Complexity Measures for Directed Graphs

Roman Rabinovich
Matrikelnummer 248505

Diplomarbeit
vorgelegt der Fakultät
für Mathematik, Informatik und Naturwissenschaften
der Rheinisch-Westfälischen Technischen Hochschule Aachen

im August 2008 (letzte Version: 31.07.2009)

angefertigt am
Lehr- und Forschungsgebiet
Mathematische Grundlagen der Informatik
Prof. Dr. Erich Grädel

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, den 08. August 2008

(Roman Rabinovich)

Contents

List of Figures

Introduction

Graphs are a powerful instrument for modelling in computer science. Many problems of practical and theoretical nature are expressed in terms of graphs. It is a computer scientist's task to develop algorithms that solve these problems. Unfortunately, there are some problems, also on graphs, that often appear in the everyday work, for which we do not know efficient algorithms. We have rather convincing arguments that solutions that would be efficient for all inputs cannot be constructed. Generations of scientists and programmers have tried to find them, but despite substantial effort no such solution was found. These difficult problems (such as NP-hard or PSPACE-hard) can, however be partially solved in practice. If there is – probably – no way to solve the whole problem then, may be, we are able to create an algorithm that would surely solve a problem on a certain subclass of "simple" graphs. We hope or can prove, that the graphs we have to deal in practice are "simple".

To guarantee this property, we need a measure for complexity of graphs. Many difficult problems become easy if considered on undirected trees, connected graphs without cycles. The idea to measure, how similar to a tree a given graph is, led, on undirected graphs, to the introduction of treewidth [49]. It can be proved for multiple problems that fast algorithms exist, if we restrict the input to graphs with bounded treewidth [15]. It is worth mentioning that there exist problems whose solving eludes such approach [39]. However, on problems where the idea works, a related notion of tree decomposition allows *constructive* proofs of existence of efficient algorithms. The construction for treewidth and most other measures relies on a *decomposition* of the given graph into parts connected in a tree-like manner. Exact decomposition type defines the measure as follows. The tree structure restricts connections in the graph in the sense that each such connection (any path in the graph) between elements of two parts A and B goes only through elements of parts C that lie on the way in the tree. This *separation* property builds the base for efficiency of algorithms. About the structure within the parts nothing is said, they can be arbitrary complex. The only trivial bound is the size of the largest part. We, therefore, try to find such a decomposition that the largest part is possibly small. This minimal size defines the measure we want to define.

Besides its practical use in constructing algorithms, treewidth proved that it has nice theoretical aspects. A large number of various characterisations of treewidth is an evidence that the notion is essential. Treewidth also plays an important role in the theory of minors by Robertson and Seymour [47].

We shall focus on the notion of *cyclicity* and consider graph complexity measures for directed graphs. Several attempts were made in the last two decades to generalise treewidth. The first one, directed treewidth, was by Johnson et al. in [35]. The measure proposed by the authors, even though it covers a whole class of directed graphs that are "simple" inputs for algorithms has some theoretical disadvantages, as discussed in Section 1.5. Other attempts were made (see Chapter 1), but a measure with as good properties as treewidth has, were not found yet. Reed argues in [44] that this situation roots at directed graphs themselves, which have features making a development of a appropriate measure difficult.

An intuitive concept for studying graph complexity measures are searching games. There are two players, the Robber player controlling a robber who runs across the graph, and the Cop player operating a certain number of cops. In general, the moves of the robber are bounded by the graph structure. Cops often (depending on the particular game) are free in there moves. When the robber and a cop find themselves on the same place, the robber is captured. Precise rules vary with the measure under consideration. The minimal number of cops needed to capture the escaping robber in a particular searching game corresponds to a measure.

A measure we pay most attention is *entanglement*. It was introduced by Berwanger and Grädel in [10]. Entanglement was used in [12] to prove the strictness of the variable hierarchy of the μ-calculus, a logic of "prominent position" [12] in computer science. We, however, study entanglement analogously to other measures, as an instrument for construction efficient algorithms for problems on graphs where the entanglement is bounded. Here, several aspects are important.

The first one is that, unlike other measures, entanglement is defined in terms of a game rather than via a decomposition. This is a reason why it is difficult to determine, for a class of graphs, whether its members are of bounded entanglement. On the other hand, a decomposition characterisation would lead to a non-deterministic algorithm to compute the entanglement on a given graph. Until now we know only a rather trivial algorithm that is based on a simulation of all possible plays.

Outline

In Chapter 1 we introduce measures we want to study. First we describe two measures, thetreewidth and the pathwidth on undirected graph as a source of motivation for the directed measures and then switch over to directed ones. We describe possible generalisations of treewidth and the pathwidth.

Chapter 2 is devoted to some important common properties of measures. We discuss complexities of computing measures and which problems become easy if the measure is bounded.

Chapter 3 deal with the entanglement. First, we give a characterisation of the measure and describe graph classes of known or estimated entanglement. In the second part of the chapter we prove a structural characterisation of the class of graph having entanglement two.

In Chapter 4 we compare different measures with each other and give some bounds of one measure in terms of another.

Acknowledgements

I would like to thank Prof. Grädel for giving me a topic for my diploma thesis and for his advice during the time I worked on it.

I am also very grateful to Prof. Thomas for his willingness to co-examine my thesis.

I thank also all assistants of the research group Computer Science 7 for a pleasant atmosphere that favoured my work a lot. Special thanks go to Dietmar Berwanger for inspiring conversations, his ability to make me self-confident in writing the thesis and for checking it for errors and bugs. A special thank goes to Łukasz Kaiser for fruitful discussions, a thorough reading of draft versions of the thesis and a final argument in the main result.

I would like to thank my student colleagues Bernd Puchala, Jörg Olschewski and Nils Jansen who have never been tired to listen to my explanations, at which stage of my work I was being.

A special thank goes to Walid Belkhir for pointing out an error in a result that is excluded in the current version of the thesis.

At last, but not least, I thank my parents and my brother Boris for their love and for their patience with me while I was studying.

Chapter 1

Measures of Graph Complexity

In this section we introduce several measures of graph complexity. There are two aspects we consider: the structural and the computational (algorithmic) complexity.

Structural Aspects. One property of a measure is the existence of a relatively simple structure , for every graph (such as an undirected tree or a DAG) that describes, into which simpler parts the graph can be decomposed and what the connection between the parts is. Hereby, elements of the structure are assigned to subsets of graph vertices. The graph is simple with respect to a measure if and only if the largest such subset is small. For many of measures that we will discuss such *decompositions* are known, as, e.g., tree decompositions for the treewidth and the DAG-decompositions for the DAG-width. For entanglement, a measure introduced in [10], no such decomposition is known. We give a decomposition of a graphs for a special case in Chapter 3. It is an advantage if a decomposition is small, i.e., its size is polynomially bounded in the size of the graph. Such decompositions can serve as a polynomial witness in nondeterministic algorithms. Their construction is often base on dynamic programming and use distinguished parts of a graph as a ground level for their computation, see Section 2.4. We shall discuss sizes of decompositions in Chapter 2.

Another way to characterise the complexity of a graph is to describe it by an algebraic expression that is built of vertices and/or edges connected by some operations on graphs such as disjoint union or adding edges. In this way, treewidth and Kelly-width can be described. Algebraic expressions also characterise a class of graphs with bounded entanglement, see Section 3.

Algorithmic Aspects. A illustrative and intuitive way to work with graph complexity measures is to express the measure in terms of a game played according to certain rules by a fugitive who runs between vertices of the graph and searchers who try to locate the fugitive. A function, which maps the number of searchers needed to capture the fugitive on a graph is the measure of its complexity. Different rules define hereby different measures. In this way, entanglement is defined and there are game-theoretic characterisations of most of the discussed measures.

To compute the complexity of a graph via the corresponding game, one should determine which of the players has a winning strategy. Properties of strategies will be discussed in Section 2.

First, we introduce some basic definitions and notations used in this thesis.

1.1 Preliminaries

Throughout the thesis we shall use the following definitions and notations. The set of natural numbers is denoted by ω, the set of real numbers by $\mathbb{R}$. An *undirected graph* is a structure $\mathcal{G} = (V, E)$ with a *vertex* set V and a set of edges $E \subseteq \{\{u, v\} \mid u, v \in V\}$. A *directed graph* is a structure $\mathcal{G} = (V, E)$ with a *vertex* set V and a set of edges $E \subseteq V^2$. Undirected graphs can be viewed as directed ones with a symmetrical edge relation E, but we shall formally distingish between the classes. For directed graphs, if v is a vertex we denote by vE, or $N^{out}(v, \mathcal{G})$, or simply $N^{out}(v)$ the set $\{w \in V \mid (v, w) \in E\}$. By $N^{in}(v, \mathcal{G})$, or simply $N^{in}(v)$ we denote the set $\{w \in V \mid (w, v) \in E\}$. If $\mathcal{G}$ is undirected vE denotes the set $\{w \in V \mid \{v, w\} \in E\}$ of *neighbours* of v. The *degree* of v is $d(v) = |vE|$, if the graph is undirected. For directed graphs, we write $d^{in}(v)$ for $\{w \in V \mid (w, v) \in E\}$. Analogously, $d^{out}(v)$ for $\{w \in V \mid (v, w) \in E\}$.

A graph $\mathcal{G}' = (V', E')$ is a *subgraph* of $\mathcal{G}$, if $V' \subseteq V$ and $E' \subseteq E$. It is an *induced* subgraph, if $V' \subseteq V$ and $E' = \{\{v, w\} \in E \mid v \in V'$ and $w \in V'\}$ for undirected graphs, or $E' = \{(v, w) \in E \mid v \in V'$ and $w \in V'\}$, for directed graphs. In these cases we write $\mathcal{G}[V']$ for $\mathcal{G}'$. If X is a set of vertices $X \subseteq V$ then we write $\mathcal{G}\ X$ for $\mathcal{G}[V \backslash X]$. If $X = \{v\}$ is a singleton we write $\mathcal{G} \backslash v$ instead of $\mathcal{G} \backslash X$. An *undirected path* $\mathcal{P} = (P, E_P)$ of length $n \leq 0$ from v_0 to v_{n-1} is an undirected graph with $P = \{v_0, \ldots, v_{n-1}\}$ (note that all v_i are distinct) and $E_P = \{\{v_i, v_{i+1}\} \mid 0 \leq i \leq n-2\}$. For a *directed path* we replace $\{v_i, v_{i+1}\}$ by (v_i, v_{i+1}) in the definition of E_P. For vertices v and w, we write $v \leq w$ when there is a directed path in $\mathcal{G}$ from v to w. Then v is the predecessor and w the ancestor. This order is a *topological* order on $\mathcal{G}$. If v and w are distinct, we also write $v < w$. If e is an edge, we write $v < e$, if there is a vertex w with $e = (v, w)$. In undirected graphs, we say that e is *incident* with v and write $e \sim v$ if $v \in e$. A *directed cycle* of length $n \leq 0$ is a graph $\mathcal{C} = (C, E_C)$ with $C = \{v_0, \ldots, v_{n-1}\}$ and $E_C = \{(v_i, v_{i+1}) \mid 0 \leq i \leq n-2\} \cup \{(v_{n-1}, v_0)\}$. The *undirected cycle* is defined accordingly with $n \leq 3$. A *forest* is an undirected acyclic graph. An undirected graph is *connected*, if there is a path from every vertex to every vertex. A directed graph is *weakly connected*, if, for every pair of vertices v, w, we have $v < w$ or $w < v$. It is *strongly connected*, if there is a non-trivial (i.e., with at least an edge) path from every vertex to every vertex. A *directed tree*, or an *arborescence* is a (weakly connected) directed graph with exactly one path from every vertex to every vertex (in particular, there are no cycles: consider also paths of length 0). A *strongly connected component* of $\mathcal{G}$ is a maximal strongly connected induced subgraph, where $\mathcal{G}$ is directed. For undirected graphs, a *connected component* is a maximal connected subgraph. A undirected connected graph is *biconnected*, if a deletion of any vertex does not make it unconnected. A vertex w is called a *child*, or a *direct successor* of a vertex v in directed tree, if $(v, w) \in E_T$; v is then a *parent*, or a *directed predecessor* of w. If two vertices are children of the same vertex they are *siblings*. A DAG is a directed acyclic graph. The set of vertices *reachable* from a set of vertices X in $\mathcal{G}$ is

$$Reach_{\mathcal{G}}(X) = \{v \in V \mid \text{ there is a vertex } w \in X \text{ and a path in } \mathcal{G} \text{ from } w \text{ to } v\}.$$

A *(rooted) full binary tree* of length k is a directed tree (with a distinguished *root*, i.e., the vertex from that there is a path to every other vertex) such that every vertex except the root has exactly two children and all maximal paths from the root have the same length $k+1$ (and k edges in these paths).

The directed graph with *reversed edges* (with respect to $\mathcal{G}$) is the graph $\mathcal{G}^{op} = (V, E')$ with $E' = \{(w, v) \mid (v, w) \in E\}$. A *complete graph* or a *clique* on $n \geq 1$ vertices is $\mathcal{K}_n = (V_K, E_K)$ with $|V_K| = n$, and $E_K = V^2 \backslash \{(v, v) \mid v \in V\}$ in the directed case, or $E_K = \{\{u, v\} \mid u, v \in V_K, u \neq v\}$ in the undirected case.

Let $V_1, V_2 \subseteq V$ be disjoint non-empty sets of vertices. We say that V_1 *guards* V_2 if, for all edges $(v, w) \in E$, if $v \in V_2$ and $w \notin V_2$ then $w \in V_1$.

For $n \in \omega$, we denote by $[n]$ the set $\{0, 1, \dots, n-1\}$. For a set S, $[S]^{\leq k}$ ($[S]^{<k}$) is the set of all subsets of S of size at most k (less than k). The power set of a set S is denoted by 2^S.

Given directed graphs $\mathcal{G}_1 = (V_1, E_1)$ and $\mathcal{G}_2 = (V_2, E_2)$, the *directed union* of $\mathcal{G}_1$ and $\mathcal{G}_2$ is the directed graph $\mathcal{G} = (V, E)$ with $V = V_1 \cup V_2$ and $E \subseteq E_1 \cup E_2 \cup (V_1 \times V_2)$.

1.2 A framework to describe games

Hunter gives in [33] a method to describe search games in a precise and consistent manner. We shall adapt this framework to define games we want to discuss. In explanations and proofs (especially when speaking about strategies) we use more intuitive informal descriptions. We also use the names of games given in [33].

Definition 1.2.1. An *arena* is a tuple $\mathcal{A} = (V, V_0, V_1, E, v_I)$ where

- (V, E) is a directed graph,
- V_0, the set *of Player 0 positions*, and V_1, the set *of Player 1 positions*, form a partition of V, and
- $v_I \in V$ is the *initial position.*

Definition 1.2.2. A (two-player perfect information) *game* is a tuple $(\mathcal{A}, \mathrm{Win})$ where $\mathcal{A}$ is an arena and $\mathrm{Win} \subseteq v_I \cdot V^\omega$ is a *winning condition* for Player 0. A sequence of vertices $v_I, v_1, v_2, \dots$ with $(v_I, v_1) \in E$ and $(v_i, v_{i+1}) \in E$, for all $i \geq 1$, is *play* in $\mathcal{A}$. Player 0 *wins an infinite play* π if π is infinite and in Win or if it is finite, $\pi = v_I, v_1, \dots, v_m$, and $v_m E$ is empty and $v_m \in V_1$. Otherwise Player 1 wins.

A game is played by Player 0 and Player 1 on the game graph (V, E). The players move a pebble from a vertex to a vertex along (directed) edges. At the beginning of a play, the pebble is on the vertex v_I. If the vertex where the pebble currently is, is in V_0, Player 0 moves it, otherwise, it is Player 1's turn. If a Player has to move, but there is no successor of the current vertex, he loses. An infinite play is won by Player 0 if and only if it is in the winning condition Win.

In a game, we are more interested in the question who can win the game than who wins a particular play. The players should stick to a plan to play well. The next definition makes the sense of notion of precise.

Definition 1.2.3. Let $(\mathcal{A}, \mathrm{Win})$ be a game with $\mathcal{A} = (V, V_0, V_1, E, v_I)$ and let i be in $\{0, 1\}$. A partial function $\sigma : v_I V_i^* \to V$ is a *strategy* of Player i for the game $(\mathcal{A}, \mathrm{Win})$. A play $v_0, v_1 \ldots$ in $(\mathcal{A}, \mathrm{Win})$ is *consistent* with a strategy σ of Player i if, for all $j \geq 0$, if v_j is in V_i then $\sigma(v_j) = v_{j+1}$. A strategy of Player i is *winning* if Player i wins every play that is consistent with it.

In general, a Player knows the whole history of the play until the current position, i.e., he remembers all game positions that appeared in the play. His strategy tells him, which move he has to make, based on the knowledge of the history. Winning strategies (if they exist) guarantee him a win. In the games that we shall meet in this thesis, all players will have *memoryless* strategies, i.e., to determine what to do in the next move, they will not need to remember all positions they have seen in the play, but rather just to know the current position.

Definition 1.2.4. Let $(\mathcal{A}, \mathrm{Win})$ be a game with $\mathcal{A} = (V, V_0, V_1, E, v_I)$ and let i be in $\{0, 1\}$. A *memoryless strategy* of Player i is a partial function $\sigma : V_i \to V$.

The definitions of a consistent play and of a winning memoryless strategy are analogous to those given in Definition 1.2.3.

Intuitively, the game on an graph $\mathcal{G}$ is the following. Consider the cops and the robber as persons who can stay on vertices of the graph and move from one vertex to another. At the beginning, all cops are outside the graph and the robber chooses an arbitrary vertex in a connected component R_0 where she[1] places herself. The first game position is $(\emptyset, R_0)$. A game position (X, R) means that the robber is on a vertex r in a (weakly) connected component R of $\mathcal{G}$ and the cops occupy all vertices in a subset X of V. The cops move to a position (X, X', R), i.e., they announce a subset X' where they will

[1]For convenience, we shall assume that the cops are male and the robber is female.

be in the next step, and those of them who are not in $X \cap X'$ (i.e., who are going to move) get in a helicopter and start moving towards $X' \backslash X$. Now the robber moves to a position (X', R'), which means that she moves at a great speed to a (not necessarily direct) successor $r' \in R'$ of r before the cops land on X'. On this way she is, though, not permitted to pass through vertices where the idle cops from $X \cap X'$ are. Now the flying cops land on the announced vertices of $X' \backslash X$.

Definition 1.2.5. A *graph searching game type* is a function Γ which maps a graph $\mathcal{G} = (V, E)$ to a triple $(\mathcal{L}_s, \mathcal{L}_r, \mathcal{A})$ where $\mathcal{L}_c$ and $\mathcal{L}_r$ are sets of subsets of V and $\mathcal{A} = (V_A, V_0, V_1, E_A, v_I)$ is an arena satisfying

- $\emptyset \in \mathcal{L}_c$,
- $\emptyset \notin \mathcal{L}_r$, and $\mathcal{L}_r$ has a unique $\subseteq$-maximal element R_{max},
- $V_0 \subseteq \mathcal{L}_c \times \mathcal{L}_r$ consists of pairs (X, R) where $X \cap R = \emptyset$,
- $V_1 \subseteq \mathcal{L}_c \times \mathcal{L}_c \times \mathcal{L}_r$ consists of triples of the form (X, X', R) where $X \cap R = \emptyset$,
- $v_I = (\emptyset, R_{max})$,
- If $\big((X, R), (X', X'', R')\big) \in E_A$ then $X = X'$ and $R = R'$,
- If $\big((X, X', R), (X'', R')\big) \in E_A$ then $X' = X''$ and, for all $r' \in R'$, there is $r \in R$ such that r and r' are in the same (weakly) connected component of $\mathcal{G} \backslash (X \cap X')$, and
- If $S \subseteq R$ then, for all S' such that $\big((X, X', S), (X', S')\big) \in E_A$, there exists $R' \supseteq S'$ such that $\big((X, X', R), (X', R')\big) \in E_A$.

Given a graph searching type Γ, and a graph $\mathcal{G}$ with $\Gamma(\mathcal{G}) = (\mathcal{L}_c, \mathcal{L}_r, \mathcal{A})$, the *graph searching game* on $\mathcal{G}$ (defined by $\Gamma(\mathcal{G})$) is the game $\mathcal{A}, \emptyset$, so Player 1 wins all infinite plays. We call Player 0 the *Cop player* and Player 1 the *Robber player*.

To avoid the trivial strategy of the cops that consists in placing a cop on every graph vertex we make an additional restriction.

Definition 1.2.6. Let k be a natural number. A graph searching Game *with k cops* is a graph searching Game where, for every $X \in \mathcal{L}_c$, we have $|X| \leq k$.

As a next step we prove that, without loss of generality, we can restrict ourselves to considering memoryless strategies for graph searching Games with k cops. For this, we need the notion of parity games. We give only an informal description. A parity game is played by two players on a weighted graph $\mathcal{G}$ with a function c that colours all vertices of the arena with natural numbers taken from a finite subset of ω. The players (Player 0 and Player 1) push a token from a fixed initial position in turn until a player is unable to move (he loses then) or infinitely long. In the latter case Player 0 wins if and only if the minimal infinitely often seen colour is even. The players see a colour m if the token is on a vertex v with $c(v) = m$.

Observe that seaching games are safety games for the Robber player and reachability games for the Cop player, so we so get the next lemma.

Lemma 1.2.7. *Every graph searching game is positionally determined, i.e., one of the Players has a memoryless winning strategy.*

1.3 Treewidth

Treewidth was introduced in [49] , a historical overview can be found in [17]. Although we are going to concentrate on directed graphs, we describe first the treewidth and its variant pathwidth – a measure for undirected graphs that has proven its importance in constructing algorithms and in understanding the structure of the graph. Its properties serve as a model for complexity measures of directed graphs. We shall discuss them in this chapter.

Today we have a variety of characterisations of the treewidth we discuss in this thesis. Although it was not chronologically the first definition (see [32], [35]), we start with a description of treewidth in terms of a searching game first given in [53] following [33] – an approach that we use also for other measures for that we know game-theoretic definitions.

Definition 1.3.1. Let $\mathcal{G} = (V, E)$ be an undirected graph. The *cops and visible robber* *game*, or the Treewidth Game $\mathrm{TwG}(\mathcal{G})$, on $\mathcal{G}$ is a graph searching game on $\mathcal{G}$ defined by the triple $(\mathcal{L}_c, \mathcal{L}_r, \mathcal{A})$ with $\mathcal{A} = (V_A, V_0, V_1, E_A, v_I)$ where

- $\mathcal{L}_c = 2^V$, $\mathcal{L}_r = \{R \subseteq V \mid R \text{ is not empty and connected}\} \cup \{V\}$,
- $(X, R) \in V_0$ if R is a connected component of $\mathcal{G}\backslash X$,
- $(X, X', R) \in V_1$ if $(X, R) \in V_0$ and $X' \in \mathcal{L}_c$,
- $\big((X, R), (X, X', R)\big) \in E_A$, for all $(X, R) \in V_0$ and all $X' \in \mathcal{L}_c$,
- $\big((X, X', R), (X', R')\big) \in E_A$ if $R \cup R'$ is contained in a connected component of $\mathcal{G}\backslash(X \cap X')$.

We call the Searchers the Cop and the Fugitive the Robber players in this game. The Cop player controls an unlimited number of cops (we can assume that he has $|V|$ cops) and the Robber player controls a robber. At the beginning of a play, all cops are outside the graph and the robber chooses a vertex in a connected component to place herself. In a game position (X, R), the robber is on a vertex $r \in R$ and the cops occupy all vertices $v_1, \ldots, v_k$ of X that are all different from r. The Cop player makes a move as in the general graph searching Game, i.e., he announces a set X' where some of the cops from X and, possibly, some from outside are going to move to. The new position is (X, X', R). The robber does nothing or runs at great speed to a vertex r' reachable from r whereby she is not allowed to visit vertices that are occupied by cops who remained on their places. The cops land on the announced vertices. If R' is the component where the robber is, which is limited by the cops, the new position is (X', R'). The play ends when the robber is captured, i.e., cannot move or moves to a vertex that is about to be occupied by a cop. Infinite plays are won by the Robber player.

It is clear that a trivial strategy of the Cop player is to place a cop on every vertex. To avoid this, the number of cops available for him is limited. We are primarily interested in the minimal number of cops needed to capture the robber. The key task is to find strategies that allow the Cop player to win with a minimal number of cops and the Robber player to escape from fewer cops.

