Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 399 (2008) 206–219
Theoretical Computer Science
www.elsevier.com/locate/tcs

Digraph measures: Kelly decompositions, games, and orderings

Paul Hunter[a], Stephan Kreutzer[b,*]

[a] *Computer Laboratory, University of Cambridge, Cambridge CB3 0FD, UK*
[b] *Oxford University Computing Laboratory, Oxford OX1 3QD, UK*

Abstract

We consider various well-known, equivalent complexity measures for graphs such as elimination orderings, k-trees and cops and robber games and study their natural translations to digraphs. We show that on digraphs the translations of these measures are also equivalent and induce a natural connectivity measure. We introduce a decomposition for digraphs and an associated width, Kelly-width, which is equivalent to the aforementioned measure. We demonstrate its usefulness by exhibiting potential applications including polynomial-time algorithms for NP-complete problems on graphs of bounded Kelly-width, and complexity analysis of asymmetric matrix factorization. Finally, we compare the new width to other known decompositions of digraphs.

Keywords: Cop and robber games; Directed graphs; Graph decompositions

1. Introduction

An important and active field of algorithm theory is to identify natural classes of structures or graphs which are algorithmically well-behaved, i.e. on which efficient solutions to otherwise NP-complete problems can be found. A particularly rich source of tractable cases comes from graph structure theory in the form of graph decompositions and associated measures of structural complexity such as tree-width or rank-width. For instance, Courcelle's celebrated theorem [8] shows that every property of undirected graphs that can be formulated in monadic second-order logic can be decided in linear time on any class of graphs of bounded tree-width. This result immediately implies linear-time algorithms for a huge class of problems on such graphs. Since then, hundreds of papers have been published describing efficient algorithms for graph problems on classes of graphs of bounded tree-width. (See e.g. [5] and references therein.) Similarly, efficient algorithms can sometimes be found for planar graphs [13,2] or more general classes of graphs, for instance classes of graphs of bounded local tree-width [14], or graph classes excluding a minor (see e.g. [10] and references therein). Another interesting example are classes of graphs of bounded clique- or rank-width [9,25,24].

All the examples mentioned above are defined by imposing restrictions on the underlying undirected graph structure. However, there are many applications where the input structures – networks, state transition systems,

* Corresponding author.
E-mail addresses: paul.hunter@cl.cam.ac.uk (P. Hunter), stephan.kreutzer@comlab.ox.ac.uk (S. Kreutzer).

0304-3975/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.02.038

dependency graphs as in database theory, or the arenas of combinatorial games such as parity games – are more naturally modelled as directed rather than undirected graphs. In these cases, the notion of tree-width is unsatisfactory as it does not take the direction of edges into account. This information loss may be crucial, as demonstrated by the problem of finding a Hamiltonian path in a digraph: an acyclic orientation of a grid has very high tree- or clique-width, but the Hamiltonian path problem on the digraph can be solved efficiently.

As a consequence, several authors have tried to generalise notions like tree- or path-width from undirected to directed graphs (see e.g. [26,20,3,29,23,4]). In [20], Johnson, Robertson, Seymour, and Thomas concentrate on the connectivity aspect of tree-width, generalising this to strong connectivity in the directed case, to define directed tree-width. In the same paper, the authors give an algorithmic application of directed tree-width by showing that a number of NP-complete problems such as Hamiltonicity or the k-disjoint paths problems become tractable on graphs of small directed tree-width. Berwanger, Dawar, Hunter and Kreutzer [4] and, independently, Obdržálek [23] introduce the notion of DAG-width. DAG-width is a slightly weaker notion than directed tree-width, in the sense that more graphs have small directed tree-width than small DAG-width, but it has a cleaner characterisation in terms of cops and robber games and gives more control over the graph, as the guarding condition used is stricter. In [4], this has been used to show that the winner of a parity game – a form of combinatorial games played on digraphs – can be decided in polynomial time provided the game graph has bounded DAG-width. The analogous question remains open for graphs of bounded directed tree-width.

Both directed tree-decompositions and DAG-decompositions provide a natural and interesting connectivity measure for directed graphs. Both, however, also suffer from some difficulties. As Adler shows [1], directed tree-decompositions are not closed under even mild forms of directed minors (butterfly minors), the corresponding games are neither cop- nor robber-monotone, and there is no precise characterisation of directed tree-width by these games (only up to a constant factor). Although this is not really a problem for algorithmic applications, it suggests that the notion of directed tree-width may not be as well-behaved as undirected tree-width. It may, though, be possible to modify the definition of directed tree-width to overcome some of the difficulties. DAG-decompositions, on the other hand, suffer from the fact that the best upper bound for the size of DAG-decompositions of graphs of width k known so far is $\mathcal{O}(n^k)$. This is a significant problem, as the space consumption of algorithms is often more problematic than running time. It is not known whether deciding that a digraph has DAG-width at most k is in NP when k is part of the input, contrary to the authors' claims. (It is NP-hard. This follows easily from the NP-completeness of the corresponding question for tree-width.)

Whereas for undirected graphs it is widely accepted that tree-width is the "right" notion, the problems described above suggest that more research is needed to decide what the "right" notion for digraphs is — if there is any. A natural way to search for practical generalisations of undirected tree-width is to look at useful equivalent characterisations of it and translate them to digraphs.

In this paper we consider three characterisations of tree-width: partial k-trees, elimination orderings and a graph searching game in which an invisible robber attempts to avoid capture by a number of cops, subject to the restriction that he may only move if a cop is about to occupy his position. Partial k-trees are the historical forerunner of tree-width and are therefore associated with graph structure theory [28], elimination orderings have found application in the analysis of symmetric matrix factorization, such as Cholesky decomposition [22], and graph searching problems have recently been used to explore and generate robust measures of graph complexity (see e.g. [11,15]). We generalise all of these to directed graphs, resulting in partial k-DAGs, directed elimination orderings, and an inert robber game on digraphs. We show that all of these generalisations are equivalent on digraphs and are also equivalent to the width-measure associated to a new kind of decomposition we introduce. As the game is reminiscent of capturing hideout-based outlaws, we propose the name Kelly-decompositions, after the infamous Australian bushranger Ned Kelly. The fact that all these notions are equivalent on digraphs as they are on undirected graphs suggests that this might be a robust measure of complexity/connectivity of digraphs.

In addition to being equivalent to the natural generalisations of the above characterisations, we believe that Kelly-decompositions have many advantages over DAG-decompositions and directed tree-decompositions. Unlike the former, the size of these decompositions can be made linear in the size of the graph it decomposes. On the other hand, their structure and strict guarding condition make them suitable for constructing dynamic programming algorithms which can lead to polynomial-time algorithms for NP-complete problems on graphs of bounded Kelly-width. We also show how they are applicable to asymmetric matrix factorization by relating them to the elimination DAGs of [16].

The paper is organised as follows. In Section 3 we formally define elimination orderings, inert robber games, and partial k-DAGs and show the equivalence of the associated width measures. In Section 4, we introduce Kelly-decompositions and Kelly-width. In Section 5, we present applications: algorithms for Hamiltonian cycle, weighted disjoint paths and parity games that all run in polynomial time on graphs of bounded Kelly-width, and details of the connection between Kelly-decompositions and asymmetric matrix factorization. Finally, we compare our new width measure to other known measures on digraphs, in particular to directed tree-width and DAG-width.

