Digraph Decompositions and Monotonicity in Digraph Searching

Stephan Kreutzer and Sebastian Ordyniak

Oxford University Computing Laboratory {kreutzer,ordyniak}@comlab.ox.ac.uk

Abstract. We consider graph searching games on directed graphs and corresponding digraph decompositions. In particular we show that two important variants of these games – underlying DAG- and Kelly-decompositions – are non-monotone.
Furthermore, we explore the limits of algorithmic applicability of digraph decompositions and show that various natural candidates for problems, which potentially could benefit from digraphs having small "directed width", remain NP-complete even on almost acyclic graphs.

1 Introduction

The seminal work of Robertson and Seymour in their graph minor project has focused much attention on graph decompositions and associated measures of graph connectivity such as tree- or path-width. Aside from the interest in graph structure theory, these notions have also proved fruitful in the development of algorithms.

Intuitively, tree-width measures the similarity of a graph to a tree. Thus trees have tree-width one and graphs of small tree-width can be decomposed into parts with at most tree-width (plus one) vertices in a tree-like manner. Similarly to trees, tree-decompositions allow for recursive algorithms, whose running time is linear in the size of the underlying graph – but exponential in its width. Together with linear time parameterized algorithms for constructing tree-decompositions, this implies that a huge number of NP-complete problems become tractable on graph classes of bounded tree-width (see [8,7] for a survey on tree-width).

Graph Searching Games. Closely related to tree-width (and path-width) are so called graph searching games. Graph searching games are played by two players, the searcher and the fugitive, that simultaneously place tokens on the vertices of a graph. Whereas the fugitive has only one token and is restricted to move along paths in the graph that are not occupied by a searcher, the searcher controls an arbitrary amount of tokens and is free to move them anywhere on the graph. The aim of the searcher is to capture the fugitive, i.e. to force him into a position where he is not able to move any more. The minimum number of tokens needed by the searcher to capture the fugitive defines a natural graph invariant.

Within this general framework, there exist a range of variants defined by different abilities for both players. In particular one distinguishes between the *visible* and *invisible* variant. In the visible case, the searchers can see the fugitive and can adapt their strategy accordingly. In the invisible case, the fugitive's position is hidden from

the searcher. Concerning the abilities of the fugitive one distinguishes between the so called *inert* variant, where the fugitive is only allowed to move if a searcher is placed on his current position, and the *dynamic* variant, where the fugitive can move in any step of the play. Combining this yields four main variants of which only three will be considered in this paper: visible and dynamic (*vis*), invisible and inert (*inert*), and invisible and dynamic (*invis*). The forth variant, visible and inert has recently been studied by Richerby and Thilikos [24].

An important concept in the theory of graph searching games is monotonicity. A game is *monotone*, if whenever k searchers can catch a fugitive on a graph they can do so without allowing the fugitive to re-occupy vertices. In general, restricting the searchers to monotone strategies may require additional searchers. LaPaugh [18] gave a first proof of monotonicity for a graph searching game. Since then, monotonicity has been intensely studied and a large number of monotonicity results have been established. See e.g. [18,6,9,3,12,13,19,27] or the survey [2] and references therein.

The importance of monotonicity in the context of graph decompositions results from the observation that many decompositions, like tree- and path-decompositions, can be defined in terms of monotone winning strategies for the searcher. Monotonicity for a game is often established through duality theorems for the underlying decomposition. Strategies for the fugitive provide the dual notion for the existence of a decomposition and yield natural obstructions for graphs having small decompositions. For example, the notion of a *bramble* is a natural formalisation of a winning strategy for the fugitive and provides an important obstruction for small tree-width (see [11,20]).

Despite the considerable interest and the large number of results in this field, two cases have so far resisted any attempts to solve the monotonicity problem – the graph searching game with a visible, dynamic fugitive and the game with an invisible, inert fugitive, both played on digraphs. It is these games that are closely related to DAG- and Kelly-decompositions [4,15]. In this paper,we solve the problems by showing that both games are non-monotone.

Digraph decompositions. In recent years, attempts have been made to generalise the notion of tree-decompositions and their algorithmic applications to directed graphs. Clearly, we can define the tree-width of a directed graph as the tree-width of the undirected graph we get by ignoring 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 we know that this can be solved easily on graphs with small tree-width, there are directed graphs with very simple connectivity structure which have large tree-width. Therefore, several proposals have been made to extend the notions of tree-decompositions and tree-width to directed graphs (see [23,16,3,5,25,15]). In particular, Reed [23] and Johnson, Robertson, Seymour, and Thomas [16] introduce the notion of *directed tree-width* and they show that Hamiltonicity can be solved for graphs of bounded directed tree-width in polynomial time.