Definition 1.3.2. Let $k > 0$ be a natural number. The *k cops and visible robber game* $\mathrm{TwG_k}(\mathcal{G})$ on an undirected graph $\mathcal{G}$ is the Cop and visible robber game on $\mathcal{G}$ with the additional condition that the Cop player has only k cops, i.e., for every $X \in \mathcal{L}_c$, we have $|X| \leq k$.

Definition 1.3.3. Let $\mathcal{G}$ be an undirected graph. If k is the minimal number of cops such that the Cop player wins the Treewidth Game $\mathrm{TwG_k}(\mathcal{G})$ on $\mathcal{G}$, then the *treewidth* of $\mathcal{G}$ is $k-1$.

It is clear that if the Cop player has a winning strategy in the Treewidth Game then he also has a positional winning strategy. It follows from the definition that if a graph has treewidth k, then all its subgraphs have treewidths at most k. Indeed, the restriction of a positional winning strategy to the Treewidth Game on a subgraph is a winning strategy.

A useful characterisation (which is often used as a definition) is one via decompositions of the graph. It not only leads to a understanding of the graph structure, but also helps to construct faster algorithms on the graphs with bounded treewidth for problems for which we do not know efficient algorithms otherwise. We shall discuss this issue in Section 2.4.

Definition 1.3.4. Let $\mathcal{G} = (V, G)$ be an undirected graph. A *tree decomposition* of $\mathcal{G}$ is a tuple $(\mathcal{T}, \mathcal{X}, f)$ where $\mathcal{T} = (T, E_T)$ is an undirected tree with vertices called *bags*, $\mathcal{X}$ is a subset of 2^V, and $f : T \to \mathcal{X}$ is a bijection such that

(1) $\bigcup \mathcal{X} = V$,

(2) for all graph edges $\{u, v\} \in E$, there exists a bag X with $\{u, v\} \subseteq X$,

(3) for all bags t_1, t_2, t_3, if there is a path from t_1 to t_3 in $\mathcal{T}$, then $f(t_1) \cap f(t_3) \subseteq f(t_2)$.

We call the images of the function f also *bags* and say that a graph vertex v is in a bag t if $f(t)$ contains v. The width of a tree decomposition is $\max_{t\in T} |f(t)| - 1$, i.e., the size of a maximal bag.

The third condition means that, for a graph vertex v, the set of all bags containing v induces a connected subtree of $\mathcal{T}$.

Note that even if a graph has more than one connected component, its tree decomposition is a (connected) tree. It suffices to construct tree decompositions $(\mathcal{T}_i)_{i\in I}$ for each component i from an index set $I \subset \omega$, and then to connect the resulting trees in a tree manner, i.e., we take $\mathcal{T}_0$ as the root and, for each $i \in I \backslash \{0\}$, connect an arbitrary bag of $\mathcal{T}_i$ with an arbitrary set of $\mathcal{T}_{i-1}$. Then the width of the decomposition obtained in this way is the maximum of the decompositions of the connected components.

Proposition 1.3.5. [53] *Let $\mathcal{G}$ be a graph. There is a tree decomposition of $\mathcal{G}$ of width at most k if and only if the treewidth of $\mathcal{G}$ is at most $k-1$.*

A particular type of tree decomposition is the *nice* tree decomposition with special properties that are mainly used in constructing algorithms [17], see also Section 2.4.

Definition 1.3.6. A *nice tree decomposition* is a tree decomposition where each bag t is of one of four types:

- *leaf:* t is a leaf of $\mathcal{T}$ and $|f(t)| = 1$,
- *join:* t has two children t_1 and t_2 with $f(t) = f(t_1) = f(t_2)$,
- *introduce:* t has one child t' such that there is a graph vertex v with $f(t) = f(t') \cup \{v\}$,
- *forget:* t has one child t' such that there is a graph vertex v with $f(t') = f(t) \cup \{v\}$.

Nice tree decomposition can be used as a normal form: for a tree decomposition, we can always find a equivalent nice tree decomposition of roughly the same size, in the following sense.

Proposition 1.3.7.

1. *A graph $\mathcal{G}$ has a tree decomposition of width at most k if and only if it has a nice tree decomposition of width at most k [38].*
2. *A nice tree decomposition can be computed from a given tree decomposition in linear time [17] .*

It follows that a nice tree decomposition has size linear in the size of a given tree decomposition.

We now consider some examples of graph classes with known treewidth. The next example shows that, informally, the treewidth measures how "tree-like" a graph is. The higher the treewidth of the graph, the more difficult it is to press it into a shape of a tree. Trees and forests are the simplest graphs in this sense.

Example 1.3.8. Undirected trees with at least two vertices have treewidth 1. Let r_0 be a vertex of the tree. For convenience, construct a directed tree (an arborescence) with r_0 as its root by directing all edges away from r_0. Let the resulting directed tree be $\mathcal{T} = (V, E)$. The winning strategy for two cops is to set one cop on r_0. The robber goes to a subtree of $\mathcal{T}$ with root r_1 where (r_0, r_1) is an edge of $\mathcal{T}$. Place the first cop on r_1. By induction, the robber is finally captured in a subtree with only two vertices and both cops on them.

It is clear that the robber escapes from one cop. More general, the treewidth of a clique K_n with n vertices is $n - 1$. Note that the robber can remain on her vertex when no cop occupies it. It follows that the treewidth is not bounded on the class of graphs, and moreover, for every $n > 0$, there is a graph with treewidth n.

If a graph $\mathcal{G}$ consists of several connected components $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_m$ then the treewidth of the graph is the maximum of the treewidths of the components: $\operatorname{tw}(\mathcal{G}) = \max\{\operatorname{tw}(\mathcal{C}_0), \operatorname{tw}(\mathcal{C}_1), \ldots, \operatorname{tw}(\mathcal{C}_m)\}$: in the Treewidth Game, the robber chooses a vertex in a component in the first move and the Cop player uses the number of cops needed for this component. It follows that a graph has treewidth one if and only if it is a forest.

If we are interested in the exact treewidth of a graph or of a graph class we need a method to establish a lower bound for the treewidth. In Example 1.3.11 we use a characterisation via *screens* or, in different terminology, *brambles.*

Definition 1.3.9. (see [53], [45], [18]) Let $\mathcal{G} = (V, E)$ be an undirected graph and $k \geq 1$ a natural number. Let X and Y be sets of vertices of $\mathcal{G}$. The sets X and Y *touch* if either $X \cap Y \neq \emptyset$ or some vertex in X has a neighbour in Y. A *screen* or a *bramble* $\mathcal{S}$ in $\mathcal{G}$ is a set of mutually touching connected subsets of V. A screen (a bramble) $\mathcal{S}$ has *thickness (order)* $\geq k$ if there is no $X \in [V]^{<k}$ such that $X \cap H \neq \emptyset$ for all $H \in \mathcal{S}$. The *bramble number* $\operatorname{bn}(\mathcal{G})$ of $\mathcal{G}$ is the maximum of the orders of its brambles.

A screen gives a straightforward strategy for the Robber player in the Treewidth Game on a graph. One can show even that the existence of a screen is necessary condition for the existence of a winning strategy of the Robber.

Lemma 1.3.10. *[53] Let $\mathcal{G}$ be an undirected graph and let $k > 0$ be a natural number. The treewidth of $\mathcal{G}$ is at least $k-1$ if and only if $\mathcal{G}$ has a screen of thickness at least k.*

Undirected grids play an important role in the theory of minors of Robertson and Seymour in connection to treewidth (see [49],[50], [27]). As we are actually interested in directed graphs we just consider grids as an example.

Example 1.3.11 (see [18] for a similar example)**.** An (undirected) $m \times n$-grid, for $m, n > 0$, is a graph $\mathcal{G} = (V, E)$ with $V = \{(i, j) \mid i \in [m], j \in [n]\}$ and $E = \{((i, j), (k, l)) \mid i = k-1$ and $j = l$, or $i = k$ and $j = l-1\}$. The treewidth of a $m \times n$-grid is $\min(m, n)$. To see that $\min(m, n) + 1$ cops have a winning strategy, assume that the graph has m rows and n columns and that $m \leq n$. At their first move, k cops occupy row 0, i.e. vertices of form $(0, j)$, for $j \in [m]$. Then the last cop goes to vertex $(1, 0)$, then those from $(0, j)$, $j \in [m-1]$ to $(1, j+1)$; then that from $(0, m-1)$ to $(1, 0)$ and so on. In general, the cops occupy a row $r < n$. The last cop goes to the vertex $(r+1, 0)$. Then, in turn, the cop from (r, j), for $j \in [m-1]$ to vertex $(r+1, j+1)$. Thus the cops search the whole graph and finally capture the robber. Note that the Cop player does not need to know where the robber goes to.

This strategy obviously does not work if the Cop player has at most m cops. We use the characterisation by screens to show that the robber has a winning strategy in this case. Let $\mathcal{S}$ be the set of all rows and all columns of the grid. The only sets of size at most m (for all $n \geq m$) that touch all these lines are $S_1 = \{(i, i) \mid i \in [m]\}$ and $S_2 = \{(i, m-i) \mid i \in [m]\}$. We add to $\mathcal{S}$ the sets $C_1 = \{(i, 0) \mid i \in [m-1]\}$ (which does not intersect S_1) and $C_2 = \{(m, i) \mid i \in \{1, \ldots, m-1\}\}$ (which does not intersect S_2). The set $\mathcal{S} \cup C_1 \cup C_2$ is a screen of thickness at least k.

One of the historical roots of the ideas that lead to treewidth are the Kirchhoff laws allowing to compute the resistance of an electrical network (see e.g., [17]). Two resistances can be combined either in a serial or in a parallel manner. To formalise these connections, we define a structure that extends a graph by a set of distinguished vertices.

Definition 1.3.12. A *graph with final vertices* is a structure $\mathcal{G} = (V, E, F)$ with vertices V, *final vertices* $F \subseteq V$, and edges $E \subseteq [V]^{\leq 2}$ if it is undirected or $E \subseteq V^2$ if it is directed.

A series-parallel graph is a graph that we can construct from graphs with final vertices of form $\mathcal{G} = (\{v, w\}, \{\{v, w\}\}, \{v, w\})$ that have two vertices (both are distinguished) and an edge between them, using the following operations. The resulting graphs will always have exactly two distinguished vertices.

- If $\mathcal{G}_0 = (V_0, E_0, \{v_0, w_0\})$ and $\mathcal{G}_1 = (V_1, E_1, \{v_1, w_1\})$ are series-parallel graphs with $V_0 \cap V_1 = \emptyset$, then so is $\mathcal{G} = (V, E, v_0, w_1)$, where $V = \big((V_0 \backslash \{w_0\}) \cup (V_1 \backslash \{v_1\})\big) \cup \{v\}$ and $E = \big((E_0 \backslash \{\{a, w_0\} \mid a \in V_0\}) \cup (E_1 \backslash \{\{v_1, a\} \mid a \in V_1\})\big) \cup \{\{a, v\} \mid \{a, w_0\} \in E_0\} \cup \{\{v, a\} \mid \{v_1, a\} \in E_1\}$. In other words, we build the disjoint union of $\mathcal{G}_0$ and $\mathcal{G}_1$ and identify w_0 and v_1, so that the resulting vertex is not distinguished.

- If $\mathcal{G}_0 = (V_0, E_0, \{v_0, w_0\})$ and $\mathcal{G}_1 = (V_1, E_1, \{v_1, w_1\})$ are series-parallel graphs with $V_0 \cap V_1 = \emptyset$, then so is $\mathcal{G} = (V, E, v_0, w_0)$, where $V = V_0 \cup (V_1 \backslash \{v_1, w_1\})$ and $E = E_0 \cup (E_1 \backslash \{\{v_1, a\}, \{b, w_1\} \mid a, b \in V_1\}) \cup \{\{v_0, a\} \mid \{v_1, a\} \in E_1\} \cup \{\{w_0, a\} \mid \{w_1, a\} \in E_1\}$. In other words, we build the disjoint union of $\mathcal{G}_0$ and $\mathcal{G}_1$ and identify v_0 and v_1, and w_0 and w_1.

To fulfil the construction we "forget" which vertices of the resulting graph are final, i.e., from a graph with final vertices (V, E, F) we get (V, E).

Proposition 1.3.13. *[20] A graph has treewidth two if and only if each of its biconnected components is series-parallel.*

Many classes of graphs with bounded treewidth are given in [21].

We give further characterisations for treewidth which were a motivation to explore similar properties in directed graphs. For an overview of treewidth characterisations see, for example, [16].

As trees can be constructed recursively, tree decompositions give a hint how a graph of bounded treewidth can be recursively constructed along its tree decomposition. First, we need some definitions.

Definition 1.3.14 (see [4])**.** A k-tree is an undirected graph constructed recursively as follows:

(i) A clique K_k is a a k-tree.

(ii) If $\mathcal{G} = (V, E)$ is a k-tree, $\mathcal{C} = (V_C, E_C)$ is a clique of size k in $\mathcal{G}$, and v is a new vertex then the graph $\mathcal{G}' = (V', E')$ where $V' = V \cup \{v\}$ and $E' = E \cup \{\{v, c\} \mid c \in V_C\}$, is a k-tree. In other words, we obtain a new k-tree by adding a new vertex to an existing k-tree and connecting it to a k-clique in the graph.

An undirected graph is a *partial k-tree* if it is a subgraph of a k-tree with the same vertex set.

Definition 1.3.15. An *elimination ordering* of an undirected graph $\mathcal{G} = (V, E)$ is a permutation on V. Let v be a vertex in V. An *elimination* of v from $\mathcal{G}$ is the graph $\mathcal{G}' = (V \backslash \{v\}, E')$ where $E' = (E \backslash \{\{u, v\} \mid u \in V\}) \cup \{\{u, w\} \mid \{u, v\}, \{w, v\} \in E\}$, i.e., $\mathcal{G}'$ is the graph obtained from $\mathcal{G}$ by deleting vertex v and connecting all vertices that are adjacent to v in $\mathcal{G}$ (if they were not connected). Every elimination ordering $\sigma = v_0, v_1, \ldots, v_{n-1}$ (where n is the size of V) induces a sequence of eliminations $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{n-1}$ where $\mathcal{G}_0 = \mathcal{G}$ and, for $i = 1, 2, \ldots, n$, $\mathcal{G}_i$ is the elimination of v_{i-1} from $\mathcal{G}_{i-1}$. The *width* of the elimination σ is then $\max_{i=0}^{n-1}(N(v_i, \mathcal{G}_i))$.

Elimination orderings can be used (among other characterisations) to construct algorithms, see, e.g., [29],[5].

Theorem 1.3.16 (see, e.g., [23])**.** *Let $\mathcal{G}$ be an undirected graph and let k be an integer number. Then the following statements are equivalent:*

1. *The treewidth of $\mathcal{G}$ is k.*

2. *The least number such that $\mathcal{G}$ is a partial k-tree is k.*

3. *The least number such that $\mathcal{G}$ has an elimination ordering of width k is k.*

An undirected graph is *chordal*, or *triangulated*, if has no cycle of length more than 3 with no chord, i.e., an edge (a *chord*) connecting its vertices that are not connected by cycle edges. An undirected graph $\mathcal{G}' = (V, E')$ is a *triangulation* of a graph $\mathcal{G} = (V, E)$, if $E \subset E'$ and $\mathcal{G}'$ is chordal. The following result is well known (see [17]).

Proposition 1.3.17 ([17])**.** *Let $\mathcal{G}$ be a k-tree. Then the following holds.*

- *$\mathcal{G}$ has no induced clique of size $k+2$.*
- *$\mathcal{G}$ is chordal.*

Finally we give yet another characterisation of treewidth.

Proposition 1.3.18 ([17])**.** *Let $\mathcal{G}$ be an undirected graph and let k be an natural number. The treewidth of $\mathcal{G}$ is at most k if and only if $\mathcal{G}$ is the subgraph of a chordal graph with no induced clique of size more than $k+1$.*

1.4 Pathwidth

The notion of pathwidth was introduced in [48]. Pathwidth can be seen as a special case of the treewidth, if we consider the corresponding decompositions. In terms of games, the Pathwidth Game is a variant of the Treewidth Game in which the robber is invisible, i.e., the Cop player does not generally know where the robber is. What is known (or what can be concluded) is where the robber cannot be, because the cops are on those vertices or because they have been there and the robber could not have come there, because the cops cut her way.

Pathwidth gave raise to directed pathwidth, which we shall briefly discuss speaking about measures for directed graphs.

Definition 1.4.1. Let $\mathcal{G} = (V, E)$ be an undirected graph. The *cops and invisible robber game*, or the *Pathwidth Game $PwG(\mathcal{G})$*, on $\mathcal{G}$ is a graph searching game on $\mathcal{G}$ defined by the triple $(\mathcal{L}_c, \mathcal{L}_r), \mathcal{A}$ with $\mathcal{A} = (V_A, V_0, V_1, E_A, v_I)$ where

- $\mathcal{L}_c = 2^V$, $\mathcal{L}_r = 2^V \setminus \{\emptyset\}$,
- $(X, R) \in V_0$ if R is a union of non-empty connected components of $\mathcal{G}\setminus X$,
- $(X, X', R) \in V_1$ if $(X, R) \in V_0$ and $X' \in \mathcal{L}_c$,
- $\big((X, R), (X, X', R)\big) \in E_A$, for all $(X, R) \in V_0$ and all $X' \in \mathcal{L}_c$,
- $\big((X, X', R), (X', R')\big) \in E_A$ if $R' = Reach_{\mathcal{G}\setminus(X\cap X')}(R)\setminus X'$.

As we see from the second condition, the robber occupies a *union* of the connected components where she can be, i.e., she can move to any of the components and the cops do not see to which one. This is the difference to the case of treewidth. A winning strategy of the Cop player should thus be to methodically comb through the graph under the assumption that the robber can potentially return to a vertex where she has already been expelled from. The game can be understood as a decontamination process of a graph that is contaminated by the potential presence of the robber. To be sure that the robber is not in a certain part of the graph means to decontaminate it. A return of the robber to an already decontaminated subgraph is recontamination.

Again, as with treewidth we are interested in the minimal number of cops who can capture the robber.

Definition 1.4.2. Let $k > 0$ be a natural number. The *k cops and invisible robber game* $\mathrm{PwG_k}(\mathcal{G})$ on an undirected graph $\mathcal{G}$ is the Cop and invisible robber game on $\mathcal{G}$ with the additional condition that the Cop player has only k cops, i.e., for every $X \in \mathcal{L}_c$, we have $|X| \leq k$.

Definition 1.4.3. Let $\mathcal{G}$ be an undirected graph. If k is the minimal number of cops such that the Cop player wins the Pathwidth Game $\mathrm{PwG_k}(\mathcal{G})$ on $\mathcal{G}$, then the *pathwidth* of $\mathcal{G}$ is $k - 1$.

It turns out that we can also generalise the conditions of a decomposition of a given graph to get a characterisation of pathwidth. As the name already says, the decomposition graph is a path rather than a tree. This is indeed the only difference from the tree decomposition.

Definition 1.4.4. Let $\mathcal{G} = (V, G)$ be an undirected graph. A path decomposition of $\mathcal{G}$ is a tuple $(\mathcal{P}, \mathcal{X}, f)$ where $\mathcal{P} = (P, E_P)$ is an undirected path $P = (p_0, \ldots, p_m)$ with vertices called *bags*, $\mathcal{X}$ is a subset of 2^V, and $f : P \to \mathcal{X}$ is a bijection such that

(1) $\bigcup \mathcal{X} = V$,

(2) for all graph edges $\{u, v\} \in E$, there exists a bag X with $\{u, v\} \subseteq X$,

(3) for all bags p_i, p_j, p_k, if $i < j < k$ then $f(p_i) \cap f(p_k) \subseteq f(p_j)$.

We call the images of the function f also bags and say that a graph vertex v is in a bag p if $f(p)$ contains v. The width of a tree decomposition is $\max_{p\in P}|f(p)|-1$, i.e., the size of a maximal bag.

Lemma 1.4.5. *[36] Let $\mathcal{G}$ be a graph. There is a path decomposition of $\mathcal{G}$ of width at most k if and only if the pathwidth of $\mathcal{G}$ is at most $k-1$.*

Because every path decomposition is also a tree decomposition, we have the next corollary.

Corollary 1.4.6. *Let $\mathcal{G}$ be an undirected graph. Then we have $\mathrm{tw}(\mathcal{G}) \leq \mathrm{pw}(\mathcal{G})$.*

As an example we again take undirected grids (see Example 1.3.11). It shows that the equality in the last corollary holds for some graphs.

Example 1.4.7. The pathwidth of a $m \times n$-grid $\mathcal{G}$ is $\min(m,n)$. Remember that the Cop player's strategy does not depend on where the robber is, so it is also a winning strategy of the Cop player having $\min(m,n)$ cops in the game with an invisible robber, i.e., $\mathrm{pw}(\mathcal{G}) \leq \mathrm{pw}(\mathcal{G})$. From Corollary 1.4.6 it follows that $\mathrm{pw}(\mathcal{G}) = \mathrm{pw}(\mathcal{G})$.

There are also graphs whose pathwidth is strictly greater then their treewidth (so it makes sense to introduce pathwidth).

Proposition 1.4.8. *(see e.g., [16]) The treewidth of non-empty forests one, but the pathwidth of them is unbounded.*

Proof. Let $\mathcal{T}_k$ be a (rooted) full binary tree of height k. The order of children is not important here and we speak about "left" or "right" subtrees informally, so as to distinguish between different subtrees. Then we have $\mathrm{pw}(\mathcal{T}_k) = \left\lceil \frac{k}{2} \right\rceil$. (This means $\mathrm{pw}(\mathcal{T}_k) = \left\lceil \frac{k}{2} \right\rceil + 1$ cops in the Pathwidth Game.) The proofs of both inequalities "$\leq$" and "$\geq$" are done by induction on k. All base cases are simple. For the induction step of the former inequality, if k is even, then a strategy for $k+1$ cops on $\mathcal{T}_k$ is to place the additional cop on the root and play on both subtrees with k cops the strategies given by induction hypothesis. If k is odd, strengthen the induction by demanding that the cops can win on trees with odd height having a cop on the root at the start and at the end positions. This does not affect the case with even k, as this condition holds. Now the strategy of $k+1$ cops on $\mathcal{T}_{k+1}$ (with an odd k) is to play on the right subtree of the root (then a cop is on the root of the subtree), then place another cop on the root of $\mathcal{T}_{k+1}$, then again another on the root of the left subtree and, finally, play the strategy guaranteed by the induction hypothesis on the left subtree. Note that due to the strengthened condition, it is not a constraint that a cop is on the root of the subtree.

It remains to prove the induction step of the "$\geq$"-inequality. We show that if k cops are needed to capture the robber on $\mathcal{T}_k$ then at least $k+1$ cops are needed to capture the robber on $\mathcal{T}_{k+2}$ which finishes the proof. Let r be the root of the tree with its children r_0 and r_1 and their children r_{00}, r_{01}, r_{10}, and r_{11}, respectively. We call the according subtrees $\mathcal{T}_0$, $\mathcal{T}_{00}$, $\mathcal{T}_{01}$ and so on. Again, the order of children and subtrees rooted at them is not important. To decontaminate the graph, the cops must, in particular, expel the robber from the tree $\mathcal{T}_k$ rooted at r_{11}. For this all k cops are needed. The other vertices are still contaminated. The tree rooted at r_{10} must be decontaminated as well and all k cops gather themselves in that subtree of $\mathcal{T}_{k+2}$. At the latest at that moment, the subtree rooted at r_{11} becomes recontaminated being "infected" from the subtree rooted at r_0 and the Cop player gained nothing due to the symmetry between subtrees with height k. Observe that if the subtree rooted at r_0 has already been decontaminated when the cops were decontaminating the subtree rooted at r_{10} then the subtree rooted at r_{11} was recontaminated as well "infected" from $\mathcal{T}_{10}$. $\square$

1.5 Directed treewidth

We now leave the area of undirected graphs and consider measures for directed graphs. Of course, undirected graph measures can also be used to establish the complexity of arbitrary graphs by "forgetting" the direction of edges, but in this way substantial information can be lost. For example, a clique, which has high treewidth and is thus considered to be complex, can be viewed as an undirected representation of a DAG, an intuitively simple graph.