An extended abstract of the paper appeared in [18].

2. Preliminaries

We use standard graph theory notation. See e.g. [12]. Let G be a digraph. We write $V(G)$ for its vertex set and $E(G)$ for its edge set. For $X \subseteq V(G)$ we write $G[X]$ for the subgraph of G induced by X and $G \setminus X$ for $G[V(G) \setminus X]$. If $X := \{v\}$ is a singleton set, we simply write $G \setminus v$. Finally, we sometimes write $G[v_1, \ldots, v_k]$ for $G[\{v_1, \ldots, v_k\}]$. For a subgraph H of G and $v \in V(H)$, we write $Reach_H(v)$ for the set of vertices in $V(H)$ reachable from v by a directed path in H. If G is a directed, acyclic graph (DAG), we write $\preceq_G$ for the reflexive, transitive closure of the edge relation.

3. Elimination orderings, inert robber games, and partial k-DAGs

In this section we formally define directed elimination orderings, inert robber games, and partial k-DAGs and show that the associated width measures of digraphs are equivalent.

Our first definition extends the idea of vertex elimination to digraphs. Vertex elimination is the process of removing vertices from a graph but adding edges to preserve reachability. The complexity measure we are interested in is the maximum out-degree of eliminated vertices.

Definition 1 (*Directed Elimination Ordering*). Let G be a digraph.

- An *(directed) elimination ordering* $\lhd$ is a linear ordering on $V(G)$.
- Given an elimination ordering $\lhd := (v_0, v_1, \ldots, v_{n-1})$ of G, we define:
  - $G_0^{\lhd} := G$ and
  - $G_{i+1}^{\lhd}$ is obtained from $G_i^{\lhd}$ by deleting v_i and adding new edges (if necessary) (u, v) if $(u, v_i), (v_i, v) \in E(G_i^{\lhd})$ and $u \neq v$.

  $G_i^{\lhd}$ is the *directed elimination graph at step i according to* $\lhd$.
- The *width* of an elimination ordering is the maximum over all i of the out-degree of v_i in $G_i^{\lhd}$.

For convenience, given an elimination ordering $\lhd = (v_0, \ldots, v_{n-1})$, we define the *support of v_i with respect to* $\lhd$ as $\mathrm{supp}_{\lhd}(v_i) := \{x : (v_i, x) \in E(G_i^{\lhd})\}$. Note that the width of $\lhd$ is the maximum cardinality of all supports.

Immediately from the definitions, we have this simple lemma relating the support of an element in an elimination ordering to the set of vertices reachable from that vertex.

Lemma 2. *Let $\lhd$ be a directed elimination ordering of a graph G and let $v \in V(G)$. Let $R := \{u : v \lhd u\}$. Then*

$$supp_{\lhd}(v) = \left\{ u : \begin{array}{l} v \lhd u \text{ and there is } v' \in Reach_{G \setminus R}(v) \\ \text{such that } (v', u) \in E(G) \end{array} \right\}.$$

In the sequel we will often use this result without citation.

We proceed with defining inert robber games on digraphs. Intuitively, a robber occupies some vertex of a graph G. A given number of cops attempt to capture this robber by occupying the same vertex as the robber. The robber evades capture by being able to run from his position along any directed path which does not pass through a cop. Any number of cops can move anywhere on the graph but they do so by removing themselves completely from the graph and then announcing where they are moving. It is during this transition that the robber moves. In the inert robber game, the robber may only move if a cop is about to land on his current position, however he is not visible to the cops and he knows the cops' strategy in advance. The following definition makes this precise.

Definition 3 (*Inert Robber Game*). The *(k cop) inert robber game* on a digraph G is the set of all plays, where a *play* is a sequence

$$(X_0, R_0), (X_1, R_1), \ldots, (X_m, R_m),$$

such that $(X_0, R_0) = (\varnothing, V(G))$ and for all i: $X_i, R_i \subseteq V(G)$; $|X_i| \leq k$; and

$$R_{i+1} = \left(R_i \cup \bigcup_{v \in R_i \cap X_{i+1}} Reach_{G \setminus (X_i \cap X_{i+1})}(v) \right) \setminus X_{i+1}.$$

Intuitively, in a play $(X_0, R_0), (X_1, R_1), \ldots, (X_m, R_m)$, the X_i represent the cop locations, and the R_i represent the set of potential robber locations (also known as contaminated vertices).

In games such as these, we are usually concerned with *strategies*. A *cop strategy* is a function that, given a play, indicates the next set of cop locations. A *robber strategy* is a function that, given a play and a set of cop locations, indicates the next set of potential robber locations. However, in this game, we observe that the next set of robber locations is completely determined by the next set of cop locations. Thus we can simply regard a cop strategy as a sequence $X_0, X_1, \ldots$ of sets of vertices such that $|X_i| \leq k$ for all i. Note that given such a sequence we can reconstruct the play which arises if the cops always move to the location specified by the strategy. We call this the *associated play*. A strategy $X_0, X_1, \ldots, X_n$ is *winning* if $R_n = \varnothing$ in the associated play. Finally, a strategy is *monotone* if $R_i \supseteq R_{i+1}$ for all i in the associated play.

The last characterisation we consider is a generalisation of partial k-trees, called partial k-DAGs. The class of k-trees can be viewed as a class of graphs generated by a generalisation of how one might construct a tree. In the same way, k-DAGs are a class of digraphs generated by a generalisation of how one might construct a directed, acyclic graph in a top-down manner.

Definition 4 (*(Partial) k-DAG*). The class of k-DAGs is defined recursively as follows:

- A complete digraph[1] with k vertices is a k-DAG.
- A k-DAG with $n+1$ vertices can be constructed from a k-DAG H with n vertices by adding a vertex v and edges satisfying the following:
  - At most k edges from v to H are added
  - If X is the set of endpoints of the edges added in the previous subcondition, an edge from $u \in V(H)$ to v is added if $(u, w) \in E(H)$ for all $w \in X \setminus \{u\}$. Note that if $X = \varnothing$, this condition is true for all $u \in V(H)$.

A *partial k-DAG* is a subgraph of a k-DAG.

The second condition on the edges provides a method to add as many edges as possible going to the new vertex without introducing cycles. Note that this definition generalises k-trees, for if the vertices (X) adjacent to the new vertex (v) form a clique, we will add edges back from X to v, effectively creating undirected edges between v and X (and possibly some additional edges from $H \setminus X$ to v). Note that a partial 0-DAG is a DAG.

Our main result of this section is that the three measures introduced are equivalent on digraphs.

