop player is a function f from $[V]^{\leq k} \times V$ to $[V]^{\leq k}$. A play $(X_0, r_0), (X_1, r_1), \ldots$ is *consistent* with a strategy f if $X_{i+1} = f(X_i, r_i)$ for all i. The strategy f is called a *winning strategy*, if every play consistent with f is winning for the cop player.

Definition 3.3 (Game-width)**.** The *game-width gw*$(\mathcal{G})$ of $\mathcal{G}$ is the least k such that the cop player has a strategy to win the k-cops and robber game on $\mathcal{G}$.

Variants of the game where the robber moves first or only one cop can be moved at a time or the cops are lifted and placed in separate moves are all equivalent in that the game-width of a graph does not depend on the variant.

Lemma 3.4. *For every finite, non-empty, directed graph $\mathcal{G}$ the game-width gw$(\mathcal{G})$ is at least one and gw$(\mathcal{G}) = 1$ if, and only if, $\mathcal{G}$ is acyclic.*

Games similar to the one defined above have been used to give game characterisations of concepts like undirected tree-width [15] and also the directed tree-width of [8]. Directed tree-width is invariant under reversing the edges of a graph. As we see below, this is not true of the game-width we have defined. One exception are graphs of game-width 1, i.e. acyclic graphs.

Proposition 3.5. *$gw(\mathcal{G}) = 1$ if, and only if $gw(\mathcal{G}^{op}) = 1$.*

Proposition 3.6. *For any j, k with $2 \leq j \leq k$, there exists a graph $\mathcal{T}_k^j$ such that $gw(\mathcal{T}_k^j) = j$ and $gw((\mathcal{T}_k^j)^{op}) = k$.*

In the sequel we consider a restriction of the cop player to monotone strategies.

Definition 3.7 (Monotone strategy)**.**

(i) A strategy for the cop player is *cop-monotone* if in playing the strategy, no vertex is visited twice by cops. That is, if $(X_0, r_0), (X_1, r_1) \dots$ is a play consistent with the strategy, then for every $0 \leq i < n$ and $v \in X_i \setminus X_{i+1}$, $v \notin X_j$ for all $j > i$.
(ii) A strategy for the cop player is *robber-monotone* if in playing the strategy, the set of vertices reachable by the robber is non-increasing.

Lemma 3.8. *If the cop player has a cop-monotone or robber-monotone winning strategy then they also have a winning strategy that is both, cop- and robber-monotone.*

From this lemma we can define a *monotone winning strategy* in the obvious way.

3.2 DAG-Decompositions and DAG-Width

In this section we define the notion of DAG-width which measures how close a given graph is to being acyclic. We present a decomposition of directed graphs that is somewhat similar in style to tree-decompositions of undirected graphs. We show then that a graph has DAG-width k if, and only if, the cop player has a monotone winning strategy in the k-cops and robber game played on that graph. We conclude with some properties enjoyed by DAG-width.

Definition 3.9. Let $\mathcal{G} := (V, E)$ be a graph. A set $W \subseteq V$ *guards* a set $V' \subseteq V$ if whenever there is an edge $(u, v) \in E$ such that $u \in V'$ and $v \notin V'$ then $v \in W$.

Definition 3.10 (DAG-decomposition)**.** Let $\mathcal{G} := (V, E)$ be a directed graph. A DAG-*decomposition* is a tuple $\mathfrak{D} = (\mathcal{D}, (X_d)_{d \in V^{\mathcal{D}}})$ such that

(D1) $\mathcal{D}$ is a DAG.
(D2) $\bigcup_{d \in V^{\mathcal{D}}} X_d = V$.
(D3) For all $d \preceq_{\mathcal{D}} d' \preceq_{\mathcal{D}} d''$, $X_d \cap X_{d''} \subseteq X_{d'}$.
(D4) For a root d, $\mathcal{X}_d$ is guarded by $\varnothing$.
(D5) For all $(d, d') \in E^{\mathcal{D}}$, $X_d \cap X_{d'}$ guards $\mathcal{X}_{d'} \setminus X_d$, where $\mathcal{X}_{d'} := \bigcup_{d' \preceq_{\mathcal{D}} d''} X_{d''}$.

The width of $\mathfrak{D}$ is defined as $\max\{|X_d| : d \in V^{\mathcal{D}}\}$. The DAG-*width* of a graph is defined as the minimal width of any of its DAG-decompositions.