Nevertheless, undirected measures can help in constructing directed ones. The latter should generalise the former in the following sense. Let $\mathcal{G} = (V, E)$ be a directed graph with symmetrical edge relation $E \subseteq V^2$ without self-loops, i.e., if (u, v) is in E then also (v, u). We denote by $\bar{\mathcal{G}} = (V, \bar{E})$ the undirected graph, in which the edge relation is defined by $\bar{E} = \{\{u, v\} \mid (u, v) \in E \text{ and } (v, u) \in E\}$. Conversely, if $\mathcal{G} = (V, E)$ is an undirected graph, we write $\overleftrightarrow{\mathcal{G}}$ for the directed graph $(V, \overleftrightarrow{E})$ with $\overleftrightarrow{E} = \{(u, v) \mid \{u, v\} \in E\}$. A good property of a directed measure is to correspond to (e.g., to coincide with) an undirected measure such as the treewidth, e.g., for an undirected graph $\mathcal{G}$, $\text{tw}(\mathcal{G}) = k$ if and only if the new measure is k for $\overleftrightarrow{\mathcal{G}}$.

The main difference between directed and undirected cases is that, in the former, the notions of reachability and connectivity do not coincide as they do in the latter. One possible approach to a generalisation of the treewidth to directed graphs taken by T. Johnson et al. in [35] is to replace "connected components" by "strongly connected components" in the definition of the Treewidth Game. Unfortunately, this approach does not lead to a robust measure. Unlike the treewidth, it is not known whether directed treewidth is fully characterised by the Directed treewidth Game. There are some reasons to suspect that it is not, see Chapter 2. We therefore give a definition that uses a graph decomposition, as given in [35].

Definition 1.5.1. Let $\mathcal{G} = (V, E)$ be a directed graph. Let S and Z be disjoint subsets of V. The set S is *Z-normal* if there is no directed path in $\mathcal{G} \backslash Z$ with first and last vertex in S that contains a vertex in $V \backslash (S \cap Z)$.

It follows that a set S is Z-normal in a graph $\mathcal{G}$ if and only if the sets of strongly connected components of $\mathcal{G} \backslash Z$ and its vertices that are in no strongly connected component of $\mathcal{G} \backslash Z$ can be numbered $S_0, S_1, \ldots, S_d$ such that

- if $0 \leq i < j \leq d$ then no edge leads from S_j to S_i and
- either $S = \emptyset$, or $S = S_i \cup S_{i+1}, \ldots, S_j$ for some i and j with $0 \leq i < j \leq d$. [35]

The numbering order is a linearisation of the topological order on the strongly connected components of $\mathcal{G} \backslash Z$ and its vertices that are in no strongly connected component.

Definition 1.5.2. Let $\mathcal{G} = (V, E)$ be a directed graph. An *arboreal decomposition* of $\mathcal{G}$ is a tuple $(\mathcal{R}, \mathcal{X}, \mathcal{W}, f_x, f_w)$ where $\mathcal{R} = (R, E_R)$ is an arborescence (i.e., a directed tree), $\mathcal{X}, \mathcal{W} \subseteq 2^V$, and $f_x : E_R \to \mathcal{X}$ and $f_w : R \to \mathcal{W}$ are bijections such that

- $\{f_w(r) \mid r \in R\}$ is a partition of V into non-empty sets, and

- if $e \in E_R$ then $\bigcup\{f_w(r) \mid r \in R \text{ and } r > e\}$ is $f_x(e)$-normal.

The *width* of $(\mathcal{R}, \mathcal{X}, \mathcal{W}, f_x, f_w)$ is the least number k such that, for all $r \in R$,we have $|f_w(r) \cup \bigcup_{e \sim r} f_x(e)| \leq k+1$. The *directed tree width* $\mathrm{dtw}(\mathcal{G})$ of $\mathcal{G}$ is the least number k such that $\mathcal{G}$ has an arboreal decomposition of width k.

Proposition 1.5.3 ([35])**.** *If $\mathcal{G}$ is an undirected graph then* $\mathrm{tw}(\mathcal{G}) = \mathrm{dtw}(\overleftrightarrow{\mathcal{G}})$.

We now describe the corresponding game and discuss its connection to the directed treewidth and the discrepancy between the game and the width.

Definition 1.5.4. Let $\mathcal{G} = (V, E)$ be a directed graph. The *strongly connected cops and visible robber game*, or, simpler, the *strong cops and visible robber game*, or the Directed treewidth Game *dTwG*($\mathcal{G}$), on $\mathcal{G}$ is a graph searching game on $\mathcal{G}$ defined by the triple $(\mathcal{L}_c, \mathcal{L}_r, \mathcal{A})$ with $\mathcal{A} = (V_A, V_0, V_1, E_A, v_I)$ where

- $\mathcal{L}_c = 2^V$, $\mathcal{L}_r = \{R \subseteq V \mid R \text{ is not empty and strongly connected}\} \cup \{V\}$,
- $(X, R) \in V_0$ if R is a strongly connected component of $\mathcal{G} \backslash X$,
- $(X, X', R) \in V_1$ if $(X, R) \in V_0$ and $X' \in \mathcal{L}_c$,
- $\big((X, R), (X, X', R)\big) \in E_A$, for all $(X, R) \in V_0$ and all $X' \in \mathcal{L}_c$,
- $\big((X, X', R), (X', R')\big) \in E_A$ if $R \cup R'$ is contained in a strongly connected component of $\mathcal{G} \backslash (X \cap X')$.

Definition 1.5.5. Let $k > 0$ be a natural number. The *strongly connected k cops and visible robber game*, or the *strong k cops and visible robber game*, or the Directed treewidth Game with k cops $\mathrm{TwG_k}(\mathcal{G})$ on a directed graph $\mathcal{G}$ is the strongly connected k cops and visible robber game on $\mathcal{G}$ with the additional condition that the Cop player has only k cops, i.e., for every $X \in \mathcal{L}_c$, we have $|X| \leq k$.

Example 1.5.6. (see [9]) Let $\mathbb{B}$ be the set $\{0, 1\}$. Consider $\mathcal{T}_k = (V, E)$, the directed binary tree of height $k \geq 2$ with edges oriented away from the root and with additional edges going from every vertex u to every vertex v that is on the way from the root to u except u, i.e., $V = \mathbb{B}^{\leq k}$ and $E = \{(u, v) \mid u, v \in \mathbb{B}^{\leq k}, u = vi, i \in \mathbb{B}\} \cup \{(v, u) \mid$

u is a proper prefix of $v\}$. In the Directed treewidth Game, two cops suffice to capture the robber. The winning strategy is to place a *guarding* cop on the root of the tree and then, in every cops's move, place the non-guarding cop on the successor of the vertex, where the guarding cop is and in whose subtree the robber hides. This latter cop becomes guarding and the cop who is not guarding any more makes the next move. The robber can never go to a vertex that is above the guarding cop, because in this case she would not have a way back.

One could expect that the number of cops needed to capture the robber plus one equals the directed treewidth, as it is in the undirected case. This number is indeed not greater than the directed treewidth, but we do not know whether the symmetric inequality holds. However, a weaker statement can be shown.

Proposition 1.5.7 ([35]). *Let $\mathcal{G}$ be a directed graph. Let k be an integer number. If $\mathcal{G}$ has directed treewidth at most k then the Cop player wins the Directed treewidth Game $dTwG(\mathcal{G})$ with $k+1$ cops.*

Proposition 1.5.8 ([35]). *Let $\mathcal{G}$ be a directed graph and $k > 0$ an integer. If k cops have a winning strategy in the Directed treewidth Game on $\mathcal{G}$ then the directed treewidth of $\mathcal{G}$ is at most $3k+1$.*

An interesting question about a complexity measure may be whether the complexity of a graph changes if we reverse all its edges. Robustness might seem natural. However, Hunter argues in [33] that, in the algorithmic view, the edge direction can be more critical. If the edge relation is much more difficult to compute than the successor relation (as for a graph representing computations of a Turing machine) it is substantially easier to decide whether there is a path between two vertices on an arborescence than on a tree with all edges oriented towards to root. The following proposition expresses that directed treewidth is robust in this sense. First, we define the operation of reversing edges.

Let $\mathcal{G} = (V, E)$ be a directed graph. Then $\mathcal{G}^{op} = (V, E^{op})$ is the graph with reversed edges where E is the relation $\{(v, u) \mid (u, v) \in E\}$.

Proposition 1.5.9 ([35]). *Let $\mathcal{G}$ be a directed graph. Then we have* $\mathrm{dtw}(\mathcal{G}) = \mathrm{dtw}(\mathcal{G}^{op})$.

We finally give a definition of *extended directed treewidth*, a measure without any game characterisation. It was introduced by Safari in [52]. The measure is a relaxation of the notion of the arboreal decomposition. In a *arboreal pre-decomposition* that give raise to the extended directed treewidth, the non-emptiness conditions are omitted.

Definition 1.5.10. Let $\mathcal{G} = (V, E)$ be a directed graph. An *arboreal pre-decomposition* of $\mathcal{G}$ is a tuple $(\mathcal{R}, \mathcal{X}, \mathcal{W}, f_x, f_w)$ where $\mathcal{R} = (R, E_R)$ is an arborescence, $\mathcal{X}, \mathcal{W} \subseteq 2^V$, and $f_x : E_R \to \mathcal{X}$ and $f_w : R \to \mathcal{W}$ are bijections such that

- $\{f_w(r) \mid r \in R\}$ is a partition of V into (possibly empty) sets, and
- if $e \in E_R$ then $\bigcup\{f_w(r) \mid r \in R \text{ and } r > e\}$ is $f_x(e)$-normal or empty.

The *width* of $(\mathcal{R}, \mathcal{X}, \mathcal{W}, f_x, f_w)$ is the least number k such that, for all $r \in R$,we have $|f_w(r) \cup \bigcup_{e \sim r} f_x(e)| \leq k+1$. The *extended directed tree width* $\text{extdtw}(\mathcal{G})$ of $\mathcal{G}$ is the least number k such that $\mathcal{G}$ has an arboreal pre-decomposition of width k.

1.6 DAG-width

We continue with another generalisation of treewidth to directed graphs. The only difference between the game we are going to describe and the Treewidth Game is that the robber can move along directed (rather than along undirected in the Treewidth Game) paths.

Definition 1.6.1. Let $\mathcal{G} = (V, E)$ be a directed graph. The *directed cops and visible robber game*, or the DAG Game $\text{DAGG}(\mathcal{G})$, on $\mathcal{G}$ is a graph searching game on $\mathcal{G}$ defined by the triple $(\mathcal{L}_c, \mathcal{L}_r, \mathcal{A})$ with $\mathcal{A} = (V_A, V_0, V_1, E_A, v_I)$ where

- $\mathcal{L}_c = 2^V$, $\mathcal{L}_r = 2^V \setminus \{\emptyset\}$,
- $V_0 = \{(X, R) \in \mathcal{L}_c \times \mathcal{L}_r \mid R = \mathit{Reach}_{\mathcal{G} \setminus X}(r) \text{ for some } r \in V\} \cup \{(\emptyset, V)\}$,
- $V_1 = \{(X, X', R) \in \mathcal{L}_c^2 \times \mathcal{L}_r \mid (X, R) \in V_0 \text{ and } X' \in \mathcal{L}_c\}$,
- $\big((X, R), (X, X', R)\big) \in E_A$, for all $(X, R) \in V_0$ and all $X' \in \mathcal{L}_c$,

- $\big((X, X', R), (X', R')\big) \in E_A$ if and only if there is a vertex r' with $r' \in \mathit{Reach}_{\mathcal{G}\setminus(X\cap X')}(R)$ and $R' = \mathit{Reach}_{\mathcal{G}\setminus X'}(r')$.

Definition 1.6.2. Let $k > 0$ be a natural number. The *directed k cops and visible robber game* $\mathrm{DAGG_k}(\mathcal{G})$ on an directed graph $\mathcal{G}$ is the DAG Game on $\mathcal{G}$ with the additional condition that the Cop player has only k cops, i.e., for every $X \in \mathcal{L}_c$, we have $|X| \leq k$.

Again, as in the case of directed treewidth, it can be shown that the DAG-width, which we define below, is not equal to the minimal number of cops in the directed k cops and visible robber game (see Section 2). We, therefore, define the DAG-width using decompositions.

Definition 1.6.3. Let $\mathcal{G} = (V, E)$ be a directed graph. A DAG-decomposition of $\mathcal{G}$ is a triple $(\mathcal{D}, \mathcal{X}, f)$ where $\mathcal{D} = (D, E_D)$ is a DAG with vertices called *bags*, $\mathcal{X}$ is a subset of the power set of V and $f : D \to \mathcal{X}$ is a bijection such that

(1) $\bigcup \mathcal{X} = V$,

(2) for all $d_1, d_2, d_3 \in D$ with $d_1 \leq d_2 \leq d_3$, $f(d_1) \cap f(d_3) \subseteq f(d_2)$,

(3) if r is a root of $\mathcal{D}$ then $f(r)$ is guarded by $\emptyset$,

(4) for all $(d, d') \in E_D$, $f(d) \cap f(d')$ guards $X_{\geq d'} \setminus f(d)$ where $X_{\geq d'}$ is the set $\bigcup_{d'' \geq d'} f(d'')$.

The *width* of $(\mathcal{D}, \mathcal{X}, f)$ is $\max\{f(d) \mid d \in D\}$. The DAG-width of $\mathcal{G}$ is the minimum width of any of its DAG-decompositions.

Proposition 1.6.4 ([9])**.** *Let $\mathcal{G}$ be a directed graph of DAG-width k. Then the Cop player has a winning strategy in the directed k cops and visible robber game* $\mathrm{DAGG_k}(\mathcal{G})$*.*

Proposition 1.6.4 does not provide a characterisation of DAG-width. The converse direction of the implication is however false as we shall see below. To characterise the DAG-width and to prove that the minimal number of cops needed to capture the robber on a graph can be smaller than its DAG-width we need to introduce a property of strategies.

Definition 1.6.5. A winning strategy of the Cop player in a searching game on a graph is *robber-monotone*, if in every play π consistent with this strategy, the space available for the robber is non-decreasing with respect to set inclusion. More formally, if π is $(X_0, R_0), (X_0, X_1, R_0), (X_1, R_1), \dots$ then, for all $i \geq 0$, it is $X_{i+1} \subseteq X_i$.

Proposition 1.6.6 ([9]). *Let $\mathcal{G}$ be a directed graph and let k be a natural number. The DAG-width of $\mathcal{G}$ is k if and only if the Cop player has a robber-monotone strategy in the directed k cops and visible robber game $\mathrm{DAGG_k}(\mathcal{G})$ on $\mathcal{G}$.*

The existence of a robber-monotone winning strategy is an important property that holds also for other many measures. To complete the game-theoretic characterisation of DAG-width we state that, for the DAG Game, there are graphs on that general winning strategies and robber-monotone winning strategies lead to different numbers of cops needed to capture the robber.

Proposition 1.6.7 ([39]). *For every integer $k \geq 2$, there exists a directed graph $\mathcal{G}_k$ such that $3k-1$ cops suffice to capture the robber in the DAG Game $DAGG(G)$ on $\mathcal{G}$, but at least $4k-1$ cops are needed to do this with a robber-monotone strategy*[2].

The next result shows that DAG-width is a proper generalisation of treewidth.

Proposition 1.6.8 ([9]). *Let $\mathcal{G}$ be an undirected graph and k a natural number. Then $\mathrm{tw}(\mathcal{G}) = k$ if and only if $\mathrm{DAGw}(\overleftrightarrow{\mathcal{G}}) = k-1$.*

As a next step we consider the DAG-width of certain graphs.

Proposition 1.6.9 ([9]). *Let $\mathcal{G}$ be a directed graph. Then $\mathrm{DAGw}(\mathcal{G}) = 1$ if and only if $\mathcal{G}$ is acyclic.*

It is easy to see that the following statements are true.

Proposition 1.6.10. *Let $\mathcal{G}$ be a directed graph.*

1. *If $\mathcal{G}$ is a clique with k vertices then $\mathrm{DAGw}(\mathcal{G}) = k$.*
2. *If $\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_m$ are the strongly connected components of $\mathcal{G}$ then we have $\mathrm{DAGw}(\mathcal{G}) = \max_{0 \leq i \leq m}(\mathrm{DAGw}(\mathcal{C}_i))$.*

[2]In [39], the authors prove more, namely that those numbers are minimal.

Definition 1.6.11 ([10])**.** Let $\mathcal{G} = (V, E)$ and $\mathcal{G}' = (V', E')$ be graphs. Their *asynchronous product* is the graph $\mathcal{G} \times \mathcal{G}' = (V \times V', F)$ where $F = \{((u, u'), (v, v')) \mid [(u, v \in E)$ and $v = v']$ or $[u = v$ and $(u', v') \in E']\}$. Let $\mathcal{C}_n$ be a cycle with $n > 0$ vertices, i.e., $\mathcal{C}_n = (\{0, \dots, n-1\}, E_n)$ with $E_n = \{(i, j) \mid i+1 = j \pmod n\}$. Let $m, n > 0$ be natural numbers. The *directed* $(m \times n)$*-torus* is the graph $\mathcal{T}_{mn} = \mathcal{C}_m \times \mathcal{C}_n$.

Less formally, an $m \times n$-torus can be created from the directed $m \times n$-grid by identifying the left and right border and the upper and the lower border.

Example 1.6.12. Consider a directed $m \times n$-torus $\mathcal{T}_{mn}$. If $m = n$, the DAG-width of $\mathcal{T}_{mn}$ is $m + 1$. The monotone strategy for the cops is to place m cops on a diagonal and then to pursuit the robber with the last cop until she has no room to move. The robber can defend herself from m cops by standing on a cop free column until a cop goes there. The robber remains in the column one step further than the cop. When another cop comes to the column, there is a cop free path to a cop free row in the column. She goes to the free row and then to a cop free column. There is one, because two cops are in the column where the robber hid.

Let now m be smaller than n. We show that its DAG-width is $m + 1$ or $m + 2$. The strategy for $m+2$ cops is to occupy in a column all m rows and then to expel the robber from every other column using two cops. The robber then must move to a next column and is captured when she reaches the column with m cops. The robber has the following winning strategy against m cops. She places herself on a vertex in a cop free column. Whenever a cop moves to that column, there is a cop free column and a cop free row in the torus. So the robber can proceed to the free column. In this way she escapes the cops forever.

We show next that if $n = m + 1$ then the DAG-width is $m + 1$. The cops choose an $m \times m$ subgraph and occupy one of its diagonals. There remains one cop free column c. The last cop combs all vertices outside it. Note that he does not need to know where the robber is. After that the last cop places himself on the vertex in the cop free column on the row on that a cop stands (call him C) from whom the distance to the free column is minimal (it is one). Then the cop C expels the robber from column c. The last cop remains on it. Now the robber moves to the room already combed and is captured there. By letting out the first combing we make the winning strategy of the Cop player monotone.

Analogously to treewidth, graphs that have a DAG-decomposition of width k have a decomposition of the same width that is nice.

Definition 1.6.13. A DAG-decomposition $(\mathcal{D}, \mathcal{X}, f)$ is *nice* if

1. $\mathcal{D}$ has a unique root,
2. every bag has at most two successors,
3. if d_1 and d_2 are two successors of d_0 then $f(d_0) = f(d_1) = f(d_2)$,
4. id d_1 is the unique successor of d_0 then $|f(d_0) \triangle f(d_1)| \leq 1$ where $\triangle$ is the symmetric set difference operator $(A \triangle B = (A \backslash B) \cup (B \backslash A))$.

Proposition 1.6.14 ([10])**.** *Every directed graph of DAG-width k has a nice DAG-decomposition of width k.*

Corollary 1.6.15. *The minimal number of cops needed to capture the robber in a robber-monotone way in the DAG Game on a directed graph does not change, if the Cop player is permitted to move only one cop per step.*

Speaking about directed treewidth, we mentioned the result 1.5.9 from [35] that reversing edges preserves the directed treewidth. The next proposition shows that the DAG-width is sensitive to the direction of edges.

Proposition 1.6.16 ([9])**.** *For any j, k with $1 \leq j \leq k$, there exists a directed graph $\mathcal{T}_k^j$ such that* $\mathrm{DAGw}(\mathcal{T}_k^j = j)$ *and* $\mathrm{DAGw}((\mathcal{T}_k^j)^{op} = k + 1)$

Proof. Let $\mathcal{T}_k^j$ be the directed binary tree of height k with additional edges: from every vertex to each of its descendant (such that they build a transitive closure) and from every vertex to its $j-1$ nearest ancestors.For convenience, call the former edges "forward" and the latter ones "backward". We have $\mathrm{DAGw}(\mathcal{T}_k^j) = j$. The robber-monotone winning strategy of the Cop player in the DAG Game on $\mathcal{T}_k^j$ is to place a cop on the root and then follow the robber in the leap-frogging manner to the subtree she goes to. To defeat $j-2$ cops the robber chooses a leaf and waits there until a cop drives her away from it.

At this moment, there is a cop free ancestor from that leaf and a cop free path from it to another leaf. The robber goes there and continues in this way infinitely.

For $(\mathcal{T}_k^j)^{op}$, $k+1$ cops win in the following way: one cop occupies the root. The the cops place themselves consequently from the root on the path that connects the lowest cop with the current robber's vertex in the binary tree. To defeat k cops, the robber plays the same strategy that guarantees her a win in the game on $\mathcal{T}_k^j$ against $j-2$ cops. □

1.7 Directed pathwidth

As on undirected graphs pathwidth is a variant of treewidth, on directed graphs directed treewidth is a variant of DAG-width.

Definition 1.7.1. Let $\mathcal{G} = (V, E)$ be a directed graph. The *directed cops and invisible robber game*, or the *Directed pathwidth Game dPwG($\mathcal{G}$)*, on $\mathcal{G}$ is a graph searching game on $\mathcal{G}$ defined by the triple $(\mathcal{L}_c, \mathcal{L}_r), \mathcal{A}$ with $\mathcal{A} = (V_A, V_0, V_1, E_A, v_I)$ where

- $\mathcal{L}_c = 2^V$, $\mathcal{L}_r = 2^V \setminus \{\emptyset\}$,
- $(X, R) \in V_0$ if R is a union of non-empty weakly connected components of $\mathcal{G} \setminus X$,
- $(X, X', R) \in V_1$ if $(X, R) \in V_0$ and $X' \in \mathcal{L}_c$,
- $\big((X, R), (X, X', R)\big) \in E_A$, for all $(X, R) \in V_0$ and all $X' \in \mathcal{L}_c$,
- $\big((X, X', R), (X', R')\big) \in E_A$ if $R' = \mathit{Reach}_{\mathcal{G}\setminus(X \cap X')}(R) \setminus X'$.

The game is analogous to the Pathwidth Game defined in Section 1.4. The difference is that the robber is only permitted to move along directed paths. It is clear that the Directed pathwidth Game has the same relation to the DAG Game as the Pathwidth Game to the Treewidth Game. The only difference to the DAG Game is that the robber in the Directed pathwidth Game is invisible, so the Cop player can win the latter with the same number of cops as he needs for the former.

Definition 1.7.2. Let $k > 0$ be a natural number. The *directed k cops and invisible robber game* $\mathrm{dPwG_k}(\mathcal{G})$ on a directed graph $\mathcal{G}$ is the directed cop and invisible robber game on $\mathcal{G}$ with the additional condition that the Cop player has only k cops, i.e., for every $X \in \mathcal{L}_c$, we have $|X| \leq k$.

It is not known whether the defined game fully characterises the directed pathwidth. We, therefore, give a definition based on decompositions. It makes clear that the directed pathwidth is a special case of the DAG-width except that due to the definitions of both, the directed pathwidth of a graph (say, of a directed path) can be one less than the DAG-width, although the Cop players need the same numbers of cops to win. The decomposition is a path, a special case of a DAG.

Definition 1.7.3 ([6])**.** Let $\mathcal{G} = (V, E)$ be a directed graph. A *path decomposition* of $\mathcal{G}$ is a tuple $(\mathcal{P}, \mathcal{X}, f)$ where $\mathcal{P} = (P, E_P)$ is a directed path $P = (p_0, \dots, p_m)$ with vertices called *bags*, $\mathcal{X}$ is a subset of 2^V, and $f : P \to \mathcal{X}$ is a bijection such that

(1) $\bigcup \mathcal{X} = V$,

(2) for all bags p_i, p_j, p_k, if $i < j < k$ then $f(p_i) \cap f(p_k) \subseteq f(p_j)$,

(3) for all graph edges $(u, v) \in E$, there exist indices i, j with $i \leq j$ such that $u \in f(p_i)$ and $v \in f(p_j)$,

(4) for all graph edges $(u, v) \in E$, there are no indices k, m with $k < m$ and $u \in f(p_m)$ and $v \in f(p_k)$.