Following this initial paper, several alternative definitions of directed graph decompositions have been proposed, with the aim of overcoming some shortcomings of the original definition. Berwanger, Dawar, Hunter and Kreutzer [4] and Obdržàlek [22] introduce the notion of DAG-width and Hunter and Kreutzer [15] introduce the notion of

Kelly-width. All three proposals are supported by algorithmic applications and various equivalent characterisations in terms of obstructions, elimination orderings, and, in particular, variants of graph searching games on directed graphs. However, so far the algorithmic applications are restricted to few classes of problems, in particular the problem of finding disjoint paths, Hamiltonian-cycles and similar linkage problems, and certain problems in relation to combinatorial games (parity games) played on graphs motivated by the theory of computer-aided verification. Whereas the tree-width of undirected graphs has been employed to solve a huge number of problems on graphs of small tree-width, the algorithmic theory of directed graph decompositions is not nearly as rich.

It is an obvious question whether this is due to the fact that digraph decompositions are a relatively new field of research, where the fundamental machinery first needs to be developed, or whether this is due to a general limitation of this approach to algorithms on digraphs. In this paper we systematically explore the range of algorithmic applicability of digraph decompositions. For this, we look at typical NP-hard problems on graphs – as they can be found in [14], for instance – and identify those that are "suitable" for this approach, where by "suitable" we mean that the problems should be NP-hard in general but tractable on acyclic digraphs. The reason for the latter is that all digraph decompositions proposed so far measure in some way the similarity of a graph to being acyclic. In particular, acyclic graphs have small width in all of these measures. Hence, if a problem is already hard on acyclic digraphs, there is no point in studying the effect of digraph decompositions on this problem. We then identify representatives for the various types of "suitable" problems and ask whether they can be solved in polynomial time on graphs of small directed tree-, Kelly- or DAG-width or of small directed path-width.

The results we present in Section 4 show that the border for algorithmic applicability of digraph decompositions is rather tight. Essentially, as far as classical graph theoretical problems are concerned, disjoint paths and Hamiltonian-cycles can be detected efficiently on graphs of small directed tree-width, but all other problems we considered such as Minimum Equivalent Subgraph, Feedback Vertex Set (FVS), Feedback Arc Set, Graph Grundy Numbering, and several others are NP-complete even on graphs with a very low global connectivity and thus very low directed path or tree-width.

Organisation. The paper is organised as follows. In Section 2 we briefly recall basic notions from graph and game theory needed later. In Section 3 we give a formal description of graph searching games and present the first main result of this paper, the non-monotonicity of the two types of games mentioned above. In Section 4 we explore the algorithmic boundaries of the digraph decompositions known so far by showing NP-completeness for a number of problems on digraphs with bounded "width". We conclude and state some open problems in Section 5.

2 Preliminaries

We use standard notation from graph theory as can be found in, e.g., [10]. All graphs and directed graphs in this paper are finite and simple.

Let G be a (directed) graph. We denote the vertex set of G by $V(G)$ and the edge set of G by $E(G)$. For $X \subseteq V(G)$ we denote by $G[X]$ the subgraph of G induced by X and by $G \setminus X$ the subgraph of G induced by $V(G) \setminus X$. Similarly for $Y \subseteq E(G)$ we set $G \setminus Y$ to be the subgraph of G obtained by deleting all edges in Y.

Finally, if X is a set and $k \in \mathbb{N}$, we denote by $[X]^{\leq k}$ the set of all subsets of X of cardinality $\leq k$.

3 Graph Searching Games

In this section we show non-monotonicity of two important variants of graph searching on directed graphs, namely the variants underlying DAG- and Kelly-decompositions.

Graph searching games are played by two players – the searcher and the fugitive – placing tokens on the vertices of a graph. Whereas the fugitive has only one token and can only move along paths in the graph that are not blocked by a searcher, the searcher controls an arbitrary amount of tokens and is free to move them anywhere on the graph. That is, in any step of the play, the searchers can place new tokens or remove existing tokens from the board. A play begins with the fugitive choosing his initial position. In each step, the searchers first announce their intended move. The fugitive can then react to this by choosing his new position, as long as there is a path from his current to the new position that does not contain a vertex occupied by a searcher remaining on the board.