Theorem 5. *Let G be a digraph. The following are equivalent:*

(1) *G has a directed elimination ordering of width* $\leq k$.
(2) $k+1$ *cops have a monotone winning strategy to capture an inert robber.*
(3) *G is a partial k-DAG.*

Proof. $1 \Rightarrow 3$: Let $\lhd = (v_0, v_1, \ldots, v_{n-1})$ be a directed elimination ordering of G of width k. For ease of notation, define $X_i := \text{supp}_{\lhd}(v_i)$ and $m := n - k$. Let $\mathcal{K}_0$ be a complete graph on the k vertices $\{v_m, v_{m+1}, \ldots, v_{n-1}\}$, and let $\mathcal{K}_j$ $(j \geq 1)$ be the k-DAG formed by adding v_{m-j} to $\mathcal{K}_{j-1}$, and edges from v_{m-j} to X_{m-j} (together with the other edges necessarily added from $\mathcal{K}_{j-1}$ to v_{m-j} in the definition of k-DAGs.) We claim that for all $0 \leq j \leq m$, $G^{\lhd}_{m-j}$ is a partial graph of $\mathcal{K}_j$. The result then follows by taking $j = n - k$. We prove our claim by induction on j. For the base

[1] By a complete digraph we mean a digraph with edges in both directions between any pair of distinct vertices.

case ($j = 0$) the result is trivial as $\mathcal{K}_j$ is a complete graph. Now assume the result is true for $j \geq 0$, and consider the graph $G^{\lhd}_{m-j-1}$. For simplicity let $i = m - j - 1$. For every edge (u, v) in $G^{\lhd}_i$ either: (a) $v_i \notin \{u, v\}$, (b) $u = v_i$, or (c) $v = v_i$.

In the first case, $(u, v) \in E(G^{\lhd}_{i+1})$ and therefore in $E(\mathcal{K}_j) \subseteq E(\mathcal{K}_{j+1})$ by the induction hypothesis. For the second case, (u, v) is added during the construction of $\mathcal{K}_{j+1}$. For the final case, for any $w \in X_i$, (v_i, w) is an edge of $G^{\lhd}_i$, so (u, w) is an edge of $G^{\lhd}_{i+1}$ (for $u \neq w$), and therefore of $\mathcal{K}_j$ by the induction hypothesis. Thus (u, v_i) is added during the construction of $\mathcal{K}_{j+1}$, and $E(G^{\lhd}_i) \subseteq E(\mathcal{K}_{j+1})$ as required.

$3 \Rightarrow 2$: Let G be a partial k-DAG. Suppose G is a partial graph of the k-DAG, $\mathcal{K}$, formed from a complete graph, on the vertices $X_k := \{v_1, v_2, \ldots, v_k\}$, and then by adding the vertices $v_{k+1}, v_{k+2}, \ldots, v_n$ in that order. For $1 \leq i \leq n - k$ let $X_{k+i} \subseteq \{v_1, \ldots, v_{k+i-1}\}$ denote the set of successors of v_{k+i}. That is, when v_{k+i} is added during the construction of $\mathcal{K}$, edges are added from v_{k+i} to each vertex in X_{k+i}. Note that for all i, $|X_i| \leq k$. We claim that the sequence:

$$\varnothing, X_k, X_{k+1}, X_{k+1} \cup \{v_{k+1}\}, X_{k+2}, X_{k+2} \cup \{v_{k+2}\}, \ldots, X_n, X_n \cup \{v_n\}$$

is a monotone winning strategy for $k + 1$ cops. Let $R_i = \{v_j : j > i\}$, then from the definition of k-DAGs and the X_i, it is easy to see that the play associated with the strategy is:

$$(\varnothing, V(G)), (X_k, R_k), (X_{k+1}, R_k), (X_{k+1} \cup \{v_{k+1}\}, R_{k+1}), \ldots, (X_n, R_{n-1}), (X_n \cup \{v_n\}, \varnothing).$$

As $R_i \supseteq R_{i+1}$ for all i, the strategy is monotone and winning as required.

$2 \Rightarrow 1$: Suppose $k + 1$ cops have a monotone winning strategy. We assume only one cop is placed at a time. Order the vertices in terms of the point at which they are first occupied by a cop and then reverse this order, so that v_j appears later than v_i if and only if v_j was first occupied by a cop before v_i was. Call this ordering $\lhd$. We claim $\lhd$ has width $\leq k$. If this were not the case, there must exist v_i such that $|\mathrm{supp}_{\lhd}(v_i)| \geq k + 1$. The inert robber can then defeat the strategy of the cops by starting on v_i. At the point when a cop first occupies v_i there are at most k cops on $\mathrm{supp}_{\lhd}(v_i)$ so there exists $v_j \in \mathrm{supp}_{\lhd}(v_i)$ which is not currently occupied. Furthermore, no cop is on any vertex which appears earlier than v_i in $\lhd$, so the robber is able to reach v_j. However, as $j > i$, v_j has been occupied by a cop in the past and was therefore not available as a robber position — contradicting the monotonicity of the strategy. □

It follows from this theorem that the minimal width over all directed elimination orderings of G and the minimal number of cops required to capture an inert robber (less one) coincide, and this class of digraphs is characterised by partial k-DAGs. This leads to the following definition:

Definition 6 (*Elimination Width*). Let G be a digraph. The *(directed) elimination width* of G is the minimal width over all directed elimination orderings of G.

4. Decompositions

With a robust measure for digraph complexity defined, we now turn to the problem of finding a closely related digraph decomposition.

4.1. Kelly-decompositions

The decomposition we introduce is a partition of the vertex set, arranged as a directed acyclic graph, together with sets of vertices which guard against paths in the graph that do not respect this arrangement. We have an additional restriction to avoid trivial decompositions — vertices in the guard sets must appear either to the left or earlier in the decomposition. More precisely,

Definition 7 (*Guarding*). Let G be a digraph. We say $W \subseteq V(G)$ *guards* $X \subseteq V(G)$ if $W \cap X = \varnothing$ and for all $(u, v) \in E(G)$ with $u \in X$, we have $v \in X \cup W$.

Definition 8 (*Kelly-decomposition and Kelly-width*). A *Kelly-decomposition* of a digraph G is a triple $\mathfrak{D} := (D, (B_t)_{t \in V(D)}, (W_t)_{t \in V(D)})$ so that

- D is a DAG and $(B_t)_{t \in V(D)}$ partitions $V(G)$,

- for all $t \in V(D)$, $W_t \subseteq V(G)$ guards $\mathcal{B}_t^{\downarrow} := \bigcup_{t' \geq_D t} B_{t'}$, and
- for all $s \in V(D)$ there is a linear order on its children $t_1, \ldots, t_p$ so that for all $1 \leq i \leq p$, $W_{t_i} \subseteq B_s \cup W_s \cup \bigcup_{j<i} \mathcal{B}_{t_j}^{\downarrow}$. Similarly, there is a linear order on the roots such that $W_{r_i} \subseteq \bigcup_{j<i} \mathcal{B}_{r_j}^{\downarrow}$.

The *width* of $\mathfrak{D}$ is $\max\{|B_t \cup W_t| : t \in V(D)\}$. The *Kelly-width* of G is the minimal width of any of its Kelly-decompositions.

Our main result of this section is that Kelly-decompositions do in fact correspond with the complexity measure defined at the end of the previous section.