The main result of this section is an equivalence between monotone strategies for the cop player and DAG-decompositions.

Theorem 3.11. *For any graph $\mathcal{G}$ there is a* DAG-*decomposition of $\mathcal{G}$ of width k if, and only if, the cop player has a monotone winning strategy in the k-cops and robber game on $\mathcal{G}$.*

For algorithmic purposes, it is often useful to have a normal form for decompositions. The following is similar to one for tree-decompositions as presented in [3].

Definition 3.12. A DAG-decomposition $(\mathcal{D}, (X_d)_{d \in V^{\mathcal{D}}})$ is *nice* if

(N1) $\mathcal{D}$ has a unique root.
(N2) Every $d \in V^{\mathcal{D}}$ has at most two successors.
(N3) If d_1, d_2 are two successors of d_0, then $X_{d_0} = X_{d_1} = X_{d_2}$.
(N4) If d_1 is the unique successor of d_0, then $|X_{d_0} \Delta X_{d_1}| \leq 1$, where Δ is the symmetric set difference operator ($A \Delta B = (A \setminus B) \cup (B \setminus A)$).

Nice decompositions can be thought of as a strategy where we place or remove only one cop at a time. It should therefore not be surprising that we can transform any DAG-decomposition into one which is nice.

Theorem 3.13. *If $\mathcal{G}$ has a* DAG-*decomposition of width k, it has a nice* DAG-*decomposition of width k.*

Tree-width on undirected graphs also has a useful characterisation in terms of balanced separators. We are able to obtain one direction of a similar characterisation for DAG-width by showing that graphs of small DAG-width admit small balanced *directed separators*. The definition and proofs can be found in the full version. We also show that the DAG-width of graphs is closed under directed unions, which is considered (see [8]) an important property of a reasonable decomposition of directed graphs.

3.3 Algorithmic Aspects of Bounded DAG-Width

We now consider algorithmic applications of DAG-width as well as the complexity of deciding the DAG-width of a graph and computing an optimal decomposition. The following is a direct consequence of the similar result for tree-width.

Theorem 3.14. *Given a digraph $\mathcal{G}$ and a natural number k, deciding if the* DAG-*width of $\mathcal{G}$ is at most k is* NP-*complete.*

However, for any fixed k, it is possible, in polynomial time, to decide if a graph has DAG-width at most k and to compute a DAG-decomposition of this width if it has. We give an algorithm for this that is based on computing monotone winning strategies in the k-cops and robber game.

Theorem 3.15. *Let $\mathcal{G}$ be a directed graph and let $k < \omega$. There is a polynomial time algorithm for deciding if the cop player has a monotone winning strategy in the k-cops and robber game on $\mathcal{G}$ and for computing such a strategy.*

Note also that the translation of strategies into decompositions is computationally easy, i.e. can be done in polynomial time. Since winning strategies can be computed in polynomial time in the size of the graph, we get the following.

Proposition 3.16. *Given a graph $\mathcal{G}$ of* DAG-*width k, a* DAG-*decomposition of $\mathcal{G}$ of width k can be computed in time $\mathcal{O}(|\mathcal{G}|^{\mathcal{O}(k)})$.*

Algorithms on graphs of bounded DAG*-width.* As the directed tree-width of a graph is bounded above by a constant factor of its DAG-width (see Proposition 4.1), any graph property that can be decided in polynomial time on classes of graphs of bounded directed tree-width can be decided on classes of graphs of bounded DAG-width also. This implies that properties such as Hamiltonicity that are known to be polynomial time on graphs of bounded directed tree-width can be solved efficiently on graphs of bounded DAG-width too. We give a nontrivial application of DAG-width in Section 5 where we show that parity games can be solved on graphs of bounded DAG-width, something which is not known for directed tree-width.

As for the relation to undirected tree-width, it is clear that not all graph properties that can be decided in polynomial time on graphs of bounded tree-width can also be decided efficiently on graphs of bounded DAG-width. For instance, the 3-colourability problem is known to be decidable in polynomial time on graphs of bounded tree-width. However, the problem does not depend on the direction of edges. So if the problem was solvable in polynomial time on graphs of bounded DAG-width then for every given graph we could simply direct the edges so that it becomes acyclic, i.e. of DAG-width 1, and solve the problem then. This shows that 3-colourability is not solvable efficiently on graphs of bounded DAG-width unless PTIME = NP. It also implies that Courcelle's theorem does fail for DAG-width, as 3-colourability is easily seen to be MSO-definable.