The aim of the searcher is to capture the fugitive, i.e. to force him into a position where he is not able to move any more. The minimum number of tokens needed by the searcher to capture the fugitive defines the graph invariant that we are interested in.

More formally, let G be an undirected graph. A position in the game is a pair (X, r), with $X \subseteq V(G)$ and $r \in V(G)$, and a play is a sequence of positions $((X_1, r_1), \ldots, (X_n, r_n))$, such that $X_1 = \emptyset$ and a move from one position to another is legal, if there is a path from r_i to r_{i+1} in $G \setminus (X_i \cap X_{i+1})$. A play is winning for the searcher if $r_n \in X_n$, otherwise it is winning for the fugitive.

Within this general framework, there exist a range of variants defined by different abilities for both players. In particular one distinguishes between the *visible* and *invisible* variant. In the visible case, the searchers can see the fugitive and can adapt their strategy accordingly. In the invisible case, the fugitive's position is hidden from the searcher. Concerning the abilities of the fugitive one distinguishes between the so called *inert* variant, where the fugitive is only allowed to move if a searcher is placed on his current position, and the *dynamic* variant, where the fugitive can move in any step of the play. Combining these variants yields four main variants of which only three will be considered in this paper: visible and dynamic (*vis*), invisible and inert (*inert*), and invisible and dynamic (*invis*).

We are mainly interested in the type of *strategies* the searcher can employ. One can easily verify that strategies in these games only depend on the current position in the game, i.e. are deterministic and positional. Basically, there exist two types of strategies for the searcher, depending on whether or not the fugitive is visible. In the visible case, the searcher can take the position of the fugitive into account and thus a strategy is a function $f : (X, r) \to X'$ assigning a new position X' to the searcher depending on the

current position (X, r) in the game. In the invisible case, a strategy can simply be seen as a sequence of positions for the searcher. A strategy for the searcher is *winning* if all plays consistent with this strategy are, i.e. plays where the searcher always chooses the move defined by the strategy.

Let $P = ((X_1, r_1), \cdots, (X_n, r_n))$ be a play. We define the *search-width* of P, denoted by sw(P), to be sw$(P) := \max_{1 \leq i \leq n} |X_i|$. Similarly, we define the search-width of a strategy to be the maximum search-width of all plays consistent with that strategy and the search-width of a graph G, to be sw$(G) := \min\{\text{sw}(f) : f$ is a winning strategy on $G\}$. Thus the search-width of a graph defines the graph invariant that we are interested in.

We are now ready to define two important properties of a graph searching game namely fugitive- and searcher-monotonicity. We say a play is *fugitive-monotone* if the fugitive is not able to reach a vertex from which he has previously been expelled. Thus in a fugitive-monotone play the set of vertices that the fugitive can reach is not increasing. A play is *searcher-monotone* if the searcher never reoccupies a previously vacated vertex. On undirected graphs, both notions are closely related: every searcher-monotone play that is winning for the searcher is also fugitive-monotone and for every fugitive-monotone play that is winning for the searcher there is a searcher-monotone play that uses the same amount of searchers. It is thus not always necessary to distinguish between both notions and we say a play is *monotone* if it is both fugitive- and searcher-monotone.

The notion of monotonicity directly applies to strategies for the searcher, so we say that a strategy is fugitive-monotone, searcher-monotone or just monotone, if all plays consistent with that strategy are. Let G be a graph. We define mon-sw$(G) := \min\{sw(f) : f$ is monotone and winning on $G\}$ and say that a game is *monotone*, if mon-sw$(G) :=$ sw(G) for all graphs G.

On undirected graphs all three variants we consider in this paper are monotone and satisfy:

1. vis-sw$(G) =$ inert-sw$(G) =$ tw$(G) + 1$, for every graph G, where tw(G) denotes the tree-width of G (see [11] and [9]).
2. invis-sw$(G) =$ pw$(G)+1$, for every graph G, where pw(G) denotes the path-width of G (see [6]).

Depending on how one translates the notion of an undirected path to the directed setting, i.e. whether one regards it as a directed path from source to destination or as two directed paths, one in each direction, there are two natural variants of this game on directed graphs. We refer to the first variant, where the fugitive is allowed to move along (searcher-free) directed paths, as *reachability* variant (*reach*), and to the second one, where the fugitive is only allowed to move when there exist a path in each direction, as *strongly connected component* (*scc*) variant, as in this case the fugitive is only allowed to move in strongly connected components.