Theorem 9. *G has directed elimination width $\leq k$ if, and only if, G has Kelly-width $\leq k+1$.*

Proof. Let G be a digraph and $(D, (B_t)_{t \in V(D)}, (W_t)_{t \in V(D)})$ a Kelly-decomposition of G of width $k+1$. Let $\mathcal{T}$ be the spanning tree of D obtained from the depth-first traversal of D which always chooses the largest child according to the ordering on children. Let $(t_1, t_2, \ldots)$ be the order of $V(\mathcal{T})$ (and hence, $V(D)$) visited in the depth-first traversal of $\mathcal{T}$ which always chooses the smallest child according to the ordering. We claim that the sequence:

$$W_{t_1}, W_{t_1} \cup B_{t_1}, W_{t_2}, W_{t_2} \cup B_{t_2}, \ldots$$

defines a monotone winning strategy for $k+1$ cops. At step $2i-1$, it follows from the definitions that $\bigcup_{j \leq i} W_{t_j}$ cannot contain the robber, so the robber cannot move at this step. At step $2i$, the robber is forced further down the DAG, and therefore into a smaller region. Thus, this strategy will monotonely capture the robber.

For the converse, let $\lhd$ be a directed elimination ordering on G of width at most k. Let $v_1 \lhd \ldots \lhd v_n$ be an enumeration of the vertices of G ordered by $\lhd$. For convenience we associate each vertex v_i with its index i. In particular, we write $G_t := G[1, \ldots, t]$ for the induced subgraph $G[v_1, \ldots, v_t]$.

Define $(D, (B_t)_{t \in V(D)}, (W_t)_{t \in V(D)})$ as follows. $V(D) := V(G)$. For all $t \in V(D)$ let $B_t := \{t\}$ and $W_t := \mathrm{supp}_{\lhd}(t)$. Towards defining the edge relation, let $t \in V(D)$ be a vertex. Let $C_1, \ldots, C_p$ be the strongly connected components of $G_t \setminus t$. Let $t_1, \ldots, t_p$ be the $\lhd$-maximal elements of $C_1, \ldots, C_p$, respectively. We put an edge (t, t_i) between t and t_i if t_i is reachable from t in G_t and there is no t_j with $t_i \lhd t_j \lhd t$ such that t_j is reachable from t in G_t and t_i is reachable from t_j in $G_t \setminus t$.

We claim that $(D, (B_t)_{t \in V(D)}, (W_t)_{t \in V(D)})$ is a Kelly-decomposition of width $\leq k+1$. Clearly, D is a DAG, as all the edges in $E(D)$ are oriented following the ordering $\lhd$. Further, the width of the decomposition is clearly one more than the width of $\lhd$. To establish the guarding property, we first show the following claim.

Claim. For all $t \in V(D)$, $Reach_{G_t}(t) = \mathcal{B}_t^{\downarrow}$.

We first show by induction on t that $Reach_{G_t}(t) \subseteq \mathcal{B}_t^{\downarrow}$. For $t = 1$ there is nothing to show. Suppose the claim has been proved for all $i < t$. Let $v \in Reach_{G_t}(t)$. Let $C_1, \ldots, C_m$ be the strongly connected components of $G_t \setminus t$. Without loss of generality we assume that $v \in C_1$. Let s be the $\lhd$-maximal element of C_1 and let t' be the $\lhd$-maximal element such that

- t' is the $\lhd$-maximal element of some C_i
- there is a directed path from t to t' in G_t
- there is a directed path from t' to s in $G_t \setminus t$.

By construction, there is an edge $(t, t') \in E(D)$. If $t' = v$, or in fact if t' is the $\lhd$-maximal element of C_1, then there is nothing more to show. Otherwise, if t' and v are not in the same strongly connected component of $G_t \setminus t$, then s, and hence v, must be reachable from t' in $G_{t'}$. For, by construction, s is reachable from t' in $G_t \setminus t$ and t' is the $\lhd$-maximal element reachable from t in G_t and from which s can be reached in $G_t \setminus t$. Thus, if s was not reachable from t' in $G_{t'}$ then the only path from t' to s in $G_t \setminus t$ must involve an element $w \lhd t$ such that $t' \lhd w$, contradicting the maximality of t'. Hence, v is reachable from t' in $G_{t'}$ and therefore, by induction hypothesis, $v \in \mathcal{B}_{t'}^{\downarrow} \subseteq \mathcal{B}_t^{\downarrow}$.

A simple induction on the height of the nodes in D establishes the converse. $\dashv$

It remains to show that for all $s \in V(D)$ there is a linear ordering $\sqsubset$ of the children s satisfying the ordering condition required by the definition of Kelly-decompositions. For children $v \neq v'$ of s define $v \sqsubset v'$ if $v' \lhd v$, i.e. $\sqsubset$ is the inverse ordering of $\lhd$.

Let $t_1, \ldots, t_m$ be the children of s ordered by $\sqsubset$. We claim that for all $i \in \{1, \ldots, m\}$,

$$W_{t_i} \subseteq B_s \cup W_s \cup \bigcup_{j<i} \mathcal{B}^{\downarrow}_{t_j}.$$

Suppose $v \in W_{t_i}$. If $v \in B_s$ there is nothing to show. If $s \lhd v$ then $v \in W_s$ as $t_i \lhd s$ is reachable from s and therefore $W_{t_i} \cap \{s, \ldots, n\} = \mathrm{supp}_{\lhd}(t_i) \cap \{s, \ldots, n\} \subseteq \mathrm{supp}_{\lhd}(s) \cap \{s, \ldots, n\} = W_s \cap \{s, \ldots, n\}$. Finally, suppose $v \lhd s$. But then, $v \in \mathcal{B}^{\downarrow}_s$ and hence $v \in \mathcal{B}^{\downarrow}_{t_j}$ for some $1 \leq j \leq k$. By definition of support sets, $v \notin \mathcal{B}^{\downarrow}_{t_i}$ and $t_i \lhd v$. But then, $v \notin \mathcal{B}^{\downarrow}_{t_j}$ for all $j \sqsupset i$, i.e. $j \lhd i$, as then $t_j \lhd v$ and by construction, $w \lhd t_j$ for all $w \in \mathcal{B}^{\downarrow}_{t_j}$. Hence, $v \in \mathcal{B}^{\downarrow}_{t_l}$ for some $t_l \rhd t_i$. This completes the proof of the theorem. □

The proof of Theorem 9 is constructive in that given an elimination ordering of width k it constructs a Kelly-decomposition of width $k+1$, and conversely. In fact, the proof establishes a slightly stronger statement.

Corollary 10. *Every digraph G of Kelly-width k has a Kelly-decomposition $\mathfrak{D} = (D, (B_t)_{t \in V(D)}, (W_t)_{t \in V(D)})$ of width k such that for all $t \in V(D)$:*

- $|B_t| = 1$,
- *W_t is the minimal set which guards $\mathcal{B}^{\downarrow}_t$, and*
- *every vertex $v \in \mathcal{B}^{\downarrow}_t$ is reachable in $G \setminus W_t$ from the unique $w \in B_t$.*

Further, if G is strongly connected, then D has only one root.

We call such a decomposition *special*.

4.2. Computing Kelly-decompositions

In this section we mention algorithms for computing Kelly-width and Kelly-decompositions. The proofs of Theorems 5 and 9 show that Kelly-decompositions can easily (i.e. polynomial time) be constructed from directed elimination orderings or monotone winning strategies, so we concern ourselves with the problem of finding any of the equivalent characterisations.