We call the images of the function f also bags and say that a graph vertex v is in a bag p if $f(p)$ contains v. The *width* of a tree decomposition is $\max_{p \in P} |f(p)| - 1$, i.e., the size of a maximal bag minus one.

Definition 1.7.4. Let $\mathcal{G}$ be a graph. Its *directed pathwidth* is the least number k such that $\mathcal{G}$ has a directed path decomposition of width k.

Barát establishes connections between the pathwidth and the Directed pathwidth Game, namely that the difference between them is at most 1 [6]. The next proposition also states that robber-monotonicity can be achieved at cost of at most one cop. He also suggests that the two notions describe, in fact the same value.

Proposition 1.7.5 ([6])**.** *Let $\mathcal{G}$ be a directed graph and let $k > 0$ be a natural number. Then the following downward implications apply:*

1. *The Cop player has a robber-monotone winning strategy in the game* $\mathrm{dPwG_k}(\mathcal{G})$.

2. *The directed pathwidth of* $\mathcal{G}$ *is at most* $k-1$.

3. *The Cop player has a winning strategy in the game* $\mathrm{dPwG_k}(\mathcal{G})$.

4. *The Cop player has a robber-monotone winning strategy in the game* $dPwG_{k+1}(\mathcal{G})$.

Conjecture 1.7.6. *The minimal number of cops needed to capture a robber in the Directed pathwidth Game on a directed graph equal to the directed pathwidth of this graph minus one.*

The following proposition states that the directed pathwidth is a generalisation of the (undirected) pathwidth. Informally, the directed pathwidth applied on undirected graphs equals the pathwidth on them.

Proposition 1.7.7 ([6])**.** *Let* $\mathcal{G}$ *be an undirected graph. Then* $\mathrm{pw}(\mathcal{G}) = \mathrm{dpw}(\overleftrightarrow{\mathcal{G}})$

We now turn to some examples and simple properties of the directed pathwidth. For the first one, recall Proposition 1.6.10.

Proposition 1.7.8. *Let* $\mathcal{G}$ *be a directed graph.*

1. *If* $\mathcal{G}$ *is a clique with* k *vertices then* $\mathrm{dpw}(\mathcal{G}) = k-1$.

2. *If* $\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_m$ *are the strongly connected components of* $\mathcal{G}$ *then we have* $\mathrm{dpw}(\mathcal{G}) = \max_{0\le i\le m}(\mathrm{dpw}(\mathcal{C}_i))$.

Example 1.7.9. In Example 1.6.12 we looked at the DAG-width of tori graphs, compare Definition 1.6.11. All Cop player's strategies discussed there did not assume that the cops know where the robber hides. That means that in the Directed pathwidth Game, the Cop player needs the same number of cops to win as in the DAG Game. We exemplarily revise the strategy on quadratic tori $\mathcal{T}_{mm}$ and check that the cops move independently of where the robber is. The Cop player has m cops. Initially, $m-1$ of them occupy all vertices but one of a diagonal. For clarity, assume that the rightmost bottom vertex is not occupied. The last cop searches the left-bottom corner under the standing cops without visiting the bottom line. Then he does the same in the right-ceiling corner leaving out the rightmost border. Finitely he searches the bottom row from left to right continuing then with the rightmost column upwards.

Proposition 1.7.10. *The directed pathwidth of arborescences is 0.*

Proof. The route of the single cop in the Directed pathwidth Game is according to the depth-first search from the root in the undirected tree that we get by forgetting the direction of the edges. □

1.8 Kelly-width

We recall characterisations of the treewidth of a graph via elimination orderings and partial k-trees given in Section 1.3. In the area of directed graphs they gave raise to a measure that can be characterised in this way by adjusting the definitions to the directed case. The Kelly-width was introduced by Hunter and Kreutzer in [34].

As for the DAG-width, a definition of Kelly-width via the corresponding search game must involve monotonicity [39]. We, therefore, give a definition that uses decompositions.

Definition 1.8.1. Let $\mathcal{G} = (V, E)$ be a directed graph. A *Kelly-decomposition* of $\mathcal{G} = (V, E)$ is a tuple $(\mathcal{D}, \mathcal{B}, \mathcal{W}, f_B, f_W)$ where $\mathcal{D} = (D, E_D)$ is a DAG with vertices called *bags*, $\mathcal{B}$ and $\mathcal{W}$ are subsets of the power set of V, and $f_B : D \to \mathcal{B}$ and $f_W : D \to \mathcal{W}$ are bijections such that

(1) $\mathcal{B}$ is a partition of V,

(2) for all $d \in D$, $f_W(d)$ guards $B_{\geq d}$ where $B_{\geq d}$ is the set $\bigcup_{d' \geq d} f_B(d')$,

(3) for all $d \in D$, there is a linear order of its successors $d_1, \ldots, d_p$ so that, for all i with $1 \leq i \leq p$, $f_W(d_i) \subseteq f_B(d) \cup f_W(d) \cup \bigcap_{j<i} B_{\geq d_j}$.

The *width* of $(\mathcal{D}, \mathcal{B}, \mathcal{W}, f_B, f_W)$ is $\max\{f_B(d) \cup f_W(d) \mid d \in D\}$. The *Kelly-width* $\mathrm{Kw}(\mathcal{G})$ of $\mathcal{G}$ is the minimum width of any of its Kelly-decompositions.

Hunter [33] shows that one can always give a *special* Kelly-decomposition of the same size with additional properties, namely that for every bag $d \in D$ it holds:

- $|f_B(d)| = 1$,
- $f_W(d)$ is the minimal set which guards $B_{\geq d}$,

- every vertex $v \in B_{\geq d}$ is reachable in $\mathcal{G} \backslash f_W(d)$ from the unique $w \in f_B(d)$, and
- $\mathcal{D}$ is an arborescence.

First, we show show that the Kelly-width is a generalisation of the treewidth.

Proposition 1.8.2 ([33])**.** *Let $\mathcal{G}$ be an undirected graph and let k be a natural number. Then* $\mathrm{tw}(\mathcal{G}) = k$ *if and only if* $\mathrm{Kw}(\overleftrightarrow{\mathcal{G}}) = k + 1$.

We proceed with various characterisations of Kelly-width and start with a search game. It is played as the DAG Game with an *inert invisible* robber, i.e., the Cop player does not see where the robber is and can just deduce it from the play (invisibility) played so far, and the robber cannot move until a cop is about to land on her vertex (inertness). When moving, the robber can still run with infinite speed.

Definition 1.8.3. Let $\mathcal{G} = (V, E)$ be a directed graph. The *directed cops and inert robber game*, or the *Kelly Game $KG(\mathcal{G})$*, on $\mathcal{G}$ is a graph searching game on $\mathcal{G}$ defined by the triple $(\mathcal{L}_c, \mathcal{L}_r, \mathcal{A})$ with $\mathcal{A} = (V_A, V_0, V_1, E_A, v_I)$ where

- $\mathcal{L}_c = 2^V$, $\mathcal{L}_r = 2^V \backslash \{\emptyset\}$,
- $V_0 = \{(X, R) \in \mathcal{L}_c \times \mathcal{L}_r \mid R \cap X = \emptyset\}$,
- $V_1 = \{(X, X', R) \in \mathcal{L}_c^2 \times \mathcal{L}_r \mid (X, R) \in V_0 \text{ and } X' \in \mathcal{L}_c\}$,
- $\big((X, R), (X, X', R)\big) \in E_A$, for all $(X, R) \in V_0$ and all $X' \in \mathcal{L}_c$,
- $\big((X, X', R), (X', R')\big) \in E_A$ if and only if $R' = \big(R \cup \mathit{Reach}_{\mathcal{G} \backslash (X \cap X')}(X' \cap R)\big) \backslash X'$.

Definition 1.8.4. Let $k > 0$ be a natural number. The *directed k cops and inert robber game* $\mathrm{DAGG_k}(\mathcal{G})$ on an directed graph $\mathcal{G}$ is the DAG Game on $\mathcal{G}$ with the additional condition that the Cop player has only k cops, i.e., for every $X \in \mathcal{L}_c$, we have $|X| \leq k$.

The defined games are closely connected to the Kelly-width, but to establish correspondence we have to use the notion of a robber-monotone strategy from Definition 1.6.5.

Proposition 1.8.5 ([34])**.** *Let $\mathcal{G}$ be a directed graph and let k be a natural number. The Kelly-width of $\mathcal{G}$ is k if and only if the Cop player has a robber-monotone strategy in the directed k cops and inert robber game* $\mathrm{KG_k}(\mathcal{G})$ *on $\mathcal{G}$.*

As in the DAG Game and in contrast to the Treewidth Game, robber-monotone strategies are a substantial constraint for the Cop player.

Proposition 1.8.6 ([39]). *For every integer $k \geq 2$, there exists a directed graph $\mathcal{G}_k$ such that $6k$ cops suffice to capture the robber in the Kelly Game $KG(G)$ on $\mathcal{G}$, but at least $7k$ cops are needed to do this with a robber-monotone strategy*[3].

Another characterisation of the Kelly-width uses elimination orderings on directed graphs that are a generalisation of a similar notion (see Definition 1.3.15) on undirected graphs for the treewidth.

Definition 1.8.7. Let $\mathcal{G} = (V, E)$ be a directed graph. Let $v \in V$ be a vertex in it. A *directed elimination* of v from $\mathcal{G}$ is the graph $\mathcal{G}' = (V \backslash \{v\}, E')$ where E' is $E' = (E \backslash \{(u,v),(v,u) \mid u \in V\}) \cup \{(u,w) \mid (u,v),(v,w) \in E\}$, i.e., $\mathcal{G}'$ is the graph obtained from $\mathcal{G}$ by deleting the vertex v and all edges incident with it and replacing every two edges (u,v) and (v,w) with an edge (u,w) if it is not yet in the graph. A *directed elimination ordering* σ of $\mathcal{G}$ is a permutation $v_0, v_1 \ldots, v_{n-1}$ of V where n is the size of V.

Definition 1.8.8. Let $\mathcal{G} = (V, E)$ be a digraph and let V' be a subset of V. A *(partial) directed elimination ordering* on V' is a permutation on V'. A *directed elimination ordering* is a partial directed elimination ordering on V. Every directed elimination ordering $\sigma = v_0, v_1 \ldots, v_{n-1}$ where n is the size of V, induces a sequence of eliminations $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{n-1}$ where $\mathcal{G}_0 = \mathcal{G}$ and, for $i = 1, 2, \ldots, n$, $\mathcal{G}_i$ is the directed elimination of v_{i-1} from $\mathcal{G}_{i-1}$. The *width* of the directed elimination σ is then $\max_{i=0}^{n-1}(N^{out}(v_i, \mathcal{G}_i))$.

Proposition 1.8.9 ([33]). *Let $\mathcal{G}$ be a directed graph and let k be a natural number. Then there is a directed elimination ordering of width k if and only if the Cop player has a robber-monotone strategy in the directed $k+1$ cops and inert robber game $KG_{k+1}(\mathcal{G})$ on $\mathcal{G}$.*

Recall Definition 1.3.14 of k-trees and Proposition 1.3.16. It follows a generalisation to directed graphs.

[3]In [39], the authors prove more, namely that those numbers are minimal.

Definition 1.8.10. A k-DAG is a directed graph constructed recursively as follows:

(i) A (directed) clique K_k is a a k-DAG.

(ii) If $\mathcal{G} = (V, E)$ is a k-DAG, $X \subseteq V$ is a subset of V of size at most k, and v is a new vertex then the graph $\mathcal{G}' = (V', E')$ where $V' = V \cup \{v\}$ and $E' = E \cup \{\{v, c\} \mid c \in X\} \cup \{(u, v) \mid (u, w) \in E \text{ for all } w \in X \backslash \{u\}\}$, is a k-DAG.

In other words, we obtain a new k-DAG by adding a new vertex to an existing k-DAG and connecting it to a small subset of graph vertices in the graph without introducing new cycles.

An directed graph is a *partial k-DAG* if it is a subgraph of a k-DAG with the same vertex set.

Note that every DAG is a 0-DAG that can be constructed using the rules above by choosing always $X = \emptyset$.

For the next property recall Proposition 1.3.17. The next proposition generalises it for the case of directed graphs.

Proposition 1.8.11 ([17]). *Let $\mathcal{G}$ be a k-DAG. Then the following holds.*

- *$\mathcal{G}$ has no induced clique of size $k + 2$.*
- *Any cycle of $\mathcal{G}$ with at least tree vertices has an (directed) chord.*
- *Any bidirected cycle of at least four vertices has a bidirected chord.*

The Kelly-width has a characterisation analogous to the one given in Proposition 1.3.16.

Proposition 1.8.12 ([33]). *Let $\mathcal{G}$ be a directed graph and let k be an natural number. The Kelly-width of $\mathcal{G}$ is at most k if and only if $\mathcal{G}$ is a partial k-DAG.*

Next we look at some simple properties of the Kelly-width.

Proposition 1.8.13 ([17]). *Let $\mathcal{G}$ be a directed graph.*

1. *Let $\mathcal{G}'$ be a subgraph of $\mathcal{G}$. Then* $\mathrm{Kw}(\mathcal{G}') \leq \mathrm{Kw}(\mathcal{G})$.
2. *Let K_n a clique of size n. Then* $\mathrm{Kw}(\mathcal{G} \bullet K_n) = n \cdot \mathrm{Kw}(\mathcal{G})$.

3. *If $\mathcal{G}$ is a directed union of the directed graphs $\mathcal{G}_0$ and $\mathcal{G}_1$ then* $\mathrm{Kw}(\mathcal{G}) = \max\{\mathrm{Kw}(\mathcal{G}_0), \mathrm{Kw}(\mathcal{G}_1)\}$.

Example 1.8.14 ([17])**.** Let $\mathcal{T}_k^j$ be as in Proposition 1.6.16. The winning strategies described there are also winning strategies in the Kelly Game, so the Kelly-width and the DAG-width coincide on $\mathcal{T}_k^j$.

Example 1.8.15. Let $\mathcal{T}_{mn}$ be as in Proposition 1.6.16. Again, the winning strategies described there are also winning strategies in the Kelly Game, so we have $\mathrm{Kw}(\mathcal{T}_{mm}) = m+1$, $\mathrm{Kw}(\mathcal{T}_{m,m+1}) = m+1$ and $\mathrm{Kw}(\mathcal{T}_{mm}) \leq m+2$ for $m < n+1$.

1.9 Other measures for directed graphs

There are three other widths that we mention here. One of them, the D-width was introduced by Safari in [51].

Definition 1.9.1. Let $\mathcal{G} = (V, E)$ be a directed graph. A *D-decomposition* of $\mathcal{G}$ is a triple $(\mathcal{T}, \mathcal{W}, f)$ where $\mathcal{T} = (T, E_T)$ is an undirected tree, $\mathcal{W}$ is a subset of 2^V and $f : T \to \mathcal{W}$ is a bijection such that for every set $S \subseteq V$ that is strongly connected or a singleton the following holds:

- $T_S := \{t \in T \mid f(t) \cap S \neq \emptyset\} \neq \emptyset$, and
- the subgraph $(T_S, \{(s,t) \mid f(t) \cap f(s) \cap S \neq \emptyset\})$ of $\mathcal{T}$ forms a (connected) substree of $\mathcal{T}$.

The *width* of $(\mathcal{T}, \mathcal{W}, f)$ is the minimum k such that, for all $t \in T$, $|f(t)| \leq k+1$. The *D-width* $\mathrm{dw}(G)$ of $\mathcal{G}$ is the minimum width over all D-decompositions.

Safari conjectured that, for all directed graphs $\mathcal{G}$, $\mathrm{dw}(\mathcal{G}) = \mathrm{dtw}(\mathcal{G})$. A known result is that the directed treewidth of a graph is at least its D-width.

Proposition 1.9.2 ([51])**.** *Let $\mathcal{G}$ be a directed graph. Then we have* $\mathrm{dw}(\mathcal{G}) \geq \mathrm{dtw}(\mathcal{G})$.

The next proposition shows that the D-width is a generalisation of the treewidth.

Proposition 1.9.3 ([51])**.** *Let $\mathcal{G}$ be an undirected graph. Then* $\mathrm{tw}(\mathcal{G}) = \mathrm{dw}(\overleftrightarrow{\mathcal{G}})$.

The second measure is connected to the star height of a regular language, which is known to be equal to the cycle rank of the directed graph underlying a finite automaton that recognises the language [28].

Definition 1.9.4 ([28])**.** Let $\mathcal{G} = (V, E)$ be a directed graph. The *cycle rank* $\mathrm{cr}(\mathcal{G})$ of $\mathcal{G}$ is defined recursively as follows:

- if $\mathcal{G}$ is not strongly connected then
  - if $\mathcal{G}$ has a cycle then $\mathrm{cr}(\mathcal{G}) = \max\{\mathrm{cr}(\mathcal{C}) \mid \mathcal{C} \text{ is an SCC of} \mathcal{G}\}$,
  - if $\mathcal{G}$ has no cycle then $\mathrm{cr}(\mathcal{G}) = 0$,
- if $\mathcal{G}$ is strongly connected then $\mathrm{cr}(\mathcal{G}) = 1 + \min\{\mathcal{G}\backslash v \mid v \in V\}$.

Gruber and Holzer in [30] show that the *cyclerank* can be described as a search game similar to those that characterise the treewidth, the DAG-width and others. The rules follow the recursive definition. The cops are *immutable*, i.e., once a cop comes into the play and occupies a graph vertex, he remains there until the end of the play. The robber, in turn, has her restrictions. The first is that she is *hot-plate*, i.e., she must move along a directed path, no matter whether a cop lands on her vertex. The second restriction is that if the cops occupy a subset X of graph vertices then the robber has to go to a strongly connected component of the graph induced by the vertices that are not in X.

Definition 1.9.5. Let $\mathcal{G} = (V, E)$ be a directed graph. The *directed strong immutable cops and hot-plate visible robber game*, or the Cycle rank Game $CrG(\mathcal{G})$, on $\mathcal{G}$ is a graph searching game on $\mathcal{G}$ defined by the triple $(\mathcal{L}_c, \mathcal{L}_r, \mathcal{A})$ with $\mathcal{A} = (V_A, V_0, V_1, E_A, v_I)$ where

- $\mathcal{L}_c = 2^V$, $\mathcal{L}_r = 2^V \backslash \{\emptyset\}$,
- $V_0 = \{(X, R) \in \mathcal{L}_c \times \mathcal{L}_r \mid R \text{ is an SCC of } \mathcal{G}\backslash X\}$,
- $V_1 = \{(X, X', R) \in \mathcal{L}_c^2 \times \mathcal{L}_r \mid (X, R) \in V_0 \text{ and } X \subseteq X'\}$,
- $\big((X, R), (X, X', R)\big) \in E_A$, for all $(X, R) \in V_0$ and all $X' \in \mathcal{L}_c$,
- $\big((X, X', R), (X', R')\big) \in E_A$ if $R' \subseteq R$.

Proposition 1.9.6 ([30]). *Let $\mathcal{G}$ be a directed graph and let k be a natural number. Then $\mathcal{G}$ has cycle rank k if and only if the minimal number of cops needed to capture a robber in the Cycle rank Game is k.*

Courcelle and Olariu defined in [26] *clique-width* as a graph complexity measure that is analogous to treewidth as it measures how similar a given graph is to a clique, similar to treewidth that measures how similar a graph is to a tree. There are two notions of clique-width: one for directed and one for undirected graphs. We look only at *directed* clique-width.

Let C be a finite set of variables. A C*-labelled* graph is a structure $\mathcal{G} = (V, E, \text{lab})$ where (V, E) is a (directed) graph and lab is a *labelling function* $\text{lab} : V \to C$. We consider the following operations on graphs.

- Creation of a new graph with a single vertex labelled with a, $a \in C$.
- Renaming $\rho_{a\to b}$ of all vertices labelled with a to vertices labelled with b, for $a, b \in C$, $a \neq b$. Formally, if $\mathcal{G} = (V, E, \text{lab})$ then $\rho_{a\to b}(\mathcal{G}) = \mathcal{G}'$ where $\mathcal{G}' = (V, E, \text{lab}')$ with $\text{lab}'(v) = \text{lab}(v)$, if $\text{lab}(v) \neq a$ and $\text{lab}'(v) = b$, otherwise.
- Creating edges $\eta_{a,b}$ leading from vertices labelled with a to vertices labelled with b, for $a, b \in C$, $a \neq b$. Formally, if $\mathcal{G} = (V, E, \text{lab})$ then $\eta_{a,b}(\mathcal{G}) = \mathcal{G}'$ where $\mathcal{G}' = (V, E', \text{lab})$ with $E' = E \cup \{(v, w) \mid \text{lab}(v) = a, \text{lab}(w) = b\}$.
- Disjoint union $\oplus$ of two graphs, i.e., if $\mathcal{G} = (V, E, \text{lab})$ and $\mathcal{G}' = (V', E', \text{lab}')$ are directed graphs with $V \cap V' = \emptyset$ then $\mathcal{G} \oplus \mathcal{G}' = (V \cup V', E \cup E', \text{lab}'')$ where $\text{lab}''(v) = a$ if $\text{lab}(v) = a$ or $\text{lab}(v') = a$.

The *clique-width* $cw(\mathcal{G})$ of a graph $\mathcal{G}$ is the size of the smallest set C such that $\mathcal{G}$ can be obtained by omitting the last component of a C-labelled graph $\mathcal{G}'$ whereby $\mathcal{G}'$ is the result of applying operations from above. It is clear that an algebraic expression corresponds to each such construction and thus to each graph.

Example 1.9.7 ([26]). All directed cliques and all acyclic tournaments on at least 2 vertices have clique-width 2. All directed trees have clique-width at most 3.

At the end of this chapter we hint that yet another measure, entanglement, will be discussed. We devote an own chapter to it (Chapter 3).

Chapter 2

Properties of Measures

In this chapter we are going to consider properties that are important for complexity measures. In Section 2.1 we discuss *monotonicity*, a property of a game that corresponds to a measure as discussed in the previous chapter (see also Definition 1.6.5). It concerns the possibility for the Cop player to win in a particular nice way: either not to revisit any vertex (*cop-monotonicity*) or not to allow the robber to go to a vertex that has already been unavailable for her. Monotonicity provides methods to determine lower bounds for some measures: If the robber can violate monotonicity then she has a winning strategy. In Section 2.2 we shall see still other ways to describe a Robber's winning strategy.

2.1 Monotonicity

In the previous chapter we introduced the notion of robber-monotone strategies, see Definition 1.6.5. First, we give an related definition of a *cop-monotone* strategy.

Definition 2.1.1. A winning strategy of the Cop player in a search game, on a graph is *cop-monotone*, if in every play π consistent with this strategy, no cop returns to a vertex he has already left. Formally, if π is $(X_0, R_0), (X_0, X_1, R_0), (X_1, R_1), \ldots$ then, for all $0 \leq i \leq j \leq k$, it is $X_i \cap X_k \subseteq X_j$.

If the Cop player has a cop-monotone winning strategy in a search game then in many cases this strategy is also robber-monotone. We need some definitions to describe the restrictions to the search game with which the game has this property. A search game *permits idling*, if the robber can remain on a vertex when no cop is going to land on it.

A game is *vacating sensitive*, if whenever the robber gets a new vertex where she can go to, then she gets a vertex previously occupied by a cop. Formal definitions follow.

Definition 2.1.2 ([33]). Let $(\mathcal{L}_c, \mathcal{L}_r, \mathcal{A})$ be a search game on a graph $\mathcal{G} = (V, E)$ with $\mathcal{A} = (V, V_0, V_1, E, v_I)$. We say that it *permits idling*, if for all $(X, X', R) \in V_1$ and all $r \in R \backslash X'$, there exists $R' \subseteq V$ such that $r \in R'$ and there is an edge in E_A from (X, X', R) to (X', R'). We say that $(\mathcal{L}_c, \mathcal{L}_r, \mathcal{A})$ is *vacating sensitive*, if whenever there is an edge in E_A from (X, X', R) to (X', R') with $R' \not\subseteq R$ then $X \cap R' \neq \emptyset$.

Proposition 2.1.3 ([33]). *Let $(\mathcal{L}_c, \mathcal{L}_r, \mathcal{A})$ be a search game on a graph $\mathcal{G} = (V, E)$ that permits idling and is vacating sensitive. If σ is a cop-monotone wining strategy of the Cop player then σ is robber-monotone.*

Corollary 2.1.4. *A cop-monotone strategy of the Cop player in the Treewidth Game, in the Pathwidth Game, in the Directed treewidth Game, in the Directed pathwidth Game, or in the DAG Game is robber-monotone.*

Definition 2.1.5. A winning strategy of the Cop player in a search game is *monotone*, if it is both cop-monotone and robber-monotone.