Combining these two ways of defined games on directed graphs with the variants discussed for the undirected setting yields a number of interesting games on directed graphs of which the following have been discussed in literature: strongly connected component, visible and dynamic (*scc-vis*); reachability, visible and dynamic (*reach-vis*); reachability, invisible and dynamic (*reach-invis*); and reachability, invisible and

inert (*reach-inert*). We briefly relate these games to the corresponding digraph decompositions and recall what is known about monotonicity.

scc, visible, and dynamic. This variant is closely related to directed tree-width as it is known that scc-vis-sw$(D) - 1 \leq dtw(D) \leq 3 \cdot$ scc-vis-sw$(D) + 5$, for every digraph D, where dtw(D) is the directed tree-width as defined in [16]. It has been shown to be neither fugitive- nor searcher-monotone [1,16]. However, although not explicitly stated, [16] gives an upper bound for the monotonicity costs with respect to fugitive-monotonicity. It remains an interesting open question whether this holds for the searcher-monotone variant as well.

reachability, invisible, and dynamic. This variant defines directed path-width and has been shown to be monotone in [3].

reachability, visible, and dynamic. The monotone version of this variant defines DAG-width [4,22]. We therefore refer to these games as *DAG-games* and write dag-sw(D) and mon-dag-sw(D) for the non-monotone and monotone search-width of a graph D, with respect to this variant.

reachability, invisible, and inert. The monotone version of this variant defines Kelly-width [15]. We therefore refer to these games as *Kelly-games* and write kelly-sw(D) and mon-kelly-sw(D) for the non-monotone and monotone search-width of a graph D, with respect to this variant.

We are now ready to state our main results of this section, proving that DAG- and Kelly-Games are non-monotone.

3.1 Non-Monotonicity of DAG-Games

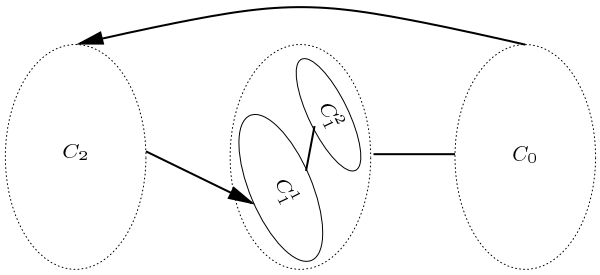

Fig. 1. The graph D_p with dag-sw$(D_p) \neq$ mon-dag-sw(D_p).

Theorem 31 *For every $p \geq 2$ there exists a digraph D_p with* mon-dag-sw$(D_p) = 4p - 2$ *and* dag-sw$(D_p) = 3p - 1$.

Proof. A schematic overview of D_p is given in Figure 1. The graph consists of three main parts with $2p - 1$ vertices each. C_0 and C_2 are cliques on $2p - 1$ vertices, C_1^2 is a

clique on $p-1$ vertices and C_1^1 forms an independent set having p vertices. A directed edge between two parts A and B means that there are edges from every vertex in A to every vertex in B. Undirected edges mean that there are edges between A and B in both directions. Thus there are edges in both directions between C_1^1 and C_1^2, and between C_0 and $C_1^1 \cup C_1^2$. Furthermore there are edges from C_0 to C_2, and edges from C_2 to C_1^1.

It is easy to see that dag-sw$(D_p) \geq 3p-1$ since the vertices in $C_0 \cup C_1^2$ together with a vertex of C_1^1 form a clique of size $3p-1$. To show that dag-sw$(D_p) \leq 3p-1$ consider the following strategy for $3p-1$ searchers on D_p. In the first move the searchers occupy $C_0 \cup C_1^1$. If the fugitive plays to C_2 the searchers capture him by playing on $C_1^1 \cup C_2$. Otherwise, if the fugitive plays to C_1^2 the searchers move to $C_0 \cup C_1^2$. Now the fugitive has to be on a vertex $v \in C_1^1$. Since the vertices in C_1^1 form an independent set the fugitive is now captured by playing to $\{v\} \cup C_1^2 \cup C_0$.

It remains to show that mon-dag-sw$(D_p) = 4p-2$. It is easy to see that $4p-2$ searchers can capture the fugitive on D_p by playing $C_0 \cup C_2$ and then $C_0 \cup C_1^1 \cup C_1^2$. To show that mon-dag-sw$(D_p) \geq 4p-2$ we give a strategy for the fugitive against $4p-3$ searchers playing monotonously on D_p.