In a recent paper [6] Bodlaender et al. study exact algorithms for computing the (undirected) tree-width of a graph. Their algorithms are based on dynamic programming to compute an elimination ordering of the graph. In the same paper, the authors remark on actual experiments with these algorithms. Using some preprocessing techniques, the dynamic programming approach seems to perform reasonably well (in particular for not too large instances). The algorithms translate easily to directed elimination orderings and can therefore be used to compute Kelly-width. Hence, we get the following theorem.

Theorem 11. *The Kelly-width of a graph with n vertices can be determined in time $\mathcal{O}^*(2^n)$ and space $\mathcal{O}^*(2^n)$, or in time $\mathcal{O}^*(4^n)$ and polynomial space.*

Here, $\mathcal{O}^*(f(n))$ means that polynomial factors are suppressed.

For a given k, the problem whether a digraph G has Kelly-width $\leq k$ is decided in exponential time with the above algorithms. As the minimization problem is NP-complete (it generalises the NP-complete problem of deciding the tree-width of an undirected graph), we cannot expect polynomial-time algorithms to exist. It seems plausible though that, as in the case of DAG-width, studying strategies in the inert robber game will lead to a polynomial-time algorithm when k is fixed. This is part of ongoing research.

5. Applications

5.1. Algorithms on graphs of small Kelly-width

In this section we present algorithmic applications of the decomposition introduced above, including a general scheme that can be used to construct algorithms based on Kelly-decompositions. We assume that a Kelly-decomposition (or even an elimination ordering) has been provided or pre-computed. We mention two example

algorithms which run in polynomial time on graphs of bounded Kelly-width. The first is an algorithm for the NP-complete optimization problem of computing disjoint paths of minimal weight in weighted graphs. The second is an algorithm to compute the winner of certain forms of combinatorial games.

Similar to algorithms on graphs of small tree-width, Kelly-decompositions are suitable for dynamic programming style algorithms: starting with a special Kelly-decomposition $(D, (B_t)_{t\in V(D)}, (W_t)_{t\in V(D)})$, the algorithm works bottom up to compute for each node $t \in V(D)$ a data set containing information on the set $\mathcal{B}_t^{\downarrow} := \bigcup_{t'\geq t} B_{t'}$. The general pattern is therefore described by the following steps (after the special Kelly-decomposition has been computed):

Leaves: Compute the data set for all leaves.

Combine: If $t \in V(D)$ is an inner node with children $t_1, \ldots, t_p$ ordered by the ordering guaranteed by the Kelly-decomposition (we observe that such an ordering can be computed easily with a greedy algorithm), combine the data sets computed for $\mathcal{B}_{t_1}^{\downarrow}, \ldots, \mathcal{B}_{t_p}^{\downarrow}$ to a data set for the union $\bigcup_{1\leq i\leq p} \mathcal{B}_{t_i}^{\downarrow}$.

Update: Update the data set computed in the previous step so that the new vertex u with $B_t = \{u\}$ is taken into account. Usually, the vertex u will have been part of at least some guard sets W_{t_i}. As $u \notin W_t$, it can now be used freely.

Expand: Finally, expand the data set to include guards in $W_t \setminus \bigcup_i W_{t_i}$ and also paths etc. starting at u.

5.1.1. Weighted Hamiltonian cycle and disjoint paths.

A weighted digraph is a pair (G, ω) where G is a digraph and $\omega : V(G) \to \mathbb{R}$ is a weight function. The Kelly-width of (G, ω) is the Kelly-width of G. In [20] Johnson et al. provided a dynamic-programming algorithm for computing whether a digraph has a Hamiltonian cycle which is based on arboreal decompositions. This algorithm is readily extended to weighted digraphs and Kelly-decompositions, giving us the following:

Theorem 12 (*Essentially [20]*). *For any k, given a weighted digraph (G, ω) and a Kelly-decomposition $(D, (B_t)_{t\in V(D)}, (W_t)_{t\in V(D)})$ of G of width $\leq k$, there exists a polynomial-time algorithm which computes a Hamilton cycle of (G, ω) of minimal weight or determines that G is not Hamiltonian.*

The algorithm introduced above can easily be extended to solve the following, more general problem. The *weighted w-linkage problem* is the problem, given a weighted digraph (G, ω), a tuple $\boldsymbol{s} := ((s_1, t_1), \ldots, (s_w, t_w))$, and a set $M \subseteq \{1, \ldots, |V(G)|\}$, to compute for each $l \in M$ an $\boldsymbol{s}$-linkage of order l of minimal weight (among all $\boldsymbol{s}$-linkages of order l).

Theorem 13 (*Essentially [20]*). *For every $w, k \in \mathbb{N}$, given a weighted digraph (G, w) and a Kelly-decomposition of G of width $\leq k$, the weighted w-linkage problem can be solved in polynomial time.*

5.1.2. Parity games.

Another example for an algorithm on graphs of bounded Kelly-width is an algorithm for solving parity games on game arenas of small Kelly-width. Parity games are a form of combinatorial games played on digraphs with many applications in the area of verification. See [17] for a definition. It is well-known that deciding the winner of a parity game is in NP ∩ co-NP and it is a longstanding open problem if the problem is in P. In [4], Berwanger et al. describe an algorithm for computing the winner of a parity game of bounded DAG-width. This algorithm can easily be translated to arenas of small Kelly-width and, in some sense, becomes more efficient as the size of a Kelly-decomposition of width k is linear in the order of the graph, whereas DAG-decompositions of width k may contain n^k nodes.

Theorem 14. *For any k, given an arena $\mathcal{A}$ of a parity game and a Kelly-decomposition of $\mathcal{A}$ of width $\leq k$, the winning region of $\mathcal{A}$ can be computed in polynomial time.*

5.2. Asymmetric matrix factorization

The use of elimination orderings and elimination trees to investigate symmetric matrix factorizations is well-documented (see e.g. [22]). For example, the height of an elimination tree gives the parallel time required to factor a matrix [7]. In [16], Gilbert and Liu introduced a generalisation of elimination trees, called elimination DAGs, which

can be similarly used to analyse factorizations in the asymmetric case. Kelly-decompositions are closely related to these structures, as we show in this section.

Let $M = (a_{ij})$ be a square $n \times n$ matrix. We define G_M as the directed graph with $V(G_M) = \{v_1, \ldots, v_n\}$, and for $i \neq j$, $(v_i, v_j) \in E(G_M)$ if, and only if, $a_{ij} \neq 0$. We define $\lhd_M := (v_1, \ldots, v_n)$, and $\mathfrak{D}_M$ to be the Kelly-decomposition of G_M obtained by applying the proof of Theorem 9 with elimination ordering $\lhd_M$.

Definition 15 (*Upper and Lower Elimination DAGs [16]*). Let M be a square matrix that can be decomposed as $M = LU$ without pivoting. The *upper (lower) elimination DAG* is the transitive reduction of the directed graph G_U (G_L respectively).

The connection between Kelly-decompositions and elimination DAGs is reflected in the following observation.