Cop-monotone strategies bound the number of rounds of a play that the Cop player needs to win. For treewidth and pathwidth, we can construct decompositions of a given graph using monotone strategies (Cf. Proposition 2.1.6). The decompositions are linear in the size of the graph. For the treewidth, it follows from the fact that the bags of different branches of the decomposition tree do not contain common graph vertices that are not in their ancestors.

We now go through measures we defined in Chapter 1 and see whether there is always a monotone strategy of the Cop player such that he would not need more cops for it than for a general one.

Proposition 2.1.6 ([53],[37]).

1. *The Cop player has a winning strategy in the k cops Treewidth Game if and only if he has a monotone winning strategy in the k cops Treewidth Game.*

2. *The Cop player has a winning strategy in the k cops Pathwidth Game if and only if he has a monotone winning strategy in the k cops Pathwidth Game.*

On directed graphs, we unfortunately do not know any measures expressible in terms of search games that would have an analogous property, except the cycle rank, for which it is trivial. For the directed treewidth, the DAG-width, the Kelly-width, and for the Entanglement, monotone strategies would require additional cops.

For the directed treewidth, Johnson et al. who introduced the measure, proved that winning strategies and cop-monotone winning strategies do not coincide.

Proposition 2.1.7 ([35]). *There is a graph $\mathcal{G}$ where 3 cops have a winning strategy, but 4 cops are needed for a cop-monotone winning strategy in the game* $\mathrm{dtw}(\mathcal{G})$.

Adler showed by modifying the example given in [35] that neither robber-monotonicity holds.

Proposition 2.1.8 ([1]). *There is a directed graph $\mathcal{G}$ where 4 cops have a winning strategy, but they have no robber-monotone winning strategy in the game* $\mathrm{dtw}(\mathcal{G})$.

However, Kreutzer and Ordyniak remark in [39] that in [35] an upper bound for the monotonicity costs with respect to robber-monotonicity is implicitly given. Indeed, an arboreal decomposition of width k provides a robber-monotone winning strategy of the Cop player with $k+1$ cops, see Proposition 1.5.7. A general winning strategy for k cops leads, in turn, to an arboreal decomposition of size at most $3k-1$, see Proposition 1.5.8.

Corollary 2.1.9 ([39]). *Let $\mathcal{G}$ be a directed graph and let k be a natural number. If k cops have a winning strategy in the game $dTwG(\mathcal{G})$ then $3k$ cops have a robber-monotone winning strategy in the game $dTwG(\mathcal{G})$.*

We note that for the DAG-width, the notions of cop-monotone and robber-monotone strategies are the same, but to win in a monotone way, the Cop player needs in general more cops.

Proposition 2.1.10 ([9]). *If the Cop player has a cop-monotone winning strategy in the DAG Game on a graph, or a robber-monotone winning strategy then he has a monotone strategy.*

Proposition 2.1.11 ([39])**.**

1. *For every natural number $p \geq 2$, there exists a directed graph $\mathcal{G}_p$ such that the minimal number of cops who have a winning strategy in the game $DAGG(\mathcal{G})$ is $3p-1$, but the minimal number of cops who have a monotone winning strategy in $DAGG(\mathcal{G})$ is $4p-2$.*

2. *For every natural number $p \geq 2$, there exists a directed graph $\mathcal{G}_p$ such that the minimal number of cops who have a winning strategy in the game $KG(\mathcal{G})$ is $6p$, but the minimal number of cops who have a monotone winning strategy in $KG(\mathcal{G})$ is $7p$.*

As the last measure with a "non-monotone" game we look at the entanglement. The Entanglement Game is neither cop-monotone nor robber-monotone.

Proposition 2.1.12. *Let $m \geq 6$ be a natural number. Let $\mathcal{P}_m = (P_m, E_m)$ be the undirected path with shortcuts in one direction, i.e., $P_m = \{0, 1, \dots m-1\}$ and $E_m = \{(i,j) \mid 0 \leq i < j \leq m-1\} \cup \{(j,i) \mid 0 \leq i = j+1 \leq m-1\}$. Then the entanglement of $\mathcal{P}_m$ is two, but there is no cop-monotone or robber-monotone winning strategy of the Cop player.*

Proof. One cop cannot capture the robber even on $\mathcal{P}_4$ and two cops have the following strategy. One goes to the vertex that the robber chooses in the first move. Due to the symmetry, we can assume that the robber goes to the right. The second cop follows her until she goes to the right of the second (and the first) cop. Then the first cop takes the role of the second one and so on.

We show that there is neither a cop-monotone nor a robber-monotone strategy. The strategy of the Robber player is to choose vertex 0 and switch between 0 and 1 until a cop goes into the play. Then she "jumps" over a shortcut to the rightmost vertex $m-1$ and switches between $m-1$ and $m-2$ until a cop follows her. If it is the first one then there is a cop free path to the vertices 0 and 1, one of which was already unavailable for the robber. She switches between 0 and 1 and a cop has to go there again. So both kinds of monotonicity are disproved in this case. If the second cop comes on $m-1$ or $m-2$, the robber switches between $m-3$ and $m-4$. Now one of the cops has to move and the situation is analogous to the one considered above. □

In Chapter 3 we prove a kind of weak monotonicity for the Entanglement Game in a special case of two cops and conjecture that it holds also in the general case.

2.2 Havens and brambles

This section is devoted to several methods to prove bounds for complexity measures on concrete classes of graphs: havens, brambles and separators. We already defined brambles in Section 1.3, see Definition 1.3.9. While decompositions provide a way to establish an upper bound for the corresponding widths, they do not guarantee a lower bound. Indeed, for every measure (if decompositions are defined for them), there always exist a decomposition consisting of only one bag, but its width is roughly the number of vertices in the graph. To provide a lower bound, we need a method to prove that the robber escapes from k cops. One of such methods is to give a function that to every arrangement of at most k cops assigns a set of vertices where the robber should hide such that when the cops move, she can reach the new "cover" prescribed by the function for the new arrangement.

Remember the notion of touching sets in a graph from Definition 1.3.9.

Definition 2.2.1 ([35]). Let $\mathcal{G} = (V, E)$ be an undirected graph and $k \geq 1$ a natural number. Let X and Y be sets of vertices of $\mathcal{G}$. The sets X and Y *touch* if either $X \cap Y \neq \emptyset$ or there is an edge connecting an vertex from X with a vertex from Y. The set of vertices of a connected component of $\mathcal{G} \setminus X$ is an *X-flap*. A *haven* in $\mathcal{G}$ of order k is a function $h : [V]^{<k} \to 2^V$ where $h(X)$ is an X-flap and, for all $X, Y \in [V]^{<k}$ with $Y \subseteq X$, $h(X) \subseteq h(Y)$. The *haven number* $\operatorname{hn}(\mathcal{G})$ of $\mathcal{G}$ is the largest k such that $\mathcal{G}$ has a haven of order k.

It is clear that a haven of order k of a graph induces a winning strategy of the Robber player in the Treewidth Game $TwG(\mathcal{G})$: whenever at most $k{+}1$ cops leave a set Y vertices to occupy the set X (whereby X and Y may have a non-empty intersection), the robber runs from the Y-flap $h(Y)$ where she was to the X-flap $h(X)$, which is possible, because $h(X)$ and $h(Y)$ touch. It turns out that, for the treewidth, also the other direction of the implication holds: if the robber has a winning strategy, he has a winning strategy that is induced by a haven.

Proposition 2.2.2 ([53]). *Let $\mathcal{G}$ be an undirected graph. Then it has treewidth at least $k-1$ if and only if it has a haven of order at least k.*

We do not have a measure for directed graphs that would be fully analogous to Proposition 2.2.2. Johnson et al. analyse the connection between havens on directed graphs and the directed treewidth.

Definition 2.2.3. Let $\mathcal{G} = (V, E)$ be a directed graph and k a natural number. A *(directed) haven* of order k is a function $h : [V]^{<k} \to 2^V$ such that, for all $X \in [V]^{<k}$, $h(X)$ is a strongly connected component of $\mathcal{G}\backslash X$ and, for all vertex sets X, Y with $|X| < k$ and $|Y| < k$, if $X \subseteq Y$ then $h(Y) \subseteq h(X)$.

We mention that minimal orders of havens correspond to the minimal number of cops needed to capture a robber in the Directed treewidth Game. If a directed graph $\mathcal{G}$ has a haven of order k then the robber has a winning strategy described by the haven. Conversely, any strategy of the robber against less than k cops defines, for every set X that is occupied by cops a vertex v_X where the robber hides. We extend the vertex to a maximal set S_X containing the strongly connected component of $\mathcal{G}\backslash X$ with v_X and all vertices w with $w \in \mathit{Reach}_{\mathcal{G}\backslash X}(v_X)$ such that the robber still has a winning strategy from each of these w. Then the function $h(X) := S_X$ is a haven of order k.

Proposition 2.2.4 (see [35]). *Let $\mathcal{G}$ be a directed graph. Then the Robber player has a winning strategy in the Directed treewidth Game if and if $\mathcal{G}$ has a haven of order k.*

Proposition 2.2.5 ([35]). *Let $\mathcal{G} = (V, E)$ be a directed graph and $k \geq 1$ a natural number. If $\mathcal{G}$ has a haven of order k then the directed treewidth of $\mathcal{G}$ is at least $k - 1$.*

Adler shows in [1] that the other direction fails.

Proposition 2.2.6 ([1]). *There is a directed graph with directed treewidth at least 4, which has no directed haven of order 5.*

Nevertheless, in [35], an upper bound for the discrepancy between havens and the directed treewidth is given.

Proposition 2.2.7 ([35]). *Let $\mathcal{G}$ be a directed graph and let $k \geq 1$ be a natural number. If the directed treewidth of $\mathcal{G}$ is k then $\mathcal{G}$ has a directed haven of order $3k - 1$.*

Hunter defines in his dissertation the D-havens and relates them to the DAG-width.

Definition 2.2.8 ([33])**.** Let $\mathcal{G} = (V, E)$ be a directed graph and $k \geq 1$ a natural number. A *D-haven* of order k is a function $h[V]^{<k} \to 2^V$ such that, for all $X \subseteq V$ with $|X| < k$:

- $h(X)$ is a non-empty subset of $V \backslash X$, and
- if $Y \subseteq X$ then $h(X) \subseteq h(Y)$ and, for all $y \in h(Y)$, $h(X) \cap \mathit{Reach}_{h(Y)}(y) \neq \emptyset$.

Observe that $h(X)$ must not be connected.

Proposition 2.2.9 ([33])**.** *Let $\mathcal{G}$ be a directed graph. Then the Robber player has a winning strategy against k cops in the DAG Game if and only if $\mathcal{G}$ has a D-haven of order $k + 1$.*

Although the DAG Game does not characterise the DAG-width, havens give a lower bound for the DAG-width.

Corollary 2.2.10 ([33])**.** *Let $\mathcal{G}$ be a directed graph. Let k be a natural number. If $\mathcal{G}$ has a D-haven of order k, then the DAG-width of $\mathcal{G}$ is at least k.*

Obdržálek gives another variant of directed haven (that resembles brambles) that characterises the existence of winning (but not monotone) strategies of the Cop player in the DAG Game.

Proposition 2.2.11 ([42])**.** *Let $\mathcal{G}$ be a directed graph and let $k \geq 1$ be a natural number. Then the Robber player has a winning strategy in the DAG Game on $\mathcal{G}$ if and only if there is a function $\sigma : [V]^{<k} \to 2^V$ that maps a vertex set X to a non-empty union of strongly connected components of $\mathcal{G} \backslash X$ such that if $X \subseteq Y \in [V]^{<k}$ then*

1. *for all sets $S \in \sigma(X)$ there is a set $T \in \sigma(Y)$ such that there is a directed path from S to T in $\mathcal{G} \backslash X$, and*
2. *for all sets $S \in \sigma(Y)$ there is a set $T \in \sigma(X)$ such that there is a directed path from S to T in $\mathcal{G} \backslash X$.*

As havens define strategies of the Robber player and games with an invisible robber (as in the Kelly Game) are essentially one-player games (where the robber has only one strategy), we cannot expect that havens would help us in establishing lower bounds for corresponding measures. So we do not consider the Kelly-width and the pathwidth in in connetion with havens.

Brambles supply efficient algorithms for constant approximation algorithms for planar graphs, see [18] where an algorithm for computing (not necessarily minimal) brambles on general undirected graphs is given..

Bramble numbers are closely connected to *linkedness* and *well linkedness*. In fact, these three invariants are essentially the same (see [44]).

Definition 2.2.12 ([44])**.** Let $\mathcal{G}$ be a directed or an undirected graph and let S be a set of its vertices. Let k be an integer number. The set S is *k-linked*, if for any set X of at most $k-1$ vertices there is a (unique) (strongly) connected component of $\mathcal{G}\backslash X$ containing more than a half of vertices of S. The *linkedness* $\operatorname{link}(\mathcal{G})$ of $\mathcal{G}$ is the largest k, for which $\mathcal{G}$ contains a k-linked set.

Definition 2.2.13 ([44])**.** Let $\mathcal{G}$ be a directed or an undirected graph and let S be a set of its vertices. The set S is *well linked*, if, for every pair X, Y of subsets of S with $|X| = |Y|$, there are $|X|$ disjoint paths from X to Y in $\mathcal{G}\backslash\big((S\backslash X)\backslash Y\big)$. The *well linkedness* $\operatorname{wlink}(\mathcal{G})$ of $\mathcal{G}$ is the size of the largest well linked set in $\mathcal{G}$.

The brambles, which we have not defined for directed graphs can be generalised in the following way.

Definition 2.2.14 ([44])**.** Let $\mathcal{G} = (V, E)$ be a directed graph and $k \geq 1$ a natural number. Let X and Y be sets of vertices of $\mathcal{G}$. The sets X and Y *strongly touch* if either $X \cap Y \neq \emptyset$ or there are edges from a vertex in X to a vertex in Y and from a vertex in Y to a vertex in X. A *directed screen* or a *directed bramble* $\mathcal{S}$ in $\mathcal{G}$ is a set of mutually strongly touching strongly connected subsets (or singletons) of V. A directed screen (a directed bramble) $\mathcal{S}$ has *thickness* $\geq k$ if there is no $X \in [V]^{<k}$ such that $X \cap H \neq \emptyset$ for all $H \in \mathcal{S}$.

Proposition 2.2.15 ([44])**.** *For every undirected graph $\mathcal{G}$, we have*

$$\operatorname{link}(\mathcal{G}) \leq \operatorname{bn}(\mathcal{G}) \leq \operatorname{wlink}(\mathcal{G}) \leq 4 \cdot \operatorname{link}(\mathcal{G}).$$

On directed graphs, brambles and havens also define roughly the same invariant.

Proposition 2.2.16 ([51])**.** *Let $\mathcal{G}$ be a directed graph. Then we have*

$$\lceil \frac{hn(\mathcal{G})}{2} \rceil \leq \text{bn}(\mathcal{G}) \leq \text{hn}(\mathcal{G})$$

and for each inequality there are graphs for that the equality holds.

Safari in [51] used these results to extend the havens and the brambles characterisations of the treewidth to the D-width, but he needed to restrict the class of directed graphs.

For the DAG-width and for the Kelly-width, one can define two kinds of brambles such that they correspond to the measures.

Definition 2.2.17 ([33])**.** Let $\mathcal{G} = (V, E)$ be a directed graph. Let $\mathcal{H} = (V_H, E_H)$ be a strongly connected component of $\mathcal{G}$ or a single vetrex. Then $\mathcal{H}$ is an *initial component* if it is closed under predecessors, i.e., if $v \in V$ with $(v, w) \in E$ for some $w \in V_H$ then $v \in V_H$. Analogously, $\mathcal{H}$ is a *terminal component* if it is closed under successors, i.e., if $v \in V$ with $(w, v) \in E$ for some $w \in V_H$ then $v \in V_H$. Denote by $Init(\mathcal{G})$ the set of all vertices in initial components and $Term(\mathcal{G})$ the set of all vertices in terminal components. For a subset of vertices $B \subseteq V$ we write $Init(B)$ and $Term(B)$ for $Init(\mathcal{G}[B])$ and $Term(\mathcal{G}[B])$ respectively when it is clear what $\mathcal{G}$ is.

Definition 2.2.18 ([33])**.** Let $\mathcal{G} = (V, E)$ be a directed graph. An *initial bramble* in $\mathcal{G}$ is a set $\mathcal{B}$ of subsets of V such that, for all pairs $B, B' \in \mathcal{B}$ and for all $x \in Init(B)$, there exists $y \in Init(B')$ such that $y \in Reach_{B \cup Init(B')}(x)$.

Definition 2.2.19 ([33])**.** Let $\mathcal{G} = (V, E)$ be a directed graph. An *terminal bramble* in $\mathcal{G}$ is a set $\mathcal{B}$ of subsets of V such that, for all pairs $B, B' \in \mathcal{B}$ and for all $x \in Term(B)$, there exists $y \in Term(B')$ such that $y \in Reach_{Term(B) \cup B'}(x)$.

Proposition 2.2.20 ([33])**.** *Let $\mathcal{G}$ be a directed graph and let k be a natural number.*

1. *If $\mathcal{G}$ has an initial bramble of order k then $\mathcal{G}$ has DAG-width at least k.*
2. *If $\mathcal{G}$ has an terminal bramble of order k then $\mathcal{G}$ has Kelly-width at least k.*

2.3 Complexity of computing measures

In this section we discuss how difficult it is to compute a particular measure on a given graph. For measures that have decompositions, we are also interested in the complexity of computing decompositions. One of the motivations for this is that given decompositions of bounded width often lead to fast algorithms for problems that are difficult otherwise (for example, NP-complete). There is no measure among those we discussed in Chapter 1 that could be computed in deterministic polynomial time, as far as we know. There are, however, certain graph classes on which this is possible.

We also discuss graph classes for which we either know their measures or can estimate them.

Proposition 2.3.1 ([3])**.** *The following decision problem is* NP*-complete:*

Given: *An undirected graph $\mathcal{G}$ and a natural number k.*

Question: *Is the treewidth of $\mathcal{G}$ at most k:* $\mathrm{tw}(\mathcal{G}) \leq k$*?*

The membership in NP is easy to prove: one has to guess, for example, an elimination ordering, which has linear size, and then to check whether its order is at most k. If a tree decomposition of width at most k is needed we can guess such a decomposition instead of a elimination ordering.

Note that in Proposition 2.3.1 the parameter k is a part of input. If k is fixed the problem becomes polynomially solvable.

Proposition 2.3.2 ([14])**.** *For any fixed integer k, there is a linear-time algorithm that tests whether a given undirected graph $\mathcal{G}$ has treewidth at most k, and if so, outputs a tree decomposition of $\mathcal{G}$ with treewidth at most k.*

Proposition 2.3.3 ([3])**.** *The following decision problem is* NP*-complete:*

Given: *An undirected graph $\mathcal{G}$ and a natural number k.*

Question: *Is the pathwidth of $\mathcal{G}$ equal to k:* $\mathrm{pw}(\mathcal{G}) = k$*?*

For deterministic algorithms for computing the treewidth and the pathwidth on special graph classes and known treewidths and pathwidths on some graphs see [31], [19], [46], [47], [13].

We now turn to directed measures. The following result originates from [9]. It also follows from Proposition 2.3.1 using Proposition 1.6.8.

Proposition 2.3.4 ([9], see also [33]). *The following decision problem is* NP*-hard:*

Given: *A directed graph $\mathcal{G}$ and a natural number k.*

Question: *Is the DAG-width of $\mathcal{G}$ at least k: $\mathrm{DAGw}(\mathcal{G}) \leq k$?*

We do not know whether the problem from the last proposition is in NP, but it seems to be so [33]. The problem with an approach to compute the DAG-width nondeterministically by guessing a minimal decomposition is that the best bound for the size of a DAG-decomposition is $\mathcal{O}(n^k)$ where n is the size of the graph and k is the DAG-width, which is not bounded in n in general. Brambles and similar characterisations that work for the treewidth, are not applicable on the DAG-width, because they correspond to general Cop's strategies in the DAG Game rather than to monotone ones as needed for the DAG-width.

When the DAG-width k is known, a DAG-decomposition can be computed in time $|\mathcal{G}|^k$ [9].

The problem, given a directed graph $\mathcal{G}$ and natural number k, decide whether the graph has Kelly-width at most k is also NP-complete. The hardness follows (as for the DAG-width) from the fact that the Kelly-width generalises the treewidth (see 1.8.2). For membership, note that that a Kelly-decomposition is polynomial in the size of the graph [34]. A nondeterministic algorithm guesses the Kelly-decomposition and then tests in polynomial time whether it suffices all requirements of a Kelly-decomposition.

Unlike for the DAG-width, it is an open question, what is the complexity of deciding whether a given $\mathcal{G}$ graph has Kelly-width at most k for a fixed k. For a given k, we can compute the Kelly-width in exponential time and space.

Proposition 2.3.5 ([33]). *Given a directed graph $\mathcal{G} = (V, E)$ the Kelly-width of $\mathcal{G}$ can be computed in*

- *time $\mathcal{O}(|V| + |E| \cdot 2^{|V|})$ and space $\mathcal{O}(|V| \cdot 2^{|V|})$, or*
- *time $\mathcal{O}(|V| + |E| \cdot 4^{|V|})$ and space $\mathcal{O}(\cdot 2^{|V|})$.*

For $k = 2$, there is a polynomial algorithm to test whether a given graph has Kelly-width at most 2. For larger k, the complexity of this problem remains an open question.

Proposition 2.3.6 ([41]). *There is a deterministic polynomial-time algorithm that tests whether a given graph $\mathcal{G}$ has Kelly-width at most* 2*.*

The directed treewidth can be nondeterministically computed in polynomial time. Let $(\mathcal{R}, \mathcal{X}, \mathcal{W}, f_x, f_w)$ be an arboreal decomposition of a directed graph $\mathcal{G} = (V, E)$. It is easy to see that the decomposition has polynomial size in the size $|V| + |E|$ of $\mathcal{G}$. So as to decide, whether $\mathcal{G}$ has directed treewidth at most k we guess a tuple $(\mathcal{R}, \mathcal{X}, \mathcal{W}, f_x, f_w)$ and test in polynomial time that a this tuple is actually an arboreal decomposition. The NP-completeness for the directed treewidth follows analogously as for the DAG-width and the Kelly-width.

We finally consider the complexity of determining the directed pathwidth of a graph. The same question about entanglement is postponed until Chapter 3.

Proposition 2.3.7 ([40]). *The following decision problem is* NP*-complete:*

Given: *A directed graph $\mathcal{G}$ and a natural number k.*

Question: *Is* $\mathrm{dpw}(\mathcal{G}) \leq k$*?*

Restricted on trees, the problem can be decided intime PTIME.

Proposition 2.3.8 ([40]). *There is a deterministic linear-time algorithm that computes the directed treewidth of a given directed graph.*

2.4 Solving difficult problems for bounded measure

An important application of complexity measures of graphs is the construction of efficient algorithms for generally difficult (e.g., NP-complete) problems on graphs with complexity measures bounded by a constant. Pathwidth served as a model for measures we consider here. We first describe the general approach, which uses tree decompositions.

A useful property of tree decompositions is that every bag that is not a leaf in the decomposition tree is a separator in the graph, i.e., it divides the graph into two parts with no edges between them. Formally, let $\mathcal{G} = (V, E)$ be a graph and let $S \subseteq V$ be a set of its vertices. We say that S is a separator, if $\mathcal{G} \backslash S$ is not connected.

This property of tree decompositions allows to construct algorithms that combine partial solutions for graphs induced by subtrees rooted at a bag's successors in a common solution for the graph induced by the subtree rooted at the bag. This technique appeared in [4], see also [8],we describe now this technique in more detail, as it served as an example for analogous techniques on directed graphs. We give only a brief description, details can be found in [17].

Consider an undirected graph $\mathcal{G} = (V, E)$ of treewidth k with a fixed natural number k. Consider a fixed problem.