First the fugitive stays in C_0 until the searchers occupy all vertices of C_0. There are two cases to consider.

1. The searchers occupy (at least) $C_0 \cup C_1^1$. In this case there is a vertex $v \in C_1^2$ which is not occupied by a searcher and which the fugitive can reach from his current position in C_0. Since every $v \in C_1^2$ has an edge to every other vertex in $C_0 \cup C_1^1 \cup C_1^2$ the searcher cannot capture the fugitive monotonously with less than $4p-2$ searchers.
2. The searchers occupy (at least) C_0 and there is at least one vertex in C_1^1 which is not occupied by a searcher. Then there exists a vertex $v \in C_2$ which is not occupied by a searcher and which the fugitive can reach from his current position in C_0. Since from every vertex in C_2 there is a path to every other vertex in the graph (as long as there is at least one vertex in C_1^1 not occupied by a searcher) the fugitive can stay in C_2 until the searchers occupy all vertices in C_1^1. And if they do the fugitive can move to a vertex in C_1^2 and play as in the first case. □

3.2 Non-Monotonicity of Kelly-Games

We now consider Kelly-games. Recall that in a Kelly-game, the fugitive is invisible. Hence, a strategy must be independent of the current position of the fugitive. We can therefore represent a searcher-strategy in a digraph D by a sequence $(X_1, \ldots, X_n)$ of searcher-positions. Furthermore fugitive-monotone strategies can simply be given by a sequence of vertices $(v_1, \ldots, v_{|D|})$, reflecting the order in which the vertices become cleared by the searcher. Note also that since Kelly-games are also inert, the notion of searcher-monotonicity cannot be applied.

Theorem 32 *For every $p \geq 2$ there exists a digraph D_p with* kelly-sw$(D_p) = 6p$ *and* mon-kelly-sw$(D_p) = 7p$.

Proof. A schematic overview of D_p is given in Figure 2. The graph consists of five cliques with $|C_0| = p$, $|C_2| = |C_1| = |X_1| = 2p$, $|X_2| = 3p$. An edge between two

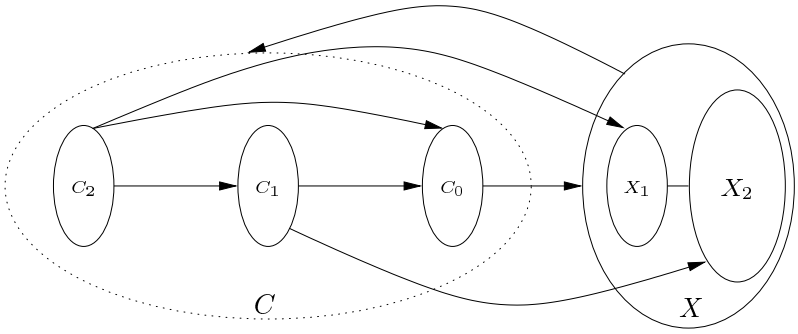

Fig. 2. The graph D_p with kelly-sw(D_p) $\neq$ mon-kelly-sw(D_p).

parts A and B means that there are edges from every vertex in A to every vertex in B, where again an undirected edge between A and B means that there are edges in D_p in both directions.

The following strategies show that mon-kelly-sw(D_p) $\leq 7p$ and kelly-sw(D_p) $\leq 6p$. For the monotone game we use the strategy $(X \cup C_0, X_2 \cup C_0 \cup C_1, X_1 \cup C_0 \cup C_1, X_1 \cup C)$, i.e. the searchers first occupy all of X and C_0, then proceed to $X_2 \cup C_0 \cup C_1$, and $X_1 \cup C_0 \cup C_1$ and finally move to $X_1 \cup C$. For the non-monotone case we use $(X \cup C_0, X_2 \cup C_0 \cup C_1, X_1 \cup C_1, X_1 \cup C_1 \cup C_2, X, X \cup C_0)$.