Theorem 16. *Let M be a square matrix that can be decomposed as $M = LU$ without pivoting. Let $\mathfrak{D}_M = (D, (B_t)_{t\in V(D)}, (W_t)_{t\in V(D)})$. Then*

(a) *$(D, (B_t)_{t\in V(D)})$ is equivalent to the lower elimination DAG, and*
(b) *$G_U = (V(G_M), \{(v, w) : w \in W_v\})$, thus the upper elimination DAG is equivalent to the transitive reduction of the relation $\{(v, w) : w \in W_v\}$.*

Proof. For $v \in V(G_M)$, let $X_v = \{v\} \cup \{w \in V(G_M) : w \lhd_M v\}$, and let $G_v = G_M[X_v]$. We require two observations. First, from Theorem 1 of [27]:

$$(E(G_L))^{TC} = \{(v, w) : w \lhd_M v, \text{ and } v \in Reach_{G_v}(w)\}, \tag{1}$$

where R^{TC} denotes the transitive closure of R. Secondly, from Theorem 4.6 of [16], we have

$$E(G_U) = \{(v, w) : v \lhd_M w, \text{ and } \exists v' \in Reach_{G_v}(v) \text{ with } (v', w) \in E(G_M)\}. \tag{2}$$

Now the first result of the theorem follows from the observation that in the construction of the Kelly-decomposition, $(D, (B_t)_{t\in V(D)})$ is the transitive reduction of the right-hand side of (1). The second result follows from Lemma 2, which shows that $\{(v, w) : w \in W_v\} = \{(v, w) : w \in \text{supp}_{\lhd_M}(v)\}$ is equivalent to the right-hand side of (2). □

We can use the results of [16] to make the following observation when we construct Kelly-decompositions on undirected graphs.

Corollary 17. *Let G be an undirected graph, $\lhd$ an elimination ordering on G and $(D, (B_t)_{t\in V(D)}, (W_t)_{t\in V(D)})$ the Kelly-decomposition of G (considered as a bidirected graph) obtained by applying the proof of Theorem 9 with elimination ordering $\lhd$. Then D is a tree, and more precisely, $(D, (B_t)_{t\in V(D)})$ is equivalent to the elimination tree associated with the (undirected) elimination ordering $\lhd$.*

6. Is it better to be invisible but lazy or visible and eager?

In this section we use graph searching games to compare Kelly-width to DAG-width and directed tree-width. In the undirected case, all games require the same number of searchers, however we show that in the directed case there are graphs on which all three measures differ by an arbitrary amount. Our results do imply that Kelly-width bounds directed tree-width within a constant factor, but the converse fails as there are classes of graphs of bounded directed tree-width and unbounded Kelly-width. We also provide evidence to suggest that Kelly-width and DAG-width are within a constant factor of each other. We begin by introducing the games associated with DAG-width and directed tree-width (see [4,23,20] for formal definitions).

Definition 18 (*Visible Robber Game*). The *visible robber game* is played as the inert robber game except that the robber's position is always known to the cops and the robber is free to move during a cop transition irrespective of where the cops intend to move (however, he still cannot run through a stationary cop). The *strong visible robber game* adds the further restriction that the robber can only move in the same strongly connected component (of the graph with the stationary cops' locations removed). A *strategy* for the cops is a function that, given the current locations of the cops and the robber, indicates the next location of the cops. A strategy is *winning* if it captures the robber, and it is *monotone* if the set of vertices which the robber can reach is non-increasing.

The following theorem summarises the results of [4,23,20]. We refrain from giving the definitions and background on directed tree-decompositions and DAG-decompositions and refer to [20,4] instead.

Theorem 19. *Let G be a digraph.*

(1) *G has* DAG*-width k if, and only if, k cops have a monotone winning strategy in the visible robber game on G.*
(2) *G has directed tree-width $\leq 3k+1$ or k cops do not have a winning strategy in the strong visible robber game on G.*

Our first result shows that a monotone winning strategy in the inert robber game can be translated to a (not necessarily monotone) winning strategy in the visible robber game.

Theorem 20. *If k cops can catch an inert robber with a robber-monotone strategy, then $2k-1$ cops can catch a visible robber.*

Proof. Suppose k cops have a robber-monotone winning strategy on a graph G. By Theorem 5 this implies that there is a directed elimination ordering $\lhd$ on G of width $\leq k-1$. We use the elimination ordering to describe the winning strategy of $2k-1$ cops against a mobile, visible robber, thereby establishing the result.

The cops are split into two groups, k cops called the *blockers* and $k-1$ cops called the *chasers*. Similarly, the cop moves are split in two phases, a blocking move and a chasing phase.

In the first move, k cops are placed on the k highest elements with respect to $\lhd$. These cops form the set of blockers. Let the robber choose some element v. This concludes the first (blocking) move. We observe:

> If u is the $\lhd$-smallest vertex occupied by a blocker, then every directed path from v to a vertex greater than u has at least one vertex occupied by a cop. $(*)$

This invariant is maintained by the blocking cops during the play. Now suppose after r rounds have been played, the robber occupies vertex v and the blockers occupy vertices in X so that the invariant $(*)$ is preserved. Let u be the $\lhd$-smallest element in X and let $C_1, \ldots, C_s$ be the set of strongly connected components of $G[\{u' : u' \lhd u\}]$. Further, let $\sqsubset$ be a linear ordering on $\mathcal{C} := \{C_1, \ldots, C_s\}$ so that $C_i \sqsubset C_j$ if, and only if, the $\lhd$-maximal element in C_i is $\lhd$-smaller than the $\lhd$-maximal element of C_j. Now the cops move as follows. Let $C \in \mathcal{C}$ be the component such that $v \in C$ and let $w \in C$ be the $\lhd$-maximal element in C. The cops place the $k-1$ cops not currently on the graph on $\mathrm{supp}_{\lhd}(w)$. These cops are the chasers. Seeing the chasers approach, the robber has two options. Either he stays within C or he escapes to a vertex in a different strongly connected component C'. If the robber runs to a vertex $x \in C$ or $x \in C'$ for some $C' \sqsubset C$ then after the chasers land on $S := \mathrm{supp}_{\lhd}(w)$ there is no path from x to a node u such that $u \rhd u'$ for the $\lhd$-minimal vertex u' in S. Hence, the chasers become blockers and the chasing phase is completed. Otherwise, if the robber escapes to a C' with $C \sqsubset C'$, then the chasers repeat the procedure and move to $\mathrm{supp}_{\lhd}(w')$ for the $\lhd$-maximal element in C'. However, as the robber always escapes to a $\sqsubset$-larger strongly connected component and also can not bypass the blockers, this chasing phase must end after finitely many steps with the robber being on a vertex $v \in C$ for some component C and the chasers being on $\mathrm{supp}_{\lhd}(w)$ for the $\lhd$-maximal element in C. At this point the chasers become blockers. One of the old blockers is now placed on w and all others are removed from the board. The cop on w makes sure that in each such step the robber space shrinks by at least one vertex. By construction, the invariant in $(*)$ is maintained. Further, as the robber space shrinks by at least one after every chasing-phase, the robber is eventually caught by the cops. □

One consequence of this theorem is that Kelly-width bounds directed tree-width by a constant factor.

Corollary 21. *If G has Kelly-width $\leq k$ then G has directed tree-width $\leq 6k-2$.*

Since it is not known whether monotone strategies are sufficient in the visible robber game, we cannot obtain a similar bound for DAG-width. We can, however, ask whether we can improve the bound, i.e. assuming that k cops have a robber-monotone winning strategy against an invisible, inert robber can we define a winning strategy for less than $2k-1$ cops in the visible robber game? Although it might be possible to improve the result, the next theorem shows that we cannot do better than with $\frac{4}{3}k$ cops.