The obvious question that arises is whether one can define a suitable notion of "directed problem" and then show that every MSO-definable "directed" graph problem can be decided efficiently on graphs of bounded DAG-width. This is part of ongoing work.

4 Relation to other graph connectivity measures

As a structural measure for undirected graphs, the concept of tree-width is of unrivaled robustness. On the realm of directed graphs, however, its heritage seems to be split among several different concepts. Comparing DAG-width with tree-width, it is easily seen that every tree-decomposition of an undirected graph $\mathcal{G}$ is a DAG-decomposition of the directed graph formed by replacing every edge by two edges, one in each direction. Conversely, the DAG-width of the graph formed in this way is exactly its tree-width. On the other hand a clique with an acyclic orientation provides an example of a digraph with small DAG-width but arbitrarily large tree-width. In the sequel we compare DAG-width with other connectivity measures for digraphs, namely directed tree-width introduced by Johnson et al. [8], and entanglement proposed by Berwanger and Grädel [2].

Directed tree-width. Aiming to reproduce the success of tree-decompositions in allowing divide-and-conquer algorithms, directed tree-width is associated to a tree-shaped representation of the input graph. It was proved that this representation leads to efficient algorithms for solving a particular class of NP-complete problems, including, e.g., Hamiltonicity, when directed tree-width is bounded. Unfortunately this generic method does not cover many interesting problems. In particular, the efficient solution of parity games on bounded tree-width has failed so far to generalise to directed tree-width.

In terms of games, directed tree-width is characterised by a restriction of the cops-and-robber game for DAG-width, in which the robber is only permitted to move to vertices where there exists a directed cop-free path from his intended destination back to his current position. On the basis of the game characterisation, it is clear that the directed tree-width of a graph provides a lower bound for its DAG-width. Conversely, the DAG-width of a graph cannot be bound in terms of its directed tree-width.

Proposition 4.1.

(i) *If a graph has* DAG-*width* k*, its directed tree-width is at most* $3k + 1$.
(ii) *There are graphs with arbitrarily large* DAG-*width and directed tree-width* 1.

Entanglement. The notion of entanglement measures the nesting depth of directed cycles in a graph. In terms of cops-and-robber games, it is obtained by restricting the mobility of both the robber and the cops so that in any round, the cop player may send one cop to the robber's current position (or do nothing) while the robber can only move to a successor of his current residence.

Unlike the other graph widths considered here, entanglement is not associated to an efficient tree-shaped graph representation. Nevertheless, it was shown that parity games on arenas of bounded entanglement can be solved in polynomial time.

The following proposition shows that having bounded DAG-width is more general than having bounded entanglement. On the other hand, the gap between DAG-width and entanglement can be at most logarithmic in the number of graph vertices.

Proposition 4.2.

(i) *If a graph has entanglement* k*, its* DAG-*width is at most* $k + 1$.
(ii) *There are graphs with arbitrarily large entanglement but with* DAG-*width 2.*
(iii) *If a graph* $\mathcal{G}$ *has* DAG-*width* k*, its entanglement is at most* $(k + 1) \cdot \log |V^{\mathcal{G}}|$.

We conclude that, despite their conceptual affinity, directed tree-width, entanglement, and DAG-width are rather different measures.

5 Parity Games on Graphs of Bounded DAG-Width

A parity game $\mathcal{P}$ is a tuple (V, V_0, E, Ω) where (V, E) is a directed graph, $V_0 \subseteq V$ and $\Omega : V \to \omega$ is a function assigning a priority to each node. There is no loss of generality in assuming that the range of Ω is contained in $[n]$ where $n = |V|$ and we will make this assumption from now on.

Intuitively, two players called Odd and Even play a parity game by pushing a token along the edges of the graph with Even playing when the token is on a vertex in V_0 and Odd playing otherwise. Formally, a play of the game $\mathcal{P}$ is an infinite sequence $\pi = (v_i \mid i \in \omega)$ such that $(v_i, v_{i+1}) \in E$ for all i. We say π is winning for Even if $\liminf_{i\to\infty} \Omega(v_i)$ is even and π is winning for Odd otherwise.