1. Find a nice tree decomposition $(\mathcal{T}, \mathcal{X}, f)$ of $\mathcal{G}$ of width k. This can be done in linear time (Proposition 2.3.2 and Proposition 1.3.7). Choose a root r of $\mathcal{T}$.
2. Compute in a bottom-up manner (i.e., using dynamic programming), for each bag a table containing certain information. What information is computed depends on the problem. For the computation of the table, one only uses the tables of the children, the type of the bag (*leaf*, *join*, *introduce*, *forget*) and the information about $\mathcal{G}\backslash X$.
3. The answer to the problem can be extracted from the table of the root bag.
4. Construction versions usually need another step where tables are used again to construct a solution (if one exists).

In this way many NP-hard problems can be solved efficiently on graphs with bounded treewidth, see [46, 2, p.222] for a list of references. Examples of such problems are Independent Set, Hamiltonian Circuit, Vertex Cover. For Pspace-hard problems that becomes easy if the treewidth is bounded see for example [19, p.359] and [13].

A class of efficiently solvable problems are those definable in MSO [24]. MSO is an extention of FO by allowing *monadic* quantifiers for sets of vertices and sets of edges and the membership relation $\in$. An analogous result for graphs with bounded clique-width is shown in [25].

Directed treewidth can also lead to fast algorithms. Johnson et al. define *k-linkage* and introduce *itineraries*, a concept similar to information tables used with treewidth. We now follow [35] and describe how these notions help in constructing algorithms.

Let $\mathcal{G} = (V, E)$ be a directed graph, let $S \subseteq V$ be a set of vertices and let k be a natural number.Recall the definition of a Z-normal set (Definition 1.5.1). Define the set S to be *k-protected*, if it is Z-normal for a set $Z \subseteq V$ with $|Z| < k$. The genearal algorithm scheme introduced in [35] prescribes to compute certain information, the itinerary, for k-protected subsets of V. The latter can be characterised axiomatically. We require that, for every natural number k there is a real number α and two algorithms such that the following axioms hold.

Axiom 1. *Let $\mathcal{G} = (V, E)$ be a directed graph and let $A, B \subseteq V$ be disjoint sets such that no edge leads from a vertex in B to a vertex in A. Then an itinerary for $A \cup B$ can be computed from itineraries for A and B in time $\mathcal{O}((|A| + |B|)^\alpha)$.*

Axiom 2. *Let $\mathcal{G} = (V, E)$ be a directed graph and let $A, B \subseteq V$ be disjoint sets such that A is k-protected and $|B| \leq k$. Then an itinerary for $A \cup B$ can be computed from itineraries for A and B in time $\mathcal{O}((|A| + 1)^\alpha)$.*

Hereby the constants hidden in the $\mathcal{O}$ notation depend on k, but not on $\mathcal{G}$, A or B.

Proposition 2.4.1 ([35])**.** *For every natural number $k \geq 1$ there exists an algorithm satisfying the following specifications.*

Input *A directed graph $\mathcal{G} = (V, E)$ and an arboreal decomposition $(\mathcal{R}, \mathcal{X}, \mathcal{W}, f_x, f_w)$ of $\mathcal{G}$ of width at most $k - 1$.*

Output *An itinerary for $\mathcal{G}$.*

Running time: $\mathcal{O}(|V|^{\alpha+1})$*, assuming Axioms 1 and 2 are satisfied.*

The algorithm resembles the scheme described for the graphs with bounded treewidth. It uses the arboreal decomposition and the normality of sets that induce subtrees similar to the algorithms for the treewidth, which use that the vertex sets inducing the subtree of a tree decomposition are disjoint.

Definition 2.4.2 ([35])**.** Let $\mathcal{G} = (V, E)$ be a graph. A *linkage* of $\mathcal{G}$ is a subgraph $\mathcal{L}$ of $\mathcal{G}$ such that every weakly connected component of $\mathcal{L}$ is a directed path. Let $\sigma = (s_1, t_1, s_2, t_2, \ldots, s_m, t_m)$ be a sequence of $2m$ vertices of $\mathcal{G}$ (not necesseraly distinct). A linkage $\mathcal{L}$ is a *σ-linkage* if the weak components of $\mathcal{L}$ can be numbered $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m$ in such a way that, for $1 \leq i \leq m$, the directed path P_i has first vertex s_i and last vertex t_i.

Proposition 2.4.3 ([35]). *Let $k \geq 1$ and $m \geq 0$ be natural numbers. Then there is a polynomial-time algorithm that solves the following problem.*

Given: *A directed graph $\mathcal{G} = (V, E)$, an arboreal decomposition $(\mathcal{R}, \mathcal{X}, \mathcal{W}, f_x, f_w)$ of $\mathcal{G}$ of width at most $k-1$, a sequence σ of $2m$ vertices of $\mathcal{G}$, and a set $M \subseteq \{1, 2, \ldots, |V|\}$.*

Question: *Does there exist a σ-linkage $\mathcal{L} = (V_L, E_L)$ in $\mathcal{G}$ with $|V_L| \in M$?*

The linkage problem with a fixed number of terminals (given a sequence σ as above, is there a σ-linkage),Hamilton Path, Hamilton Circuit, Hamilton Path with Prescribed Ends, the problem whether there is an even cycle containing a given vertex are problems that can be solved with the algorithm guaranteed by Proposition 2.4.3. If the arboreal decomposition is not given, but it is known that the graph has directed treewidth at most $k-1$, we can efficiently construct an arboreal decomposition of width at most $3k+1$ and then proceed with the known algorithm replacing k with $3k+2$.

We shall see in Chapter 4 that DAG-width and Kelly-width are special cases of directed treewidth. So problems that can be efficiently solved on graphs with bounded directed treewidth can also be solved on graphs with bounded DAG-width or Kelly-width.

For Kelly-width, Hunter and Kreutzer give in [34] a similar scheme as for the directed treewidth. In a bottom-up manner, one computes the needed data set for the *leaves*, *combines* data sets for successors of a bag, then *updates* the information taking the bag itself into account, and finally *expands* the data set to include the guards of the sub-DAG.

A *weighted* (directed or undirected) graph is a structure $(\mathcal{G}, w)$ where $\mathcal{G} = (V, E)$ is a graph and $w : V \to (R)$ is a weight function of its vertices.

Definition 2.4.4 ([34]). The Weighted w-Linkage problem is given a weighted graph $(\mathcal{G}, w)$, a sequence $\sigma = (s_1, t_1, s_2, t_2, \ldots, s_m, t_m)$, and a set $M \subseteq \{1, 2, \ldots, |V|\}$, to compute for each $l \in M$ an s-linkage $\mathcal{L}_l$ with $|\mathcal{L}| = l$ of minimal weight.

Proposition 2.4.5 ([34]). *The* Weighted w-Linkage *problem on directed graphs with bounded Kelly-width can be solved in polynomial time.*

Another problem that allows a polynomial algorithm on graphs with bounded Kelly-width is Parity, the problem to decide whether in a given parity game Player 0 has a winning strategy. See for a description Section 1.1. It is unlikely to be NP-complete, but no deterministic polynomial algorithm is known. We know that PARITY$\in$ NP$\cap$co$-$NP.

Proposition 2.4.6 ([33])**.** *Let k be a fixed natural number. Let $\mathcal{G}$ be a directed graph of Kelly-width k and let w be a weight function on $\mathcal{G}$. Then there is a polynomial-time algorithm that decides the* PARITY *problem on $(\mathcal{G}, w)$.*

Note that an analogous result for the directed treewidth is not known.

The concept of DAG-width also leads to a fast algorithm to compute PARITY [9]. Besides that, as already mentioned above, all problems that are efficiently solvable on graphs with bounded directed treewidth can be efficiently solved on graphs with bounded DAG-width, see Chapter 4.

Proposition 2.4.7 ([43])**.** *Let k be a fixed natural number. Let $\mathcal{G}$ be a directed graph of clique-width k and let w be a weight function on $\mathcal{G}$. Then there is a polynomial-time algorithm that decides the* PARITY *problem on $(\mathcal{G}, w)$, provided a corresponding algebraic expression describing the construction of $\mathcal{G}$ is given.*

For the entanglement measure, there is until now only one problem of this kind, namely PARITY. We shall discuss the algorithm of Berwanger and Grädel in Chapter 3. One reason for this may be that we do not know any decomposition of a graph with small entanglement in parts that are in some sense independent.

As a conclusion of this chapter we state some boundaries to algorithmical application of cyclicity measures on directed graphs given by Kreutzer and Ordyniak in [39]. While problems that are similar to the linkage problem allow fast algorithms on graphs with small cyclicity (i.e., graphs with small values of measures we discuss), only few other problems have this property. Some explicitly directed problems that are NP-complete even on "almost acyclic" graphs follow.

Definition 2.4.8. The MINIMUM EQUIVALENT SUBGRAPH problem is stated as follows.

Given: A directed graph $\mathcal{G} = (V, E)$ and a natural number k.

Question: Is there a set of edges $E' \subseteq E$ with $|E'| \leq k$ such that the directed graph $\mathcal{G}' = (V, E')$ contains a path between two vertices if and only if such a path exists in $\mathcal{G}$.

Proposition 2.4.9 ([39])**.** *The* MINIMUM EQUIVALENT SUBGRAPH *problem is* NP-*complete on graphs with DAG-width less than* 4.

Definition 2.4.10. The Feedback Vertex (Arc) Set problems are as follows.

Given: A directed graph $\mathcal{G} = (V, E)$ and a natural number k.

Question: Is there a set of vertices (edges) $V' \subseteq V$ ($E' \subseteq E$) with $|V'| \leq k$ ($|E'| \leq k$) such that the directed graph $\mathcal{G}[V']$) ($\mathcal{G}' = (V, E \backslash E')$) contains no cycles?

Proposition 2.4.11 ([39])**.** *The* Feedback Vertex (Arc) Set *problems are* NP-*complete on graphs with DAG-width at most* 4*.*

Definition 2.4.12. The Graph Grundy Numbering problem is as follows.

Given: A directed graph $\mathcal{G} = (V, E)$.

Question: Is there a function $f : V \to \omega$ such that, for all $v \in V$, $f(v)$ is the smallest natural number not contained in $\{f(u) \mid u \in V, (v, u) \in E\}$?

Proposition 2.4.13 ([39])**.** *The* Graph Grundy Numbering *problem is* NP-*complete on graphs with DAG-width at most* 2*.*

Definition 2.4.14. The Kernel problem is as follows.

Given: A directed graph $\mathcal{G} = (V, E)$.

Question: Is there an independent set $V' \subseteq V$ such that, for every $v \in V \backslash V'$, there exists a $u \in V'$ with $(v, u) \in E$?

Proposition 2.4.15 ([39])**.** *The* Kernel *problem is* NP-*complete on graphs with DAG-width at most* 2*.*

Chapter 3

Entanglement

Entanglement is a measure that (unlike measures we considered in previous chapters) was defined in terms of games [10]. It proved its theoretical importance in showing that the variable hierarchy in μ-calculus is strict [11]. We, however, concentrate on entanglement as a complexity measure for directed graphs do not consider its connection to the μ-calculus.

We do not know any graph decomposition that would correspond to entanglement in a way as there are decompositions for other measures. To find such a decompositions can be important for constructing efficient algorithms for difficult problems on graphs with bounded entanglement as discussed in Chapter 2.4 for other complexity measures. Until now we know only one such problem that seems to be difficult: Parity defined in Section 1.1. One step that could lead to this direction was made by Belkhir and Santocanale in [7] where a structural characterisation of undirected graphs with entanglement at most two is given. We generalise this result to arbitrary (i.e., directed) graphs in Section 3.3.

The *Entanglement Game* $\mathrm{EG}(\mathcal{G})$ is played on $\mathcal{G}$ as follows. There are two players: the Cop player who controls $k+1$ cops numbered with $\{1, \dots, k+1\}$, and the Robber player who controls a robber. At the beginning of the play, all cops stay outside the graph and the Robber chooses a vertex to place the robber there in the first move. The players move in turn. The Cop player can take one cop from his vertex or from outside and place him on the vertex where the robber currently is, or he can let his cops idle. The robber must move to a vertex where an edge from her current vertex leads to, no matter whether a cop occupied her current vertex in the previous move or not. The target vertex must be free of cops. If the robber is unable to do that, the play ends and she loses. If the play lasts infinitely long, the Robber player wins. Formally, we have the following definition.

Definition 3.0.16. Let $\mathcal{G} = (V, E)$ be a directed graph. The *Entanglement Game* $\mathrm{EG}(\mathcal{G})$ on $\mathcal{G}$ is a graph searching game on $\mathcal{G}$ defined by the triple $(\mathcal{L}_c, \mathcal{L}_r, \mathcal{A})$ with $\mathcal{A} = (V_A, V_0, V_1, E_A, v_I)$ where

- $\mathcal{L}_c = 2^V$, $\mathcal{L}_r = \{\{r\} \mid r \in V\} \cup \{V\}$,
- $V_0 = \{(\emptyset, V)\} \cup \{(X, \{r\}) \mid r \notin X\}$,
- $V_1 = \{(\emptyset, \emptyset, V)\} \cup \{(X, X', \{r\}) \mid (X, \{r\} \in V_0 \text{ and } X' \subseteq X \cup \{r\})\}$,
- $\big((X, R), (X, X', R)\big) \in E_A$, for all $(X, R) \in V_0$ and all $(X, X', \{r\} \in V_1)$,
- there is an edge from $(\emptyset, \emptyset, V)$ to $(\emptyset, \{r\})$, for all $r \in V$,
- for all $(r, r') \in E$ and $(X, X', \{r\}) \in V_1$ with $r' \notin X'$, there is an edge in E from $(X, X', \{r\})$ to (X', r'), and
- there are no other edges in E.

Definition 3.0.17. Let $\mathcal{G}$ be a directed graph. The *k cops Entanglement Game* $\mathrm{EG_k}(\mathcal{G})$ on $\mathcal{G}$ is the Entanglement Game on $\mathcal{G}$ where the Cop player has only k cops, i.e., for every $X \in \mathcal{L}_c$, we have $|X| \leq k$.

Definition 3.0.18. Let $\mathcal{G}$ be a directed graph. The *entanglement* $\mathrm{ent}(\mathcal{G})$ of $\mathcal{G}$ is the minimal number k such that the Cop player has a winning strategy in the game $\mathrm{EG_k}(\mathcal{G})$.

Definition 3.0.19. A directed graph $\mathcal{T} = (V, E)$ is a *tree with back-edges* if E can be partitioned in *tree-edges* F and *back-edges* B such that (V, F) is an arborescence and whenever $(u, v) \in B$, then there is a path from v to u in (V, F).

Lemma 3.0.20 ([10])**.** *The decomposition of the edge set of a tree with back-edges into tree-edges and back-edges is unique up to the choice of the root.*

Definition 3.0.21. Let $\mathcal{T} = (V, E)$ be a tree with back-edges with a partition of E into tree-edges F and back-edges B. The *feedback* of a vertex v is the number of ancestors of v (including v) that are reachable by a back-edge from a successor of v (including v). The feedback $\mathrm{fb}(\mathcal{T})$ of $\mathcal{T}$ is the minimal feedback of vertices in $\mathcal{T}$, i.e.,

$$\mathrm{fb}(\mathcal{T}) = \max_{v \in V} |\{u \in V \mid \text{ there is } w \in V \text{ such that } u \leq v \leq w \text{ and } (w, u) \in B\}|.$$

The back-edge (u, v) and vertex u are *active* at vertex v, if $u \leq v \leq w$.

Definition 3.0.22. Let $\mathcal{G} = (V, E)$ be a directed graph. A *finite unravelling* of $\mathcal{G}$ is a directed graph $\mathcal{U}_\mathcal{G} = (U, E_U)$ that is constructed from $\mathcal{G}$ in the following way. Choose a vertex $v \in V$ and let v. be in U. Mark edges of E until every edge in E is marked doing the following. For every unmarked edge $(u, w) \in E$, either add w to U and (u, w) to E_U, or (only if $w \in U$) add (u, w) to E_U. The decision which option to choose is made non-deterministically. Then mark (u, v) in $\mathcal{G}$.

Proposition 3.0.23 ([10])**.** *The entanglement of a directed graph is the minimal feedback (and the minimal entanglement) of its finite unravellings, i.e.,*

$$\begin{aligned} \mathrm{ent}(\mathcal{G}) &= \min\{fb(\mathcal{T}) \mid \mathcal{T} \text{ is a finite unravelling of } \mathcal{G}\} \\ &= \min\{fb(\mathcal{T}) \mid \mathcal{T} \text{ is a finite unravelling of } \mathcal{G}\} \end{aligned}$$

We turn to the computational aspects of the entanglement.

Proposition 3.0.24 ([10])**.** *The problem whether a given directed graph has entanglement at most k can be decided in polynomial time, if k is fixed.*

Proposition 3.0.25 ([10])**.** *Parity games on graphs with bounded entanglement can be solved in polynomial time.*

We shall see in Proposition 4.0.8 that if a graph has Kelly-width or DAG-width k then its lexicographic product with the complete graph on n vertices has Kelly-width resp. DAG-width $n \cdot k$. This will play an important role in proving discrepancies between monotone and general Cop's strategies. We show that this property does not hold for entanglement.

Proposition 3.0.26. *Let $\mathcal{K}_n$ be the complete graph with n vertices. There is a class of graphs such that, for any graph $\mathcal{G}$ from this class,* $\mathrm{ent}(\mathcal{G}) = 1$ *and* $\mathrm{ent}(\mathcal{G} \bullet \mathcal{K}_n) = 2n - 1$.

Proof. Consider the class of directed cycles of length at least 3. Every cycle has entanglement one. We claim that, for each cycle $\mathcal{G}$, $\mathrm{ent}(\mathcal{G} \bullet \mathcal{K}_n) = 2n - 1$. A winning strategy of $2n-1$ cops is to occupy a clique of n vertices in $\mathcal{G}$ and use the rest $n-1$ cops to expel the robber from any other clique. Conversely, the robber escapes $2n-2$ cops, because when she is expelled from a maximal clique, there is always a cop free path to another maximal clique with at most $n-1$ cops. □

3.1 Entanglement of special graph classes

Proposition 3.1.1 ([10]). *Let $\mathcal{G}$ be a directed graph.*

1. $\mathrm{ent}(\mathcal{G}) = 0$ *if and only if $\mathcal{G}$ is acyclic.*
2. *If $\mathcal{G}$ is the graph of a unary function, then* $\mathrm{ent}(\mathcal{G}) = 1$.
3. *If $\mathcal{G}$ an undirected tree, then* $\mathrm{ent}(\overleftrightarrow{\mathcal{G}}) \leq 2$.
4. *If $\mathcal{G}$ is the complete directed graph with n vertices, then* $\mathrm{ent}(\mathcal{G}) = n$.

Proposition 3.1.2 ([10]). *Let $\mathcal{G}$ be a directed graph. The entanglement of $\mathcal{G}$ is one if and only if the graph is not acyclic and in every strongly connected component there is a vertex whose removal makes the component acyclic.*

Corollary 3.1.3 ([10]). *For $k = 0$ and $k = 1$, the problem whether a given graph has entanglement k is* NLOGSPACE*-complete.*

Lemma 3.1.4 ([10]). *Let $\mathcal{G} = (V, E)$ be a directed graph such that, for some $k \in \omega$, there exists a partial labelling $i : V \to [k]$ under which every strongly connected subgraph $\mathcal{C} = (V_C, E_C)$ of $\mathcal{G}$ contains a vertex v whose label is unique in C, that is, $i(v) \neq i(w)$ for all $w \in V_C$ with $w \neq v$. Then* $\mathrm{ent}(\mathcal{G}) \leq k$.

Proposition 3.1.5 ([10]). *For every natural n, the undirected $(n \times n)$-grid has entanglement at most $3n$.*

Proof. We show that there exists a labelling of graph vertices satisfying the requirements of Lemma 3.1.4. Assign the values $0, \dots, n$ to the horizontal median of the grid, i.e.,

$i(\lfloor \frac{n}{2} \rfloor, j) = j$ for all $j \in [n]$. For the two $\frac{n}{2}\times n$-grids obtained when removing the positions already labelled, we proceed independently and assign the values $n, \ldots, n + \frac{n}{2}$ to their vertical medians, and so on, in step k applying the procedure to the yet unlabelled domain consisting of 2^k many $\frac{n}{2^k} \times \frac{n}{2^k}$ disconnected grids. □

We state the following simple lemma.

Lemma 3.1.6. *Let $\mathcal{G}$ be a graph. Let $m \geq 1$ and $k \geq 0$ be natural numbers. If there exist m vertices $v_0, \ldots, v_{m-1}$ in $\mathcal{G}$ such that k cops have a winning strategy on $\mathcal{G}\backslash\{v_0, \ldots, v_{m-1}\}$ then* $\mathrm{ent}(\mathcal{G}) \leq m + k$.

Proof. The Cop player has the following winning strategy. Play on $\mathcal{G}\backslash\{v_0, \ldots, v_{m-1}\}$ the strategy for k cops guaranteed by conditions of the lemma. When the robber enters a vertex in $\{v_0, \ldots, v_{m-1}\}$ place there one of m cops. Continue in this way until all vertices in $\{v_0, \ldots, v_{m-1}\}$ are occupied by the cops. Then win on $\mathcal{G}\backslash\{v_0, \ldots, v_{m-1}\}$ with k □

Here a straightforward way to win is to place m cops on $\{v_0, \ldots, v_{m-1}\}$ and then use k cops for the rest graph. A reuse of some of the m cops may be an obvious improvement of the Cop's strategy. In Section 3.2 we shall strengthen this idea to get a characterisation of graphs with entanglement two. For the next proposition, recall the definition of a torus from Section 1.6. We characterise the entanglement of such graphs.

Proposition 3.1.7 ([10])**.** *Let $m, n \geq 1$ be natural numbers. Let $\mathcal{T}_{mn}$ be a torus.*

1. *If $m = n$ then* $\mathrm{ent}(\mathcal{T}_{mn}) = n$.
2. *If $m < n$ then* $\mathrm{ent}(\mathcal{T}_{mn}) = m + 1$.

Proof. For $m = n$, a winning strategy of the Cop player is to place cops on a diagonal of the grid. The graph with deleted diagonal becomes acyclic, so 0 cops win on it. From Lemma 3.1.6 for $k = 0$ follows that $\mathrm{ent}(\mathcal{T}_{mn}) \leq m$. The converse is shown similarly to Examples 1.6.12 and 1.7.9. The Robber can hold the invariant that there is a cop free column in the torus and a cop free path there from the robber's vertex. At the beginning of a play it is clear. In general, the robber moves on such a column until a cop announces to land on her vertex. In that moment, there is another cop free column (we have m

columns and $m-1$ cops, but a cop is flying to the robber's vertex). Then there is a cop free row in the torus. The robber runs to the crossing of that row and the column she is on and then along the row to the column that became free.

If $m < n$ then m cops occupy diagonal of a $(m \times m)$-subtorus. The rest graph consists of $n-m$ strongly connected components. These are rows of $\mathcal{T}_{mn}$). Every row is a directed cycle and one cop suffices to expel the robber from it. According to Proposition 3.1.2, the entanglement of the rest graph is one. From Lemma 3.1.6 follows that $\operatorname{ent}(\mathcal{T}_{mn}) \leq m+1$. The winning strategy of the robber against m cops is the same as on $\mathcal{T}_{mm}$) against $m-1$ cops. □

Example 3.1.8. Consider undirected $(m \times n)$-grid $\mathcal{G}_{mn}$ where m is much less than n. We want to give an upper bound for the entanglement of $\mathcal{G}_{mn}$ that depends only on m. We start with small values of m and n.

Proposition 3.1.9.

1. $\operatorname{ent}(\mathcal{G}_{2,n}) = 1$ *if and only if* $n = 1$.
2. $\operatorname{ent}(\mathcal{G}_{2,n}) = 2$ *if and only if* $n = 2$.
3. $\operatorname{ent}(\mathcal{G}_{2,n}) = 3$ *if and only if* $3 \leq n \leq 9$.
4. $\operatorname{ent}(\mathcal{G}_{2,n}) = 4$ *if and only if* $n \geq 10$.

Proof. We show that $\operatorname{ent}(\mathcal{G}_{2,n}) \leq 4$ for all n and that $\operatorname{ent}(\mathcal{G}_{2,n}) \leq 3$ for $n = 8$. For $n = 9$ the strategy is similar. For simplicity of description, we imagine natural coordinates on the grid as shown in Figure 3.3. The vertices have coordinates (i, j) with $i \in \{0, 1\}$ and $j \in \{0, \ldots, n-1\}$.