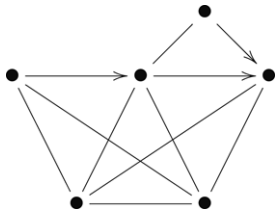

Fig. 1. Graph G to show difference between DAG-width and inert robber game.

Theorem 22. *For every $k \in \mathbb{N}$, there is a digraph such that $3k$ cops have a robber-monotone winning strategy in the inert robber game but no fewer than $4k$ cops can catch a visible robber.*

Before we prove this result we need to introduce the idea of lexicographic product.

Definition 23 (*Lexicographic Product*). Let G, H be graphs. The *lexicographic product* $G \bullet H$ of G and H is defined as the graph with vertex set $V(G \bullet H) := V(G) \times V(H)$ and edge set

$$E(G \bullet H) := \{\big((x, y), (x', y')\big) : (x, x') \in E(G) \text{ or } x = x' \text{ and } (y, y') \in E(H)\}.$$

The lexicographic product is also known as *graph composition* as $G \bullet H$ can also be viewed as a graph obtained from G by replacing vertices by copies of H. This observation is useful for the following proposition:

Proposition 24. *Consider the cops and robber game on a directed graph G, and let $\mathcal{K}_n$ be the complete digraph on n vertices. Then at least k cops have a winning strategy on G if, and only if, at least $n \cdot k$ cops have a winning strategy on $G \bullet \mathcal{K}_n$.*

Proof. If k cops have a winning strategy on G, then a winning strategy for $n \cdot k$ cops on $G \bullet \mathcal{K}_n$ is obtained by simulating the game on G. If the robber's position is $(r, s) \in V(G \bullet \mathcal{K}_n)$ then we position a robber on $r \in V(G)$. We then consider the cops' play on G and play on $G \bullet \mathcal{K}_n$ by placing n cops on $\{(x, y) : y \in V(\mathcal{K}_n)\}$ whenever a cop would be placed on $x \in V(G)$.

For the converse we show that if the robber can defeat $k - 1$ cops on G then he can defeat $nk - 1$ cops on $G \bullet \mathcal{K}_n$. Again we simulate the game for $G \bullet \mathcal{K}_n$ on G, but this time from the robber's perspective. We place a cop on $x \in V(G)$ only if all vertices in $V(G \bullet \mathcal{K}_n)$ of the form (x, y), $y \in V(\mathcal{K}_n)$ are occupied. By the pigeon-hole principle, this places at most $k - 1$ cops on G. The robber's current position is projected as before. The robber's response r' on G is lifted to $G \bullet \mathcal{K}_n$ by playing to an unoccupied vertex of the form (r', y). As r' is unoccupied in the simulated game, at least one such vertex exists. We need to be careful if the projected play is to remain at the same vertex because the robber's position may become occupied. But as the projected vertex remains unoccupied, there is at least one unoccupied vertex in the block isomorphic to $\mathcal{K}_n$ and so the robber is able to run to that vertex. As the robber can defeat $k - 1$ cops on G, the strategy is winning. □

It is worth observing that Proposition 24 holds regardless of the visibility or mobility of the robber, as well as when the cops are restricted to monotone strategies, giving us the following:

Corollary 25. *For any directed graph G:*

(i) *DAG-width*$(G \bullet \mathcal{K}_n) = n \cdot$ *DAG-width*(G).
(ii) *Kelly-width*$(G \bullet \mathcal{K}_n) = n \cdot$ *Kelly-width*(G).

We now use this result to complete the proof of Theorem 22.

Proof. Consider the graph G in Fig. 1. It is easy to see that on G, 3 cops do not have a (non-monotone winning) strategy to catch a visible robber, however 4 cops do. On the other hand, 3 cops suffice to capture an invisible, inert robber with a robber-monotone strategy. The result follows by taking the lexicographic product of this graph with the complete graph on k vertices. □

In fact, 4 cops can capture a visible robber with a monotone strategy on the graph in the previous proof, giving us the following:

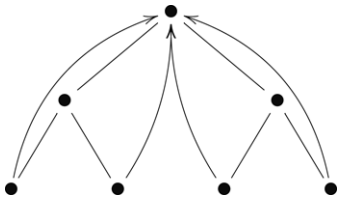

Fig. 2. Binary tree with back-edges of height 2.

Corollary 26. *For all $k \geq 1$ there are graphs of* DAG*-width* $4k$ *and Kelly-width* $3k$.

Despite this $\frac{4}{3}$ bound, for graphs of small Kelly-width we can do better.

Theorem 27. *For $k = 1$ or 2, if G has Kelly-width k, G has DAG-width k.*

Proof. If G has an elimination ordering of width 0 then it must be acyclic, as all support sets are empty. Thus it has DAG-width 1. If G has an elimination ordering $\lhd = (v_1, v_2, \ldots, v_n)$ of width 1 then a cop-monotone strategy for two cops against a visible robber is as follows. Initially, let $i = n$ and place one cop on v_i. At this point, the robber is restricted to $\{v_1, \ldots, v_{i-1}\}$. Let $j < i$ be the maximal index such that the robber can reach v_j. Place a cop on v_j. After the cop has landed, we claim that the robber is unable to reach both v_i and v_j. For otherwise, let r be the maximal index such that the robber can reach v_r (with cops on v_i and v_j) and from v_r can reach v_i (with a cop on v_j) and v_j (with a cop on v_i). By the maximality of j, $r < j$. Let $s > r$ be the first index greater than r which occurs on a path from v_r to v_i that does not go through v_j, and $t > r$ be the first index greater than r which occurs on a path from v_r to v_j that does not go through v_i. Then from the maximality of r, $s \neq t$. Furthermore, $\{v_s, v_t\} \subseteq \mathrm{supp}_{\lhd}(r)$, so $|\mathrm{supp}_{\lhd}(v_r)| > 1$, contradicting the width of the ordering. So we can remove the cop from whichever vertex the robber can no longer reach without changing the robber space, and either the robber is now restricted to $\{v_1, \ldots, v_j\}$ or the maximal index which the robber can reach is smaller. Clearly, this is a monotone winning strategy for two cops. □

We now turn to the converse problem, what can be said about the Kelly-width of graphs given their directed tree-width or DAG-width? First, we consider the binary tree with back-edges example in [4], where it was shown this class of graphs has bounded directed tree-width but unbounded DAG-width. A "binary tree with back-edges" is a balanced binary tree of height k, for some k, where all edges are oriented from the root towards the leaves. In addition every vertex has a directed edge to each of its predecessors on the unique path from the root to itself. Fig. 2 shows a tree with back-edges of height 2.

It is readily shown that this class of graphs also has unbounded Kelly-width.

Theorem 28. *There exists classes of digraphs with bounded directed tree-width and unbounded Kelly-width.*

Our final result is a step towards relating Kelly-width to DAG-width by showing how to translate a monotone strategy in the visible robber game to a (not necessarily monotone) strategy in the inert robber game.

Theorem 29. *If G has* DAG*-width $\leq k$, then k cops have a winning strategy in the inert robber game.*

Proof. Given a DAG-decomposition $(D, (X_d)_{d \in V(D)})$ of G of width k, the strategy for k cops against an invisible, inert robber is to follow a depth-first search on the decomposition. More precisely, we assume the decomposition has a single root r, and we have an empty stack of nodes of D.