A *strategy* is a map $f : V^{<\omega} \to V$ such that for any sequence $(v_0 \cdots v_i) \in V^{<\omega}$, $(v_i, f(v_0 \cdots v_i)) \in E$. A play $\pi = (v_i \mid i \in \omega)$ is consistent with Even playing f if whenever $v_i \in V_0$, $v_{i+1} = f(v_0 \cdots v_i)$. Similarly, π is consistent with Odd playing f if whenever $v_i \notin V_0$, $v_{i+1} = f(v_0 \cdots v_i)$. A strategy f is winning for Even if every play consistent with Even playing f is winning for Even. A strategy is *memoryless* if whenever $u_0 \cdots u_i$ and $v_0 \cdots v_j$ are two sequences in $V^{<\omega}$ with $u_i = v_j$, then

$f(u_0 \cdots u_i) = f(v_0 \cdots v_j)$. It is known that parity games are determined, i.e. for any game and starting position, either Even or Odd has a winning strategy and indeed, a memoryless one. However, we do not assume in our construction that the strategies we consider are memoryless.

The following ordering on $[n]$ is useful in evaluating competing strategies. For priorities $i, j \in [n]$ we say $i \sqsubseteq j$ if either

(i) i is odd and j is even, or
(ii) i and j are both odd and $i \leq j$, or
(iii) i and j are both even and $j \leq i$.

Intuitively, $i \sqsubseteq j$ if the priority i is "better" for player Odd than j.

We are interested in the problem of determining, given a parity game and starting node, which player has a winning strategy. The complexity of this problem in general remains a major open question, as explained in Section 1. We demonstrate that parity games are solvable on arenas of bounded DAG-width by an algorithm similar in spirit to that of Obdržàlek [11]. That algorithm relies on the fact that in a tree-decomposition, a set of k nodes guards all entries and exits to the part of the graph below it, and thus all cycles must pass through this set. In the case of a DAG-decomposition, while the small set guards all exits from the subgraph below it, there may be an unlimited number of edges going into this subgraph. This is the main challenge that our algorithm addresses, and is specifically solved in Lemmas 5.1, 5.2 and 5.3.

For a parity game $\mathcal{P} = (V, V_0, E, \Omega)$ consider $U \subseteq V$ and a set W that guards U. Fix a pair of strategies f and g. For any $v \in U$, there is exactly one play $\pi = (v_i : i \in \omega)$ that is consistent with Even playing f and Odd playing g. Let π' be the maximal initial segment of π that is contained in U. The *outcome* of the pair of strategies (f, g) (given U and v) is defined as follows.

$$\mathrm{out}_{f,g}(U, v) := \begin{cases} \text{winEven} & \text{if } \pi' = \pi \text{ and } \pi \text{ is winning for Even;} \\ \text{winOdd} & \text{if } \pi' = \pi \text{ and } \pi \text{ is winning for Odd;} \\ (v_{i+1}, p) & \text{if } \pi' = v_0 \cdots v_i \text{ and } p = \min\{\Omega(v_j) \mid 0 \leq j \leq i+1\}. \end{cases}$$

By construction, if $\mathrm{out}_{f,g}(U, v) = (w, p)$ then $w \in W$. More generally, for any set $W \subseteq V$, define the set of potential outcomes in W, written pot-out(W), to be the set $\{\text{winEven}, \text{winOdd}\} \cup \{(w, p) : w \in W \text{ and } p \in [n]\}$. We define a partial order $\trianglelefteq$ on pot-out(W) which orders potential outcomes according to how good they are for player Odd. It is the least partial order satisfying the following conditions:

(i) winOdd $\trianglelefteq o$ for all outcomes o;
(ii) $o \trianglelefteq$ winEven for all outcomes o;
(iii) $(w, p) \trianglelefteq (w, p')$ if $p \sqsubseteq p'$ for all $w \in W$.

In particular, (w, p) and (w', p') are incomparable if $w \neq w'$. The idea is that if g and g' are strategies such that $\mathrm{out}_{f,g}(U, v) \trianglelefteq \mathrm{out}_{f,g'}(U, v)$ then player Odd is better off playing strategy g rather than g' in response to Even playing according to f.