To see that kelly-sw(D_p) $\geq 6p$ note that $C_0 \cup X$ is a clique of size $6p$. It remains to show that mon-kelly-sw(D_p) $\geq 7p$. Suppose mon-kelly-sw(D_p) $< 7p$ and let $S = (v_1, \cdots, v_{|V(D_p)|})$ be a searcher-strategy witnessing this. For each part $Y \in \{C_0, C_1, C_2, X_1, X_2, C, X\}$ of D_p let $I(Y)$ be the greatest index of a vertex in Y, i.e. $v_{I(Y)}$ is the last vertex of Y which is searched by S. Then the following statements hold:

1. $I(X) < I(C_1)$ and $I(X) < I(C_2)$. For the sake of contradiction, suppose $I(X) > I(C_1)$ and let $v = v_{I(X)}$. Hence, when the searchers clear v, they have already cleared all vertices in X other than v and all vertices in C_1. As v has edges to every other vertex in $C_1 \cup X$, the searchers need to occupy all of $(C_1 \cup X) \setminus \{v\}$ before they can place a token on v. But this requires $7p$ searchers.
   The case of $I(X) < I(C_2)$ is analogous.
2. $I(C_0) < I(C_1)$. Again, assume the contrary, i.e. $I(C_0) > I(C_1)$. Hence, when clearing $v_{I(C_1)}$ there is a free vertex $v \in C_0$ through which the fugitive can reach all of X. As $I(X) < I(C_1)$, the searchers needs to occupy at least $(X \cup C_1) \setminus \{v_{I(C_1)}\}$ before clearing $v_{I(C_1)}$, which yields the contradiction.
3. $I(C_1) < I(C_2)$. With a similar reasoning as before we obtain that otherwise the searchers have to occupy $X \cup C_2$ when searching $v_{I(C_2)}$, using $7p$ searchers.

The statements (1)-(3) imply $I(X) < I(C_0) < I(C_1) < I(C_2)$ but now the searcher needs to occupy $|C_2 \cup C_1 \cup C_0 \cup X_1| = 7p$ vertices in order to search $v_{I(C_2)}$. So S uses at least $7p$ searchers. □

4 Limits of Algorithmic Applications

In [16] it has been shown that the k-disjoint path problem as well as related problems, including the Hamiltonian-path problem, are solvable in polynomial time on graphs of bounded directed tree-width. However, up to now only few other problems are known to be solvable with the help of digraph decompositions, a further example being parity games, which are tractable on graphs of bounded DAG- and Kelly-width [4,15]. As directed tree-width is the most general of these width-measures, tractability results for directed tree-width directly extend to all other measures. The converse is not true, for example it is not known whether parity games are tractable on graphs of bounded directed tree-width.

In this section we explore the algorithmic boundaries of the digraph measures introduced so far. In our analysis we focus on NP-complete problems that are explicitly directed. All analysed problems are solvable in polynomial time on digraphs whose underlying undirected graph has bounded tree-width – but as mentioned in the introduction, tree-width is not a good measure for the global connectivity of a digraph. Furthermore, we discard problems that are not tractable on acyclic graphs, as all measures defined so far are bounded on acyclic graphs. As representatives for various types of the remaining problems, we have considered the following problems: Minimum Equivalent Subgraph, Directed Feedback Vertex / Arc Set, Graph Grundy Numbering, and Kernel.

It turns out that all of these problems remain NP-complete even on digraphs that have very low global connectivity, i.e. digraphs that can be decomposed into components of constant size just by removing a small number of vertices. In particular, these graphs have low width with respect to all digraph decompositions defined so far, i.e. small directed path width, small DAG-, Kelly-, and directed tree-width, small Entanglement and D-width. In order to state the proofs in their most general way we define the class $\mathcal{CONN}_i^j$ as follows:

Definition 41 *Let i and j be integers. We define $\mathcal{CONN}_i^j$ to be the class of digraphs, such that for every digraph $D \in \mathcal{CONN}_i^j$ there exists a vertex set $X \subseteq V(D)$ with $|X| \leq j$, such that every component in $D \setminus X$ has at most i vertices.*

As mentioned above it is easy to see that:

Proposition 42 *For all i and j the class $\mathcal{CONN}_i^j$ has bounded directed path-width, directed tree-width, D-width, DAG-width, Kelly-width and Entanglement* [1].

4.1 Minimum Equivalent Subgraph

The *Minimum Equivalent Subgraph (MES)*-problem is the problem to compute in a given digraph D an edge-minimal subgraph $D' \subseteq D$ that preserves reachability in D.