Four cops have the following winning strategy. Two of them place themselves on $(0, \lfloor \frac{n}{2} \rfloor)$ and $(1, \lfloor \frac{n}{2} \rfloor)$. We call the other cops the *chasers*. Due to symmetry, assume without lost of generality that the robber hides in the right part of the grid (with larger second coordinate). The goal of the cops is to shift to the right the wall of two cops on $(0, \lfloor \frac{n}{2} \rfloor)$ and $(1, \lfloor \frac{n}{2} \rfloor)$ such that the robber remains on the right of the wall. The new wall can be built also by the chasers. The chasers remain free from guarding the left part of the grid. One of them follows the robber until she moves vertically, i.e., from (i, j) to

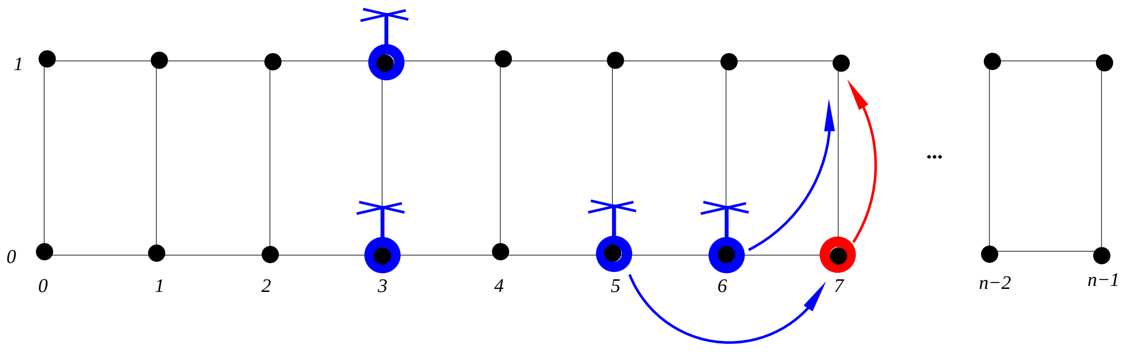

Figure 3.1: The first part of Cop's player strategy to move the wall on $\mathcal{G}_{2,n}$. The cops in helicopters are blue, the robber is red. Cops on $(0,3)$ and $(1,3)$ build the wall. The chaser from $(0,5)$ goes to $(0,7)$. Then the robber goes to $(1,7)$. The cop from $(0,6)$ follows her there.

$(1-i,j)$ for some $i \in \{0,1\}$ and $j \in \{\lfloor \frac{n}{2} \rfloor, \dots, n-1\}$. In this moment the second chaser goes to $(1-i,j)$. See Figure 3.1. If the robber moves to the right, the wall is moved (the chasers build it and the cops from the old wall become chasers). In the other case, the chaser from (i,j) follows the robber. Both chasers continue to follow the robber in the leap-frogging manner to the left until she moves vertically. (Figure 3.2, steps 1 and 2.) The rightmost chaser follows her (step 3). Again, if she goes to the right, the wall is moved. She can go further to the left (step 4) followed by the cop (step 5), but this process can continue at most until the robber meets the wall. So finally she must move vertically (step 6), is followed by rightmost cop (step 7) and must go to the right (step 8). So the wall moves to the right and the robber remains to the right of the wall. This suffices to capture her.

For $n = 8$, consider the (2×8)-grid (Figure 3.3). We show how tree cops capture the robber. The idea is to place a cop on vertex $(0,3)$ and a cop on $1,5$ such that the robber to the right of column 3. We show first that it suffices for the Cop player to win. He expels the robber from the area right from the cop on $(1,5)$. For this, the third cop places himself on $(0,6)$. If the robber remains on $(0,7)$, $(1,6)$ or $(1,7)$ the she is captured wit help of the cop from $(0,3)$. So assume that she goes to $(0,5)$ and then to $(0,4)$ (she must move every time it is her turn). The third cop follows her to $(0,4)$.

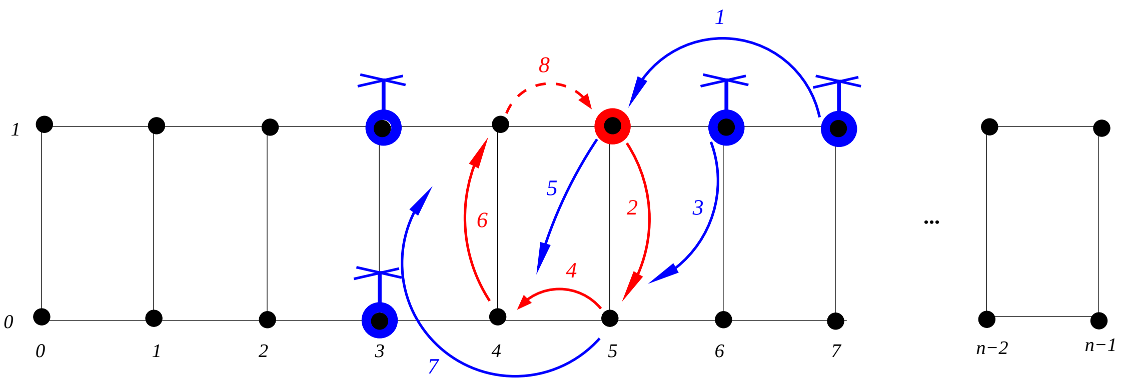

Figure 3.2: The second part of Cop's player strategy to move the wall on $\mathcal{G}_{2,n}$. Cops on $(0,3)$ and $(1,3)$ build the wall. The moves of cops (in blue) and those of the robber (in red) are labelled to indicate the order of the moves.

Again, if she goes to the right, i.e., to $(0,5)$, the cop from $(0,3)$ follows her, she must go to $(0,6)$, the cop from $(0,4)$ goes there and she is captured in the corner. So when the cops are on $(0,3),(0,4),(1,5)$ the robber goes from $(0,4)$ to $(1,4)$. The cop from $(0,4)$ goes there. The robber has two possibilities: to $(1,3)$ and then to $(1,2)$, or to $(0,4)$ and then to $(0,5)$. In the first case she is followed to $(1,2)$ by the cop from $(1,5)$ and she is captured in the left corner. In the second case she is followed to $(0,5)$ by the cop from $(1,5)$. Then the cop from $(0,3)$ forces her to go to $(1,6)$ and she is captured.

It remains to show how the cops force a position when two of them stay on $(0,3)$ and $(1,5)$ and the robber is to the right of column 3. It is clear that the Cop player can place two of his cops on those vertices. Assume that the robber is on a column with number less than 4. The third cop and the cop $(1,5)$ expel her from there placing themselves on $(1,1)$ and $(0,2)$. (One of them must return to $(1,5)$. In that moment the robber will be to the right of column 3.) She is either captured in the left corner or goes to $(1,2)$ and then to $(1,3)$. One of the cops not from $(0,3)$follows her. The robber goes to $(1,4)$, the other cop follows. She tries to avoid $(0,5)$ and goes to $(0,4)$ followed from the cop from $(1,3)$ and goes to $(0,5)$. The cop from $(0,4)$ follows her. Now the robber either goes $(0,4)$ where she is immediately captured or is blocked in the right corner. □

For $m=3$, we do not give entanglements for all n, but only for $n=1,2,\dots,6$ and

and estimate the entanglement for sufficiently large n in the general case.

Proposition 3.1.10.

1. $\operatorname{ent}(\mathcal{G}_{3,n}) = 1$ *if and only if* $n = 1$.

2. $\operatorname{ent}(\mathcal{G}_{3,n}) = 3$ *if and only if* $n = 2$.

3. $\operatorname{ent}(\mathcal{G}_{3,n}) = 4$ *if* $3 \leq n \leq 6$.

Proposition 3.1.11. $\operatorname{ent}(\mathcal{G}_{3,n}) \leq 12$.

Proof. We describe a winning strategy of the Cop player. We show that the cops are able to occupy a column with 3 cops using only 6 of cops. If they have a way to do this, they construct thus a *barrier* separating the grid. They can then use the same strategy to construct another barrier on the side where the robber escaped, say on the right. If she escapes to the right again the cops from the first barrier can be reused. If the robber moves in the area between the barriers the remaining 6 cops construct a third barrier thus releasing 3 cops from themselves and 3 cops from the first or the second barrier. Continuing in this way they shrink the free space for robber and finally capture her.

We now describe how 6 cops manage to place 3 of them on a column. The first goal is to set two cops directly one under another without using any other cops, i.e., to construct a *partial barrier*. One cop goes to the vertex where the robber is. If she goes up or down, another cop takes this place and a partial barrier is constructed. Otherwise the robber moves horizontally and is followed by both cops in a leap-frogging manner. The robber can now run some time horizontally. Finally she moves up or down and the cop whose turn to move it is completes the construction of the partial barrier.

The goal of the cops is now to force the robber to occupy the remaining free vertex under or above the barrier. They can use 4 remaining cops for that. Two other cops construct a new partial barrier and then the remaining two a third one. The two cops from a partial barrier that is the furthest from her construct a new barrier next to the robber and so on until the robber's free place is shrunken and she is forced to visit a vertex that is in the same column as a partial barrier. □

The last strategy can be easily generalised for arbitrary m. The cops build two complete barriers (of height m) with the robber between them, two partial barriers of height

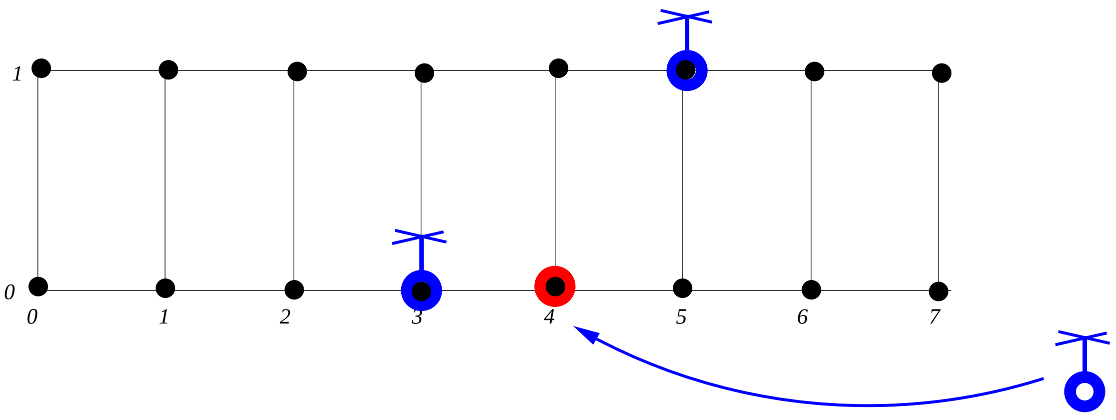

Figure 3.3: The 2×8-grid. The cops (in helicopters) stay on $(0,3)$ and $(1,5)$ and one cop is outside. He announces to go to vertex $(0,4)$ where the robber stays.

$m-1$ (with the robber between them) and so on until height 1. The total number of cops is thus $2 \cdot \sum_{i=1}^{m} i = m \cdot (m+1)$.

Proposition 3.1.12. $\text{ent}(\mathcal{G}_{m,n}) \leq m \cdot (m+1)$.

As a last graph class we discuss tournaments — graphs that are explicitly directed.

Definition 3.1.13. A *tournament on n vertices* is a directed graph $\mathcal{T}_n = (V,E)$ such that $|V| = n$ and E satisfy the condition: for every pair $u, v \in V$, either $(u,v) \in E$ or $(v,u) \in E$.

We consider classes $\mathbf{T_n}$ of tournaments $\mathcal{T}$ on n vertices. First we define entanglement of such a class. Note that, for every n, there is a tournament $\mathcal{T}$ with $\text{ent}(\mathcal{T}) = 0$. One needs to enumerate the vertices of $\mathcal{T}$ and draw an edge from smaller vertices to larger ones. The resulting tournament is acyclic and has entanglement 0.

Definition 3.1.14. Let $\mathbf{T_n}$ be the class of tournaments on n vertices. The *entanglement* $\text{ent}(\mathbf{T_n})$ of $\mathbf{T_n}$ is the maximal entanglement over all tournaments in $\mathbf{T_n}$, i.e., $\text{ent}(\mathbf{T_n}) = \max_{\mathcal{T} \in \mathbf{T_n}}(\text{ent}(\mathcal{T}))$.

By inspection we determine the entanglements of small tournament classes. See Figure 3.4 for examples of tournaments on 5, 6 and 7 vertices with entanglement respectively 2, 3 and 4.

- $\text{ent}(\mathbf{T_1}) = \text{ent}(\mathbf{T_2}) = 0$.
- $\text{ent}(\mathbf{T_4}) = \text{ent}(\mathbf{T_3}) = 1$.
- $\text{ent}(\mathbf{T_5}) = 2$.
- $\text{ent}(\mathbf{T_6}) = 3$.
- $\text{ent}(\mathbf{T_7}) = 4$.

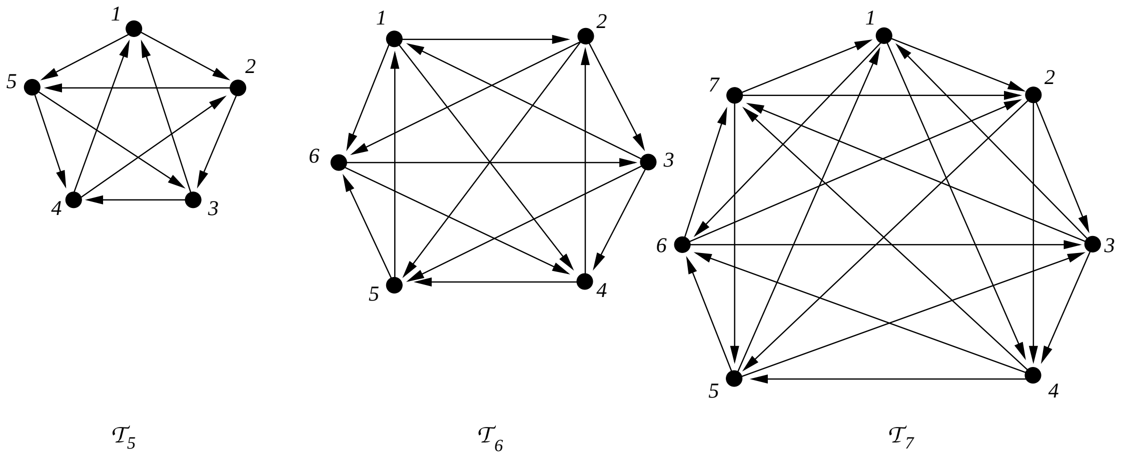

Figure 3.4: Examples of tournaments on 5, 6 and 7 vertices with entanglement respectively 2, 3 and 4. For every vertex (two, three vertices) there exists a cycle reachable from the vertex (both, all vertices) such that the vertex (two, three vertices) is (are) not in the cycle.

It seems that the entanglement of a class grows with every n by one, beginning by $n = 4$, but this is not the case. It is clear that it does not grow by 2 or more: if, for any n, $\text{ent}(\mathbf{T_n}) = k$ then $\text{ent}(\mathbf{T_{n+1}}) \leq k + 1$, because the additional cop can occupy the new vertex and the graph without this vertex is in $\mathbf{T_n}$, where k cops suffice to win. We show in that the difference between n and $\text{ent}(\mathbf{T_n})$ cannot be bounded from above.

Proposition 3.1.15. *For every natural $n \geq 2$, we have that* $\text{ent}(\mathbf{T_{2^n}}) \leq 2^n - n - 1$.

Proof. We make an induction on n. For $n = 2$, $\text{ent}(\mathbf{T_4}) = 1$. Assume that $\text{ent}(\mathbf{T_{2^n}}) \leq 2^n - n - 1$. We show that $\text{ent}(\mathbf{T_{2^{n+1}}}) \leq 2^{n+1} - n - 2$. Consider a tournament $\mathcal{T}_{2n+1} \in \mathbf{T_{2^{n+1}}}$ with $\mathcal{T}_{2^{n+1}} = (V, E)$ and $|V| = 2^{n+1}$. Then $\mathcal{T}_{2^{n+1}}$ has exactly $2^n \cdot (2^{n+1} - 1)$ edges and at least one vertex v_0 with in-degree $d^{in}(v_0) \geq \lceil \frac{2^n \cdot (2^{n+1}-1)}{2^{n+1}} \rceil = 2^n$. Let $v_1, \ldots, v_{2^n}$ be 2^n distinct vertices with $(v_i, v_0) \in E$ for each $i \in \{1, \ldots, 2^n\}$. There are $2^{n+1} - (2^n + 1)$ vertices different from any v_i ($i \in [2^n + 1]$). We place a cop on each of them. On $\mathcal{T}_{2^{n+1}}[v_0, \ldots, v_{2^n}]$, $2^n - n - 1$ cops have a winning strategy. Note that if the Robber visits v_0 she looses immediately, no matter whether there are any cops on $\mathcal{T}_{2^{n+1}}[v_0, \ldots, v_{2^n}]$ because there is no edge from v_0 to any v_i for $i \in \{1, dots, 2^m\}$. So we can apply the induction hypothesis ans state that $2^n - n - 1$ cops indeed win on $\mathcal{T}_{2^{n+1}}[v_0, \ldots, v_{2^n}]$. Summing up, we get that we used $\big(2^{n+1} - (2^n + 1)\big) + \big(2^n - n - 1\big) = 2^{n+1} - n - 2$ cops. □

On the other hand, the entanglement of a tournament class is never too small.

Proposition 3.1.16. *For every natural* $n \geq 2$, *if* $\text{ent}(\mathbf{T_n} = k)$ *then* $\text{ent}(\mathbf{T_{n+2}} \geq k + 1)$.

Proof. Let $\mathcal{T}_n = (V, E) \in \mathbf{T_n}$ be a tournament of entanglement k. We construct a tournament $\mathcal{T}_{n+1}$ on $n + 1$ vertices with $\mathcal{T}_{n+1} \geq k + 1$. Let $\mathcal{T}_{n+1}$ be the tournament $\mathcal{T}_n$ with two additional vertices u and v and additional edges (u, w) and w, v for all $w \in V$ and the edge (v, u). We claim that $\mathcal{T}_{n+1}$ has entanglement at least $k + 1$. We show how the robber escapes k cops. She can escape from $k - 1$ cops remaining in $\mathcal{T}_{n+1} \backslash \{u, v\}$, because $\text{ent}(\mathcal{T}_n = k)$. By Proposition 3.1.15, if at most $k - 1$ cops are in $\mathcal{T}_{n+1} \backslash \{u, v\}$ there is a cop free vertex w in $\mathcal{T}_{n+1} \backslash \{u, v\}$. When the k-th cop comes into the play, she goes to the cycle u, w, v and remains there until a cop occupies u or v. Then the robber returns to the strategy on $\mathcal{T}_{n+1} \backslash \{u, v\}$ using that from the cycle u, w, v there is a cop free path to any vertex of $\mathcal{T}_{n+1} \backslash \{u, v\}$. If a cop goes to w instead of to u or v, there is another free vertex w' that is used by the robber instead of w. □

In Section 3.2 we give a characterisation of graphs of entanglement two. The idea is based on a characterisation given by Belkhir and Santocanale in [7]. They show that undirected graphs of entanglement two are essentially undirected forests to those trees the following two types of operations have been applied an arbitrary number of times.

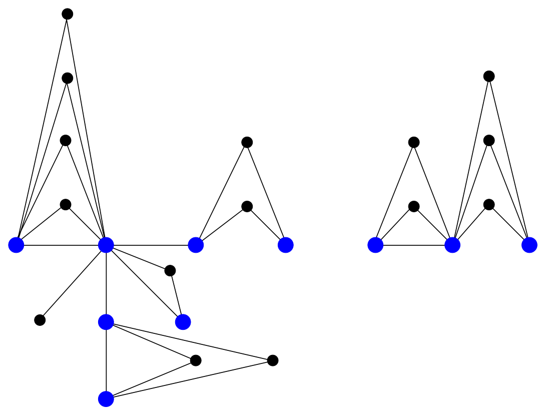

Figure 3.5: A typical graph of entanglement two. Bigger blue points denote vertices of the underlying tree.

1. Let a and b be tree vertices with an edge $\{a, b\}$ between them. Add $m \geq 1$ vertices $v_1, \dots, v_m$ to the graph and edges $\{a, v_i\}$ and $\{b, v_i\}$, for all $i \in \{1, \dots, m\}$. Delete the edge $\{a, b\}$.

2. The second operation is the same as the first, but $\{a, b\}$ is not deleted.

A typical graph of entanglement two is shown in Figure 3.5. The strategy of the Cop player is to go to the vertex where the robber is, then with another cop to the tree vertex where she goes (if she goes to an added vertex, wait). In this way, the cops move in turn and one cop always guard the way from the robber to the other cop.

We now give formal definitions and state the characterisation.

Definition 3.1.17 ([7]). A molecule $M^{\varepsilon,n}_{a,b}$, where $\varepsilon \in \{0, 1\}$ and $n \geq 0$, is the undirected graph (V, E) with $V = \{a, b, c_1 \dots, c_n\}$ and $E = E_1 \cup E_2$, where $E_1 = \{\{a, c_i\} \mid 1 \leq i \leq n\} \cup \{\{b, c_i\} \mid 1 \leq i \leq n\}$ and $E_2 = \{a, b\}$, if $\varepsilon = 1$, and $E_2 = \emptyset$, if $\varepsilon = 0$. The *glue points* of $M^{\varepsilon,n}_{a,b}$ are a and b. Its *dead points* are $\{a, b, c_1 \dots, c_n\}$.

Definition 3.1.18 ([7]). Let $\mathcal{G}_1 = (V_1, E_1)$ and $\mathcal{G}_2 = (V_2, E_2)$ be two undirected graphs with $V_1 \cap V_2 = \emptyset$, let $a \in V_1$ and $b \in V_2$. The *collapse* of $\mathcal{G}_1$ and $\mathcal{G}_2$ on vertices a and b, denoted $\mathcal{G}_1 \bigoplus_{a,b}^{z} \mathcal{G}_2$, is the graph $\mathcal{G} = (V, E)$ defined as follows:

$$\begin{aligned} V &= (V_1 \backslash \{a_1\}) \cup (V_1 \backslash \{a_1\}) \cup \{z\}, \text{ where } z \notin V_1 \cup V_2, \\ E &= \{\{x_1, y_1\} \in E_1 \mid a_1 \notin \{x_1, y_1\}\} \cup \{\{x_2, y_2\} \in E_2 \mid a_2 \notin \{x_2, y_2\}\} \\ &\quad \cup \{\{x, z\} \mid \{x, a_1\} \in E_1 \text{ or } \{x, a_2\} \in E_2\}. \end{aligned}$$

Recall the definition of a graph with final vertices from Section 1.3.12[1].

Definition 3.1.19. If $\mathcal{G}_1 = (V_1, E_1, G_1)$ and $\mathcal{G}_2 = (V_2, E_2, G_2)$ are graphs with final vertices then we say that $\mathcal{G}_1 \bigoplus_{a,b}^{z} \mathcal{G}_2$ is a *legal collapse* if $a \in G_1$ and $b \in G_2$. We shall use the notation $\mathcal{G}_1 \overline{\bigoplus}_{a,b}^{z} \mathcal{G}_2$ and define the glue points of the legal collapse to be $(G_1 \backslash \{a\}) \cup (G_2 \backslash \{b\}) \cup \{z\}$, so that $\mathcal{G}_1 \overline{\bigoplus}_{a,b}^{z} \mathcal{G}_2$ is a graph with final vertices.

We denote the singleton graph whose own vertex is a glue point by

Definition 3.1.20 ([7]). Let $\mathcal{E}_2$ be be the least class of graphs with final vertices containing the molecules, the singleton graph with the own vertex as a glue point, and closed under legal collapses and graph isomorphisms.

Before we state that $\mathcal{E}_2$ is the class of undirected graphs of entanglement two, we give some definitions (motivated in [7]) that help us to formulate another characterisation of this class.

Definition 3.1.21. An undirected graph $\mathcal{G}$ is *simple* if the following conditions are satisfied:

1. Every cycle in $\mathcal{G}$ has length is at most four.
2. Every cycle of length tree has at least one vertex with degree two in $\mathcal{G}$.
3. Every cycle of length four does not have two adjacent vertices of degree strictly greater than two.

Proposition 3.1.22 ([7]). *For an undirected graph $\mathcal{G}$ the following are equivalent:*

1. $\mathcal{G}$ has entanglement at most two.

[1]Belkhir and Santocanale call it a *glue graph.*

2. $\mathcal{G}$ is simple.