A single outcome is the result of fixing the strategies played by both players in the sub-game induced by a set of vertices U. If we fix the strategy of player Even to be f

but consider all possible strategies that Odd may play, we can order these strategies according to their outcome. If one strategy achieves outcome o and another o' with $o \trianglelefteq o'$, there is no reason for Odd to consider the latter strategy. Thus, we define $\text{result}_f(U, v)$ to be the set of outcomes that are achieved by the best strategies that Odd may follow, in response to Even playing according to f. More formally, $\text{result}_f(U, v)$ is the set of $\trianglelefteq$-minimal elements in the set $\{o : o = \text{out}_{f,g}(U, v) \text{ for some } g\}$. Thus, $\text{result}_f(U, v)$ is an anti-chain in the partial order (pot-out$(W), \trianglelefteq$), where W is a set of guards for U. We write pot-res(W) for the set of *potential results* in W. To be precise, pot-res(W) is the set of all anti-chains in the partial order (pot-out$(W), \trianglelefteq$). By definition of the order $\trianglelefteq$, if either of winEven or winOdd is in the set $\text{result}_f(U, v)$, then it is the sole element of the set. Also, for each $w \in W$, there is at most one p such that $(w, p) \in \text{result}_f(U, v)$ so the number of distinct values that $\text{result}_f(U, v)$ can take is at most $(|U| + 1)^{|W|} + 2$ (in fact, $(d + 1)^{|W|}$, where d is the number of different priorities in U). This is the cardinality of the set pot-res(W).

We also abuse notation and extend the order $\trianglelefteq$ to the set pot-res(W) pointwise. That is, for $r, s \in$ pot-res(W) we write $r \trianglelefteq s$ if, for each $o \in s$, there is an $o' \in r$ with $o' \trianglelefteq o$. With this definition, the order $\trianglelefteq$ on pot-res(W) admits greatest lower bounds. Indeed, the greatest lower bound $r \sqcap s$ of r and s can be obtained by taking the set of $\trianglelefteq$ minimal elements in the set of outcomes $r \cup s$. One further piece of notation we use is that we write $\text{Res}(U, v)$ for the set $\{\text{result}_f(U, v) : f \text{ is a strategy}\}$.

Suppose now that $\mathcal{P} = (V, V_0, E, \Omega)$ is a parity game and we are given a DAG decomposition $(\mathcal{D}, (X_d)_{d \in V^{\mathcal{D}}})$ of (V, E) of width k that is nice in the sense of Definition 3.12. For each $d \in V^{\mathcal{D}}$, we write V_d for the set $\mathcal{X}_d \setminus X_d$. The key to the algorithm is that we construct the set of results $\text{Res}(V_d, v)$ for each $v \in V_d$. Since V_d is guarded by X_d, $|X_d| \leq k$ and $|V_d| \leq n$, the number of distinct values of $\text{result}_f(V_d, v)$ as f ranges over all possible strategies is at most $(n + 1)^k + 2$.

We define the following, which is our key data structure: $\text{Frontier}(d) = \{(v, r) : v \in V_d \text{ and } r = \text{result}_f(V_d, v) \text{ for some strategy } f\}$. Note that in the definitions of $\text{result}_f(U, v)$ and Frontier(d), f and g range over *all* strategies and not just memoryless ones. The bound on the number of possible values of $\text{result}_f(V_d, v)$ guarantees that $|\text{Frontier}(d)| \leq n((n + 1)^k + 2)$. We aim to show how Frontier(d) can be constructed from the set of frontiers of the successors of d in polynomial time. There are four cases to consider.

Case 1: d has two successors e_1 and e_2. In this case, $X_d = X_{e_1} = X_{e_2}$ by (N2). We claim that $\text{Frontier}(d) = \text{Frontier}(e_1) \cup \text{Frontier}(e_2)$ (see full paper).

Case 2: d has one successor e and $X_d = X_e$. In this case, $\text{Frontier}(d) = \text{Frontier}(e)$.

Case 3: d has one successor e and $X_d \setminus X_e = \{u\}$. Then, by (D3), $u \notin V_e$. Also, by definition of V_d, $u \notin V_d$. We conclude that $V_d = V_e$. Moreover, since X_e guards V_e, there is no path from any element of V_e to u except through X_e. Hence, $\text{Frontier}(d) = \text{Frontier}(e)$.

Case 4: d has one successor e and $X_e \setminus X_d = \{u\}$. This is the critical case. Here $V_d = V_e \cup \{u\}$ and in order to construct Frontier(d) we must determine the results of all plays beginning at u.