Definition 43 *Let D be a digraph and $k \in \mathbb{N}$.* MES *is the problem to decide, if there exists a set $E' \subseteq E(D)$ with $|E'| \leq k$, such that the digraph $D' = (V(D), E')$ contains a path between two vertices if, and only if, such a path exists in D, i.e. D and D' have the same transitive closure.*

[1] An upper bound for all given width parameters is $i + j$.

MES is NP-complete for arbitrary digraphs (see [14]), but is known to be solvable in polynomial time for acyclic and undirected graphs. In [21] it is also shown that it suffices to consider MES on connected digraphs. There MES is equivalent to a generalisation of the directed hamiltonian cycle problem, the so-called round-trip-problem, in which vertices can be used more than once. This is particularly interesting because the directed hamiltonian cycle problem is a special case of the k-linkage problem, which can be solved in polynomial time on digraphs of bounded directed tree-width.

Definition 44 *Let D be a connected digraph. A round-trip $R = (v_1, \cdots, v_k, v_1)$ is a sequence of $k+1$ vertices of D, such that $(v_i, v_{i+1}) \in E(D)$ and R visits every vertex of D at least once. The size of R equals the number of distinct edges used by R.*

Lemma 45 *[21] Let D be a connected digraph and k a natural number. Then D has a MES of size less than k if, and only if, D has a round-trip of size less than k.*

The NP-completeness of MES for digraphs in $\mathcal{CONN}_3^1$ follows from a reduction of 3-SAT to the problem of finding a minimum round-trip in a connected digraph. Due to space restrictions the proof is deferred to the appendix.

Theorem 46 *The MES-problem is NP-complete even when restricted to digraphs in $\mathcal{CONN}_3^1$.*

4.2 Feedback Vertex Set / Feedback Arc Set

The *Feedback Vertex/Arc Set (FVS/FAS)*-problem is the problem to find a minimum set of vertices (edges) in a digraph D, whose removal leaves D acyclic. Both problems are known to be NP-complete on arbitrary digraphs (see [17]). Trivially both problems become efficiently solvable on acyclic graphs.

We prove the NP-completeness of FVS/FAS on digraphs in $\mathcal{CONN}_4^1$ respectively $\mathcal{CONN}_8^2$ by reducing to it a special variant of 3-SAT namely 3-SAT-2, which we introduce now.

Definition 47 *3-SAT-2 is the variant of 3-SAT, so that every literal is used in at most two clauses.*

3-SAT-2 is NP-complete. As before the proofs can be found in the appendix.

Theorem 48 *FVS respectively FAS are NP-complete even when restricted to digraphs in $\mathcal{CONN}_4^1$ respectively $\mathcal{CONN}_8^2$.*

4.3 Graph Grundy Numbering and Kernel

Definition 49 Graph Grundy Numbering *is the problem to decide for a digraph D, if there exists a function $f : V(D) \to \mathbb{N}$, such that for all $v \in V(D)$, $f(v)$ is the smallest natural number not contained in $\{f(u) : u \in V(D), (v,u) \in E(D)\}$.*

Definition 410 Kernel *is the problem to decide in a digraph D, if there exists $V' \subseteq V(D)$, such that*

1. *there is no edge between two vertices in V', i.e. V' is an independent set.*
2. *for every $v \in V(D) \setminus V'$ there exists a $u \in V'$ with $(v, u) \in E(D)$.*

Observe, that on undirected graphs the maximisation version of Kernel is the *Independent Set*-problem, whereas the minimisation version of Graph Grundy Numbering equals Vertex-Colouring. In contrast to the undirected case, where every graph has an Independent Set and a Vertex Colouring, not every digraph has a Kernel or a Graph Grundy Numbering and it is already NP-complete to decide whether a Kernel or a Graph Grundy Numbering do exist [26]. A simple example of a digraph that neither has a Graph Grundy Numbering nor a Kernel is the directed cycle with three vertices. Nevertheless it is easy to see that Kernel and Graph Grundy Numbering are trivially solvable on acyclic graphs. We are now ready to prove the NP-completeness for Graph Grundy Numbering on digraphs in $\mathcal{CONN}_4^0$ - again the proofs are deferred to the appendix.

Theorem 411 *Graph Grundy Numbering and Kernel are NP-complete even when restricted to digraphs in $\mathcal{CONN}_4^0$.*

5 Conclusion and Open Problems

In this paper we considered graph searching games on directed graphs and established non-monotonicity for two important variants of these games. Our examples show that the monotonicity costs for these games cannot be bounded by an additive term, i.e. for any k there are digraphs where at least k additional searchers are required to catch a robber with a monotone strategy. However, so far there is no upper bound for the monotonicity costs involved. It is conceivable that there is a constant $c \in \mathbb{N}$ such that whenever n searchers suffice to catch a robber on a digraph D in any of the two variants, than $c \cdot n$ searchers suffice for a monotone strategy. This, however, is left as an open problem.