(1) Initially, place the cops on X_r and push r onto the stack.
(2) At this point we assume d is on the top of the stack and the cops are on X_d. We next "process" the successors of d in turn. To process a successor d' of d, we remove all cops not on $X_d \cap X_{d'}$, place cops on $X_{d'}$, push d' onto the stack, and return to step 2. Note that a node may be processed more than once.
(3) Once all the successors of a node have been processed, we pop the node off the stack and if the stack is non-empty, return to step 2.

Because the depth-first search covers all nodes of the DAG and hence all vertices of the graph are eventually occupied by a cop, the robber will be forced to move at some point. Due to the guarding condition for DAG-decompositions, when the robber is forced to move this strategy will always force the robber into a smaller region and eventually capture him. □

Again we observe that it is unknown if monotone strategies suffice in the inert robber game, so this result does not allow us to compare Kelly-width and DAG-width.

7. Open problems and further remarks

Let us first remark on the following recent result. In [21], Kreutzer and Ordyniak give examples showing that neither the visible robber game associated with DAG-width nor the inert robber game associated with Kelly-width are monotone. The examples show that for each $k \in \mathbb{N}$ there are graphs where the difference between the number of cops needed for monotone and non-monotone winning strategies differ by k cops. However, it is not clear if the monotonicity cost can be increased any further or if there is a constant $c > 1$ such that whenever k cops have a winning strategy in one of the games then $c \cdot k$ cops have a monotone winning strategy.

Note that if there was such a constant factor giving an upper bound for the monotonicity costs in both games, then the Kelly-width and the DAG-width of a digraph would be within constant factors of one another. We believe that this is the case and therefore propose the following conjecture.

Conjecture 30. *The Kelly-width and* DAG*-width of a graph lie within constant factors of one another.*

In [20], an min-max theorem between directed tree-width and the number of cops required to catch a robber in the visible robber game where the robber moves in strong components is given. However, this theorem is only up to a constant factor and for various reasons there is no hope to make it exact. It might be possible, though, to define a modified version of directed tree-width that does have an exact min-max theorem with non-monotone strategies. One natural candidate is the version of directed tree-width given in [19], to which the reasons ruling out such an exact theorem for the standard definition do not apply. This is part of ongoing research.

References

[1] I. Adler, Directed tree-width examples, Journal of Combinatorial Theory, Series B 97 (5) (2007) 718–725.
[2] B. S. Baker, Approximation algorithms for NP-complete problems on planar graphs, Journal of the ACM 41 (1) (1994) 153–180.
[3] J. Barát, Directed path-width and monotonicity in digraph searching, Graphs and Combinatorics 22 (2) (2006) 161–172.
[4] D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, DAG-width and parity games, in: Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS), 2006, pp. 524–536.
[5] H. Bodlaender, Treewidth: Algorithmic techniques and results, in: Proceedings of the 22nd International Symposium on Mathematical Foundations of Computer Science (MFCS), 1997, pp. 19–36.
[6] H. Bodlaender, F. Fomin, A. Koster, D. Kratsch, D. Thilikos, On exact algorithms for treewidth, in: Proceedings of the 14th Annual European Symposium on Algorithms (ESA), 2006.
[7] H. Bodlaender, J. Gilbert, H. Hafsteinsson, T. Kloks, Approximating treewidth, pathwidth, frontsize, and shortest elimination tree, Journal of Algorithms 18 (2) (1995) 238–255.
[8] B. Courcelle, Graph rewriting: An algebraic and logic approach, in: J. van Leeuwan (Ed.), Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B), 1990, pp. 193–242.
[9] B. Courcelle, S. Olariu, Upper bounds to the clique width of graphs, Discrete Applied Mathematics 1–3 (2000) 77–114.
[10] E. Demaine, M. Hajiaghayi, K. Kawarabayashi, Algorithmic graph minor theory: Decomposition, approximation, and coloring, in: Proceedings of the 46th IEEE Symposium on Foundations of Computer Science (FOCS), 2005, pp. 637–646.
[11] N. Dendris, L. Kirousis, D. Thilikos, Fugitive-search games on graphs and related parameters, Theoretical Computer Science 172 (1–2) (1997) 233–254.
[12] R. Diestel, Graph Theory, 3rd edition, Springer, 2005.
[13] F. Dorn, E. Penninkx, H. Bodlaender, F. Fomin, Efficient exact algorithms on planar graphs: Exploiting sphere cut branch decompositions, in: Proceedings of the 13th Annual European Symposium on Algorithms (ESA), 2005, pp. 95–106.
[14] D. Eppstein, Subgraph isomorphism in planar graphs and related problems, Journal of Graph Algorithms and Applications 3 (3) (1999) 1–27.
[15] F.V. Fomin, P. Heggernes, J.A. Telle, Graph searching, elimination trees, and a generalization of bandwidth, Algorithmica 41 (2) (2004) 73–87.
[16] J. Gilbert, J. Liu, Elimination structures for unsymmetric sparse LU factors, SIAM Journal of Matrix Analysis and Applications 14 (1993) 334–352.
[17] E. Grädel, W. Thomas, T. Wilke (Eds.), Automata Logics, and Infinite Games, in: LNCS, vol. 2500, Springer, 2002.
[18] P. Hunter, S. Kreutzer, Digraph Measures: Kelly Decompositions, Games, and Orderings, in: Proc. of the Eighteenth ACM-SIAM Symp. on Discrete Algorithms (SODA), 2007, pp. 637–644.
[19] T. Johnson, N. Robertson, P. Seymour, R. Thomas, Addendum to "directed tree-width", unpublished manuscript, 2001.
[20] T. Johnson, N. Robertson, P. Seymour, R. Thomas, Directed tree-width, Journal of Combinatorial Theory, Series B 82 (1) (2001) 138–154.
[21] S. Kreutzer, S. Ordyniak, Digraph Decompositions and Monotonicity in Digraph Searching, unpublished manuscript, 2007.
[22] J.W.H. Liu, The role of elimination trees in sparse factorization, SIAM Journal of Matrix Analysis and Applications 11 (1) (1990) 134–172.

[23] J. Obdržálek, DAG-width: Connectivity measure for directed graphs, in: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2006, pp. 814–821.
[24] S.-I. Oum, Rank-width and vertex-minors, Journal of Combinatorial Theory, Series B 95 (2005) 79–100.
[25] S.-I. Oum, P. Seymour, Approximating clique-width and branch-width, Journal of Combinatorial Theory, Series B 96 (4) (2006) 514–528.
[26] B. Reed, Introducing directed tree width, in: 6th Twente Workshop on Graphs and Combinatorial Optimization, in: Electronic Notes in Discrete Mathematics, vol. 3, Elsevier, 1999.
[27] D. Rose, R. Tarjan, Algorithmic aspects of vertex elimination on directed graphs, SIAM Journal of Applied Mathematics 34 (1) (1978) 176–197.
[28] D. J. Rose, Triangulated graphs and the elimination process, Journal of Mathematical Analysis and Applications 32 (1970) 597–609.
[29] M. Safari, D-width: A more natural measure for directed tree width, in: Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS), 2005, pp. 745–756.