Consider the set of vertices v in $\mathcal{X}_d$ such that $(u, v) \in E^{\mathcal{G}}$. These fall into two categories. Either $v \in X_d$ or $v \in V_e$. Let $x_1, \ldots, x_s$ enumerate the first category and let $v_1, \ldots, v_m$ enumerate the second. Let $O = \{(x_i, \min\{\Omega(x_i), \Omega(u)\}) : 1 \leq i \leq s\}$. This is the set of outcomes obtained if play in the parity game proceeds directly from u to an element of X_d. Note that as no two outcomes in O are comparable with respect to $\trianglelefteq$, $O \in$ pot-res(X_d). We write $\mathcal{O}$ for $\{\{o\} : o \in O\}$ That is $\mathcal{O}$ is the set of singleton results obtained from O. For each v_i we know, from Frontier(e), the set Res(V_e, v_i). For each result $r \in$ Res(V_e, v_i), we write mod(r) for the set of outcomes defined by modifying r as follows. First, if r contains an outcome (u, p), we replace it by winEven if $\min\{p, \Omega(u)\}$ is even and winOdd if it is odd. Secondly, for any pair $(w, p) \in r$ where $w \neq u$, we replace it with $(w, \min\{p, \Omega(u)\})$. Finally, we take the set of $\trianglelefteq$-minimal elements from the resulting set. This is mod(r). Note that mod$(r) \in$ pot-res(X_d). The intuition is that mod(result$_f(V_e, v_i)$) defines the set of best possible outcomes for player Odd, if starting at u, the play goes to v_i and from that point on, player Even plays according to strategy f. For each $1 \leq i \leq m$, let $M_i = \{\text{mod}(r) : r \in \text{Res}(V_e, v_i)\}$.

We now wish to use the sets of results M_i, O and $\mathcal{O}$ to construct the Res(V_d, u). We need to distinguish between the cases when $u \in V_0$ (i.e. player Even plays from u in the parity game) and $u \in V \setminus V_0$ (i.e. player Odd plays).

The simpler case is when $u \in V_0$.

Lemma 5.1. *If $u \in V_0$, then Res$(V_d, u) = \bigcup_i M_i \cup \mathcal{O}$.*

The case when $u \notin V_0$ is somewhat trickier. To explain how we can obtain Res(V_d, u) in this case, we formulate the following lemma.

Lemma 5.2. *If $u \notin V_0$, then $r \in$ Res(V_d, u) if, and only if, there is a function c on the set $[m]$ with $c(i) \in M_i$ such that $r = O \sqcap \prod_{i \in [m]} c(i)$.*

Lemma 5.2 suggests constructing Res(V_d, u) by considering all possible choice functions c. However, as each set M_i may have as many as $(n+1)^k + 2$ elements, there are $m^{(n+1)^k+2}$ possibilities for c and our algorithm would be exponential. We consider an alternative way of constructing Res(V_d, u). Recall that Res$(V_d, u) \subseteq$ pot-res(X_d) and the latter set has at most $(n+1)^k+2$ elements. We check, for each $r \in$ pot-res(X_d), in polynomial time, whether there is a choice function c as in Lemma 5.2 that yields r. In particular, we take the following alternative characterisation of Res(V_d, u).

Lemma 5.3. *If $u \notin V_0$, then $r \in$ Res(V_d, u) if, and only if, there is a set $D \subseteq [m]$ with $|D| \leq |r|$ and a function d on D with $d(i) \in M_i$ such that*

(i) $r = O \sqcap \prod_{i \in D} d(i)$*; and*
(ii) *for each $i \notin D$ there is an $r_i \in M_i$ with $r \trianglelefteq r_i$.*

Now, any $r \in$ pot-res(X_d) has at most k elements. Thus, to check whether such an r is in Res(V_d, u) we cycle through all sets $D \subseteq [m]$ with k or fewer elements (and there are $\mathcal{O}(n^k)$ such sets) and for each one consider all candidate functions d (of which there are $\mathcal{O}(n^{k^2})$). Having found a d which gives $r = O \sqcap \prod_D d(i)$, we then need to find a suitable r_i in each $i \in [m] \setminus D$. For this we must, at worst, go through all elements of all the sets M_i and compare them to r. This can be done in time $\mathcal{O}(n^{k+1})$.