A different trait we explored in this paper are the limits of an algorithmic theory based on directed graph decompositions. We showed that while there are interesting and important examples for natural problems that become tractable on digraphs of small width, many other natural problems remain NP-complete even if the digraphs have very low global connectivity.

References

1. I. Adler. Directed tree-width examples. *Journal Combinarial Theory Series B*, 97(5):718–725, 2007.
2. B. Alspach. Searching and sweeping graphs: A brief survey. In *COMBINATORICS 04*, 2004.
3. J. Barát. Directed path-width and monotonicity in digraph searching. *Graphs and Combinatorics*, 22(2):161–172, 2006.
4. D. Berwanger, A. Dawar, P. Hunter, and S. Kreutzer. DAG-width and parity games. In *STACS 2006*, volume 3884 of *Lecture Notes in Computer Science*, pages 524–536. Springer, Berlin, 2006.

5. D. Berwanger and E. Grädel. Entanglement – A measure for the complexity of directed graphs with applications to logic and games. In *LPAR*, pages 209–223, 2004.
6. D. Bienstock and P. Seymour. Monotonicity in graph searching. *Journal of Algorithms*, 12:239 – 245, 1991.
7. H. L. Bodlaender. Treewidth: Algorithmic techniques and results. In *MFCS'97*, volume 1295 of *Lecture Notes in Computer Science*, pages 19–36, 1997.
8. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998.
9. N. D. Dendris, L. M. Kirousis, and D. M. Thilikos. Fugitive-search games on graphs and related parameters. *Theorectical Computer Science*, 172(1-2):233–254, 1997.
10. R. Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, third edition, 2005.
11. P. D.Seymour and R. Thomas. Graph searching, and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Series B*, 58:22–33, 1993.
12. D. Dyer. *Sweeping Graphs and Digraphs*. PhD thesis, Simon Fraser University, 2004.
13. F. V. Fomin and D. M. Thilikos. On the monotonicity of games generated by symmetric submodular functions. *Lecture Notes in Computer Science*, 2204:177+, 2001.
14. M. R. Garey and D. S. Johnson. *Computers and Intractibility*. W. H. Freeman and Company, 1979.
15. P. Hunter and S. Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. In *Proceedings of the 18th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 637 – 644, 2007.
16. T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas. Directed tree-width. *J. Combin. Theory Ser. B*, 82(1):138–154, 2001.
17. R. M. Karp. *Complexity of Computer Science*. Plenum Press New York, 1972.
18. A. S. LaPaugh. Recontamination does not help to search a graph. *Journal of the ACM*, 40:224 – 254, 1993.
19. F. Mazoit and N. Nisse. Monotonicity property of non-deterministic graph searching. In *In Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2007)*, 2007.
20. F. Mazoit and N. Nisse. Monotonicity of non-deterministic graph searching. *Theor. Comput. Sci.*, 399(3):169–178, 2008.
21. D. M. Moyles and G. L. Thompson. An algorithm for finding a minimum equivalent graph of a digraph. *Journal of the ACM*, 16(3):455–460, 1969.
22. J. Obdržálek. DAG-width: connectivity measure for directed graphs. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 814–821, 2006.
23. B. Reed. Introducing directed tree width. In *6th Twente Workshop on Graphs and Combinatorial Optimization (Enschede, 1999)*, volume 3 of *Electronic Notes in Discrete Mathematics*, page 8 pp. (electronic). Elsevier, Amsterdam, 1999.
24. D. Richerby and D. Thilikos. Searching for a Visible, Lazy Fugitive. In *34th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2008)*, 2008.
25. M. A. Safari. D-width: a more natural measure for directed tree width. In *Mathematical foundations of computer science 2005*, volume 3618 of *Lecture Notes in Computer Science*, pages 745–756. Springer, Berlin, 2005.
26. J. van Leeuwen. Having a grundy-numbering is NP-complete. Technical report, Pennsylvania State University, 1976.
27. B. Yang and Y. Cao. Digraph strong searching: Monotonicity and complexity. In *AAIM*, pages 37–46, 2007.