We have now obtained the set Res(V_d, u). One barrier remains to completing the construction of Frontier(d). Elements (v, r) of Frontier(e) may have outcomes in r

of the form (u, p). Since u is not in X_d, these must be resolved by combining them with results from $\mathrm{Res}(V_d, u)$. To be precise, let $r \in \mathrm{Res}(V_e, v)$ for some $v \in V_e$ and $s \in \mathrm{Res}(V_d, u)$. Define the combined result $c(r, s)$ as follows:

– if r does not contain an outcome of the form (u, p), then $c(r, s) = r$;
– otherwise, r contains a pair (u, p). Let s' be obtained from s by replacing every pair (w, q) by $(w, \min\{p, q\})$. $c(r, s) = r \sqcap s'$.

Intuitively, if $r = \mathrm{result}_f(V_e, v)$ and $s = \mathrm{result}_{f'}(V_d, u)$ then $c(r, s)$ is the set of $\trianglelefteq$-minimal outcomes that can be obtained if player Even plays according to f starting at v *until* the node u is encountered and then switches to strategy f'.

Lemma 5.4. *For* $v \in V_e$*, Res*$(V_d, v) = \{c(r, s) : r \in$ *Res*(V_e, v) *and* $s \in$ *Res*$(V_d, u)\}$.

We now obtain $\mathrm{Frontier}(d) = \{(v, r) : r \in \mathrm{Res}(V_d, v)\}$.

Theorem 5.5. *For each k, there is a polynomial p and an algorithm running in time $\mathcal{O}(p(n))$ which determines the winner of parity games on all graphs with* DAG*-width at most k.*

References

1. J. BARÁT, *Directed path-width and monotonicity in digraph searching*. To appear in *Graphs and Combinatorics*.
2. D. BERWANGER AND E. GRÄDEL, *Entanglement – a measure for the complexity of directed graphs with applications to logic and games*, in LPAR, 2004, pp. 209–223.
3. H. L. BODLAENDER, *Treewidth: Algorithmic techniques and results*, in MFCS, 1997, pp. 19–36.
4. B. COURCELLE, *Graph rewriting: An algebraic and logic approach*, in Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B), J. van Leeuwan, ed., 1990, pp. 193–242.
5. N. D. DENDRIS, L. M. KIROUSIS, AND D. M. THILIKOS, *Fugitive-search games on graphs and related parameters*, TCS, 172 (1997), pp. 233–254.
6. E. EMERSON, C. JUTLA, AND A. SISTLA, *On model checking for the μ-calculus and its fragments*, TCS, 258 (2001), pp. 491–522.
7. G. GOTTLOB, N. LEONE, AND F. SCARCELLO, *Robbers, marshals, and guards: Game theoretic and logical characterizations of hypertree width*, in PODS, 2001, pp. 195–201.
8. T. JOHNSON, N. ROBERTSON, P. D. SEYMOUR, AND R. THOMAS, *Directed tree-width*, Journal of Combinatorial Theory, Series B, 82 (2001), pp. 138–154.
9. M. JURDZIŃSKI, *Deciding the winner in parity games is in UP $\cap$ co-UP*, Information Processing Letters, 68 (1998), pp. 119–124.
10. D. KOZEN, *Results on the propositional mu-calculus*, TCS, 27 (1983), pp. 333–354.
11. J. OBDRŽÁLEK, *Fast mu-calculus model checking when tree-width is bounded*, in Proceedings of 15th International Conference on Computer Aided Verification, vol. 2725 of LNCS, Springer, 2003, pp. 80–92.
12. B. A. REED, *Introducing directed tree width*, in 6th Twente Workshop on Graphs and Combinatorial Optimization, vol. 3 of Electron. Notes Discrete Math, Elsevier, 1999.
13. N. ROBERTSON AND P. SEYMOUR, *Graph Minors. III. Planar tree-width*, Journal of Combinatorial Theory, Series B, 36 (1984), pp. 49–63.
14. M. SAFARI, *D-width: A more natural measure for directed tree width*, in MFCS 2005, vol. 3618 of LNCS, Springer, 2005, pp. 745–756.
15. P. SEYMOUR AND R. THOMAS, *Graph searching, and a min-max theorem for tree-width*, Journal of Combinatorial Theory, Series B, 58 (1993), pp. 22–33.