3. $\mathcal{G}$ belongs to class $\mathcal{E}_2$[2].

3.2 Modular entanglement

We study entanglement on strongly connected subgraphs of a graph. To express connections between them consider the notion of a graph with final vertices in Definition 1.3.12. We define a variant of the Entanglement game played on it.

Definition 3.2.1. The *Entanglement game with final vertices* $EG^k(\mathcal{G})$ is played on $\mathcal{G}$ as follows. There are two players: the Cop player who controls $k+1$ cops numbered with $\{1, \dots, k+1\}$, and the Robber player who controls a robber[3]. At the beginning of play, all cops stay outside the graph and the Robber chooses a vertex to place the robber there in the first move. The players move in turn. The Cop player can take one cop from his vertex or from outside and place him on the vertex where the robber currently is, or he can let his cops idle. The robber must move to a vertex where an edge from her current vertex leads to, no matter whether a cop occupied her current vertex in the previous move or not. The target vertex must be free of cops. If the robber is unable to do that, the play ends and she loses. If the play lasts infinitely long, the Robber player wins. Moreover, to use the final vertices, we add a new winning condition for the Robber: if the robber reaches a vertex v with $v \in F$ after the Cop player has moved the $k+1$-st cop (i.e., has brought him from outside into the graph), then the Robber player wins as well.

A graph with final vertices $\mathcal{G} = (V, E, F)$ is *k-complex*, if the robber has a winning strategy in the Entanglement game with final vertices $EG^k(\mathcal{G})$.Otherwise the graph is k-simple.

The next lemma states that the Cop player in Entanglement game with final vertices gets no advantage if we change the definition of the game in that not the Robber, but

[2]To be precise, the class $\mathcal{E}_2$ consists of graphs with final vertices. We mean that we forget the glue points, i.e., if $\mathcal{G} = (V, E, G)$ we make it to (V, E).

[3]Informally, for convenience, we shall sometimes confuse the Robber player with the robber in the play and the Cop player with his cops.

the Cop player is allowed to choose the vertex for the robber to start on.

Lemma 3.2.2. *Let $\mathcal{G} = (V, E, F)$ be a strongly connected k-complex graph with final vertices. The Robber player still wins the Entanglement game with final vertices on $\mathcal{G}$ if at the beginning of a play it is the Cop player who chooses the vertex from which the robber has to start.*

Proof. Assume that there are vertices v and w such that

1. v is a vertex from which the robber wins if she can use the additional winning condition from Definition 3.2.1, and

2. w is a vertex such that the Cop player wins the game if the robber starts from it. As $\mathcal{G}$ is strongly connected, we can take vertices such that $(w, v) \in E$.

Let σ_R^v be a winning strategy for the robber starting in v and let σ_C^w be a winning strategy for the Cop player for the case that the robber starts in w. If, in any play consistent with σ_R^v, the robber never enters w until the last $(k+1)$-st cop moves, then σ_R^v is a counter-strategy for σ_C^w: in the first step, the robber goes from w to v and plays according to σ_R^v and wins, no matter whether a cop placed himself on w in a first Cop's move. The only difference to the case when the robber starts from v is that a cop is not outside the graph, but on w[4].

If there is a strategy σ_C^v for k cops, such that in the play consistent with both σ_R^v and σ_C^v the robber visits w, then there is a counter-strategy for σ_R^v: the cops play according to σ_C^v until the robber enters w and then place a cop on w and use σ_C^w to win. $\square$

A direct corollary (if we take $F = \emptyset$) is the next proposition.

Proposition 3.2.3. *In the Entanglement game on a strongly connected graph, the number of cops needed to capture the robber does not change, if it is the Cop player who chooses the vertex from which the robber starts.*

[4] In a more general definition of modular entanglement, when more than one cop must stay outside before additional vertices become winning for the Robber, the proof does not work: a cop on w may block exit points that are guarded by several cops. So a cop on w would make several cops free.

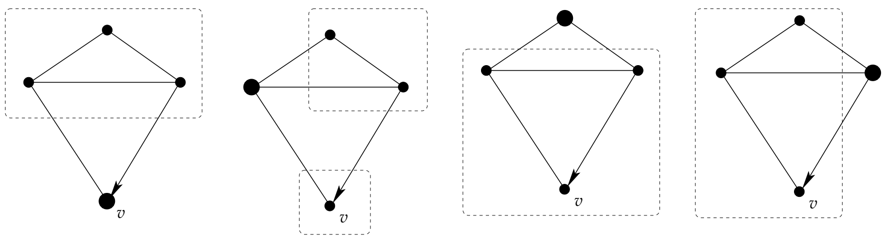

Figure 3.6: A graph and its components. Components of vertices marked with large circles are shown in dashed rectangular boxes.

To simplify the notation, for a graph with final vertices $\mathcal{G} = (V, E, F)$ or a graph $\mathcal{G} = (V, E)$ and a vertex $a \in V$, we shall write $\mathcal{G}\backslash a$ instead of $\mathcal{G}[V\backslash\{a\}]$ for the structure induced by all vertices but a. Consider the strongly connected component decomposition of $\mathcal{G}\backslash a$.

Let $\mathcal{G} = (V, E)$ be a graph and let $\mathcal{G}' = (V', E')$ be an induced subgraph of $\mathcal{G}$. We call a vertex $v \in V'$ an *exit point* if there is a vertex $u \in V\backslash V'$ in $\mathcal{G}$ with $(v, u) \in E$.

Note that strategies of both players in the Entanglement games with final vertices are strategies in the Entanglement Games.

Definition 3.2.4. Let $\mathcal{G} = (V, E)$ be a graph, let $\mathcal{C} = (C, E', F)$ be a graph with final vertices, and v a vertex in $V\backslash C$. We say that $\mathcal{C}$ is a *v-component* of $\mathcal{G}$ if (C, E') is a strongly connected component of $\mathcal{G}\backslash v$ and F is the set of exit points of (C, E'). If, for a number k, $\mathcal{C}$ is k-complex, we call strategies of the Robber player on $\mathcal{C}$ that are winning in the Entanglement game with final vertices *$\mathcal{C}$-strategies*.

Example 3.2.5. Consider the graph in Figure 3.6. Every vertex w induces a w-component, which contains all other vertices except the leftmost one with a component induced by v and a component induced by the other two vertices. The v-component $\mathcal{C}$ is 1-complex, because the robber can escape from one cop in $\mathcal{C}$ and there is always a cop free path to an exit point of the component. In fact, every strategy of the Robber player on $\mathcal{C}$ is a $\mathcal{C}$-strategy. All other vertices induce 1-simple components.

Proposition 3.2.6. *If there is a vertex v in a graph $\mathcal{G}$ such that all v-components are k-simple, then* $\operatorname{ent}(G) \leq k + 1$.

Proof. Let v be a vertex such that all v-components are k-simple. The Cop player has the following winning strategy. A cop places himself on v. (If the robber never visits v, the rest of the strategy does not change.) Wherever the robber enters a v-component (C, E') with exits points F, no matter, at which point, the rest of the cops play according to the (C, E', F)-strategy to expel her from there or block all exit points of it and win (if necessary with help of the cop from v). $\square$

Our goal is to show the converse for the case $k = 1$. For this, we first prove some lemmas.

In a strongly connected graph $\mathcal{G}$, for a vertex v, the partial order $\leq_v$ on strongly connected components of $\mathcal{G}\backslash v$ is the topological order

$$\{(\mathcal{C}, \mathcal{C}') \mid \text{ there is a path from } \mathcal{C} \text{ to } \mathcal{C}' \text{ in } \mathcal{G}\backslash v\}.$$

Lemma 3.2.7. *Let $\mathcal{G} = (V, E)$ be a graph and let v be a vertex in V. Further, let $\mathcal{C}_0 = (V_0, E_0, F_0)$ and $\mathcal{C}_1 = (V_1, E_1, F_1)$ be two k-complex v-components. If $\mathcal{C}_0$ and $\mathcal{C}_1$ are incomparable with respect to $\leq_v$, then* $\operatorname{ent}(\mathcal{G}) > k + 1$.

Proof. Without loss of generality assume that $\mathcal{C}_0$ and $\mathcal{C}_1$ have entanglement at most $k + 1$. We describe a winning strategy for the robber against $k + 1$ cops. She starts on some vertex in $\mathcal{C}_0$ and plays a $\mathcal{C}_0$-strategy waiting there for all cops to come. When the last cop moves to $\mathcal{C}_0$, the robber can reach an exit point, because the component is $(k+1)$-complex. From there, she runs to v and then to $\mathcal{C}_1$ (she can do this, because the components are incomparable). Now she plays the same strategy with $\mathcal{C}_1$ instead of $\mathcal{C}_0$. It follows that $k + 1$ cops never capture her. $\square$

Lemma 3.2.8. *Let $\mathcal{C}_v = (C_v, E_v, F_v)$ be a v-component, and $\mathcal{C}_w = (C_w, E_w, F_w)$ be a w-component of a strongly connected graph $\mathcal{G} = (V, E)$, for distinct vertices v and w. Let C_v and C_w be intersecting and let C_w be not a proper subset of C_v, i.e., $C_v \cap C_w \neq \emptyset$ and $C_w \not\subseteq C_v$. If v is in C_w, then w is in C_v.*

Proof. Assume that the conditions of the lemma hold, but w is not in C_v (Figure 3.7). Let u be in $C_v \cap C_w$ and u' in $C_v \backslash C_w$. Because u' and u are in C_v, there are paths from

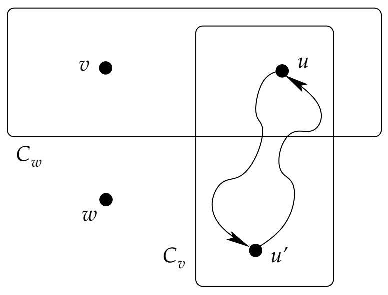

Figure 3.7: The w-component $\mathcal{C}_w$ contains v, but the v-component $\mathcal{C}_v$ does not include w.

u' to u and vice versa that do not include v. None of the paths includes w (because otherwise w would be in C_v), so u' and u lie in the same w-component, but we assumed that u' is not in C_w and u is in C_w. □

Lemma 3.2.9. *Let $\mathcal{G} = (V, E)$ be a strongly connected graph, and let a_0 and a_1 be vertices in V such that, for $i \in \{0, 1\}$, a_i is in a k-complex a_{1-i}-component $\mathcal{C}_{1-i} = (C_{1-i}, E_{1-i}, F_{1-i})$. If C_0 and C_1 are disjoint, then* $\text{ent}(G) > k + 1$.

Proof. Without loss of generality, assume that $\mathcal{C}_0$ and $\mathcal{C}_1$ have entanglement at most $k + 1$, so $\mathcal{C}_0$-strategy and $\mathcal{C}_1$-strategy prescribe the robber to go to an exit point when all $k + 1$ cops arrive in the component.

The following strategy for the Robber player against k cops is winning. The idea is to change between $\mathcal{C}_0$ and $\mathcal{C}_1$. The robber starts on a vertex in the component $\mathcal{C}_0$, in which there is no cop at this time, and plays a $\mathcal{C}_0$-strategy waiting for all cops to move there. When the last cop arrives, she runs to an exit point and then to a_0 and thus enters $\mathcal{C}_1$ (on the way to a_0). No matter which vertex in $\mathcal{C}_1$ she enters first, she has a $\mathcal{C}_1$-strategy, according to Lemma 3.2.2. Note that all cops are outside $\mathcal{C}_1$ at this moment, as $C_0 \cap C_1 = \emptyset$. The robber continues in the same way with the switched roles of $\mathcal{C}_0$ and $\mathcal{C}_1$, and a_0 and a_1, and so on.

□

Lemma 3.2.10. *Let $\mathcal{G} = (V, E)$ be a strongly connected graph. For $i \in \{0, 1\}$, let $\mathcal{C}_i = (V_i, E_i, F_i)$ be two k-complex a_i-components. Let $\mathcal{C}_0$ be maximal with respect to $\leq_{a_0}$*

and let a_1 be in C_0. If C_0 and C_1 are disjoint, then $\text{ent}(G) > k + 1$.

Proof. It suffices to prove that then a_0 is in C_1. Lemma 3.2.9 implies then the desired result. Assume, that $a_0 \notin C_1$. There are tree cases how C_1 can be combined with k-complex a_0-components.

Case 1. The component C_1 is a (not necessarily proper) subset of another k-complex a_0-component $\mathcal{C}' = (C', E', F')$. If the components $\mathcal{C}'$ and $\mathcal{C}_0$ are incomparable then Lemma 3.2.7 guarantees a winning strategy for the robber in the Entanglement game on $\mathcal{G}$ against $k + 1$ cops. So we have $\mathcal{C}' \leq_{a_0} \mathcal{C}_0$ (because $\mathcal{C}_0$ is maximal) and there is a path $\mathcal{P}_1$ from C_1 to C_0 that does not contain a_0.

There is a path $\mathcal{P}_2$ from a_0 to C_1, since $\mathcal{G}$ is strongly connected, but no such path includes vertices of C_0. Otherwise C_0 and C' would be in the same strongly connected component of $\mathcal{G}\backslash a_0$. Further, every path $\mathcal{P}_3$ from C' to a_0 (there is at least one) goes through a_1 (otherwise a_0 is in C_1), see Figure 3.8.

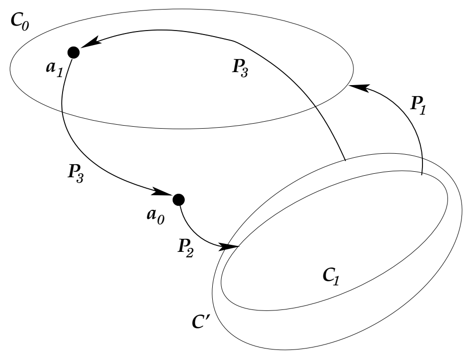

Figure 3.8: Case 1: $\mathcal{C}_1$ is in a a_0-component $\mathcal{C}'$.

This guarantees that the robber wins the Entanglement game on $\mathcal{G}$ switching between $\mathcal{C}'$ and $\mathcal{C}_0$, because playing according to a $\mathcal{C}_0$-strategy and being expelled from $\mathcal{C}_0$ by $k + 1$ cops she can reach a_0 and then $\mathcal{C}_1$. Playing according to a $\mathcal{C}_0$-strategy and being expelled from $\mathcal{C}_1$ she can reach $\mathcal{C}_0$. Lemma 3.2.2 assures that it makes no difference at which vertex the robber enters an strongly connected component.

Case 2. The component C_1 includes vertices of two different strongly connected components of $\mathcal{G}\backslash a_0$. Then there is a path in $\mathcal{C}_1$ from one such strongly connected component to the other that does not go through a_1, but through a_0: if all such paths avoid a_0, the two strongly connected components are not distinct. But then we have $a_0 \in C_1$.

Case 3. In this last case, C_1 does not include vertices of different a_0-components and is not a subset of a k-complex a_0-component. Due to our assumption, $a_0 \notin C_1$, so C_1 consists of some vertices from an a_0-component $\mathcal{C}'$ and some vertices that are in no strongly connected component of $\mathcal{G}\backslash a_0$. The latter are also a part of $\mathcal{C}'$, because all vertices of $\mathcal{C}_1$ are connected by paths that contain neither a_0 nor a_1. So, in fact, this case is not possible.

□

The next example shows that the maximality of $\mathcal{C}_0$ in Lemma 3.2.10 is essential.

Example 3.2.11. Consider the graph in Figure 3.9. All requirements of Lemma 3.2.10 are fulfilled for this graph except the maximality of $\mathcal{C}_0$: $\mathcal{C}_0$ is a 1-complex a_0-component, $\mathcal{C}_1$ is a 1-complex a_1-component, and $a_1 \in \mathcal{C}_0$. The entanglement of the graph is 2, although $\mathcal{C}_/$ and $\mathcal{C}_1$ are disjoint. The Cop player has the following winning strategy. The cops expel the robber from $\mathcal{C}_1$, if she is there, and place one cop on vertex a_1. The robber visits vertex v and a cop goes there. The robber proceeds to w and the cop who is not on v occupies w. Then, if necessary, the cop from v forces the robber to leave $\mathcal{C}_1$ and follows her to a_1. The robber visits v again, the cop from a_1 goes there too. As vertex w is occupied, the robber has to remain in $\mathcal{C}_0$ and a_0. The cop from w goes to a_1 and captures the robber.

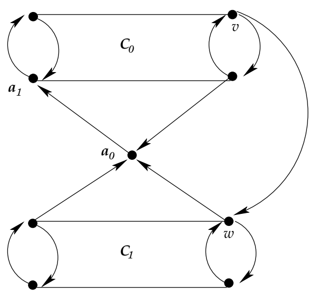

Figure 3.9: All requirements of Lemma 3.2.10 are fulfilled except the maximality of $\mathcal{C}_0$: $\mathcal{C}_0$ is a 1-complex a_0-component, $\mathcal{C}_1$ is a 1-complex a_1-component, and $a_1 \in \mathcal{C}_0$. The entanglement of the graph is 2.

3.3 Directed graphs of entanglement 2

We want to give a characterisation of the class of graphs that have entanglement at most two. Our goal is to restrict ourselves on certain winning strategies of the Cop player: he can consequently shrink the room available for the robber. Unlike some other games (e.g., for DAG-width) where we can assume a monotone winning strategy for the Cop player, in this case, a sort of weak monotonicity can be proved. The robber may be able to return several times to a vertex that has already been unavailable for her. But after playing a while, her room becomes smaller and she cannot visit a certain part of the graph any more.

We show that the Cop player can place a cop on a vertex a in the given graph $\mathcal{G}$, such that all a-components are simple, and then use the other cop to either expel the robber from every a-component or block all exit points of the component placing himself on a vertex b, in order to make the first cop free from guarding vertex a. In this case, the current component, in turn, is decomposed by b and the first cop is used to follow the robber. Finitely, there remain only components of entanglement 1 and the robber is captured.

Lemma 3.3.1. *Let $\mathcal{G} = (V, E)$ be a strongly connected graph. Let $I = \{0, 1, \ldots, m\}$, for some $m \in \{1, \ldots, |V| - 1\}$. For $i \in I$, let a_i be vertices in V and let $\mathcal{C}_i = (C_i, E_i, F_i)$ be 1-complex a_i-components. Let, further, a_i be in C_j, for all $i \neq j$, $i, j \in I$.*

If $\bigcap_{i \in I} C_i = \emptyset$, then $\operatorname{ent}(G) > 2$.

Note that the last condition of the lemma can be generalised for the case $k > 1$ to the condition: for every $\mathcal{C}_i$ and every set $\{v_1, \ldots, v_k\} \subseteq C_i$ there is a component $\mathcal{C}_m$ with $v_j \notin \mathcal{C}_m$, for all $j \in \{1, \ldots, k\}$. Though, we shall use only the above weaker condition.

Proof. If m is 1, then we have the conditions of Lemma 3.2.9, so assume that $m \geq 2$. The situation for $|I| = 3$ is shown in Figure 3.10.

We show that the robber can stay in a component $\mathcal{C}_i$ until both cops come to it and then escape from it reaching a cop free component. At the beginning, she starts in $\mathcal{C}_0$ and plays according to her $\mathcal{C}_0$-strategy until the second cop enters the component. In general, she enters a cop free component $\mathcal{C}_j$ and plays according to a $\mathcal{C}_j$-strategy waiting there for both cops to come. Let the second cop come to $\mathcal{C}_j$ on a vertex v, the first cop being on a vertex $w \in C_j$. At this point, since $\bigcap_{i \in I} C_i = \emptyset$, there is an a_i-component $\mathcal{C}_i$ with $w \notin \mathcal{C}_i$. If $v \in \mathcal{C}_i$, the robber plays her $\mathcal{C}_i$-strategy starting from v and assuming that the Cop player has put his cop on it. If $v \notin \mathcal{C}_i$, then the robber can escape from $\mathcal{C}_j$ and reach a_j, which is in C_i. On entering $\mathcal{C}_i$, the robber continues with a $\mathcal{C}_i$-strategy.

□

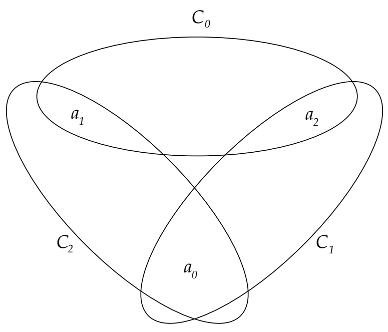

Figure 3.10: The components are 1-complex and include all vertices that induce other components. The Robber player has a winning strategy.

We give now a characterisation of graphs that have entanglement at most 2. It is clear that the entanglement is at most 2 if and only if the entanglement of all its strongly connected components is at most 2, so we can restrict ourselves to strongly connected graphs.

Lemma 3.3.2. *On a strongly connected graph $\mathcal{G} = (V, E)$, two cops have a winning strategy if and only if there exists a vertex $a \in V$ such that every a-component $\mathcal{C} = (V_C, E_C, F_C)$ is 1-simple .*

Proof. The direction from right to left is proven in Lemma 3.2.6: if every a-component is 1-simple, then $\text{ent}(\mathcal{G}) \leq 2$. We show the other direction.

Towards a contradiction, assume that the Cop player wins $\text{EG}(\mathcal{G})$, but loses $\text{EG}_2(\mathcal{C})$, for all $a \in V$ and all a-components $\mathcal{C}$ of $\mathcal{G}$.

We construct a sequence $a_0, a_1, \ldots, a_m$ of vertices from V and a sequence $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_m$ of corresponding a_i-components $\mathcal{C}_i = (V_i, E_i, F_i)$, where F_i are exit points of the components. We require that all $\mathcal{C}_i$ are maximal 1-complex a_i-components with respect to $\leq_{a_i}$. Further, we require that $\bigcap_{i=0}^{m} \mathcal{C}_i \neq \emptyset$.

Take an arbitrary vertex as a_0. There is a 1-complex a_0-component $\mathcal{C}_0$, due to the assumption. Choose among all such strongly connected components a maximal one with respect to $\leq_{a_0}$.

Suppose that a_i and $\mathcal{C}_i$ are already constructed, and every $\mathcal{C}_j$ is maximal with respect to $\leq_{a_j}$, for $j \leq i$, and we have $\bigcap_{j\leq i} \mathcal{C}_j \neq \emptyset$. Choose a vertex a_{i+1} in $\bigcap_{j\leq i} \mathcal{C}_j$ and a maximal a_{i+1}-component $\mathcal{C}_{i+1}$. Due to Lemma 3.2.10, it intersects all $\mathcal{C}_j$, for $j \leq i$ (otherwise ent$(\mathcal{G})$ would be greater than 2). According to Lemma 3.3.1, $\bigcap_{j\leq i+1} \mathcal{C}_j \neq \emptyset$ (otherwise ent$(\mathcal{G}) > 2$ and we get a contradiction to our assumption) and we can continue the construction.

Note that all a_i are not in $\mathcal{C}_i$. Finitely, for some $m < |V|$, there is no corresponding 1-complex a_{m+1}-component for a_{m+1} and the construction stops. This a_{m+1} is the desired vertex a with only 1-simple a-components, which is a contradiction to our assumption.

□

Proposition 3.3.3. *Let $\mathcal{G} = (V, E)$ be a graph. In the Entanglement game on $\mathcal{G}$, two cops have a winning strategy if and only if in every strongly connected component $\mathcal{C}$ of $\mathcal{G}$, there exists a vertex $a \in V$, such that every a-component of $\mathcal{C}$ is 1-simple.*

Corollary 3.3.4. *If a graph $\mathcal{G}$ has entanglement at most 2, then there is a vertex a in $\mathcal{G}$ such that the Cop player has a winning strategy that allows the robber to enter the vertex a only once.*

3.3.1 Deciding entanglement two

The proof of lemma 3.3.2 shows the structure of a strongly connected graph G with entanglement 2. It has a vertex a_0 such that the graph $\mathcal{G}\backslash a_0$ decomposes in simple a_0-components. We can divide them into two classes: *leaf* components, from which one cop expels the robber, and *inner* components, where one cop does not win, but blocks all exit points making the other cop free from guarding the simple components. It turns out that every inner component $\mathcal{C}_0$ again has a vertex a_1 such that $\mathcal{C}_0$ decomposes in simple a_1-components an so on. We shall show that a_1 is the vertex where the second cop stays (blocking all exit points of $\mathcal{C}_0$) when the first cop leaves a_0.

Lemma 3.3.5. *Let $\mathcal{G} = (V, E, F)$ be a graph with final vertices such that the graph (V, E) is strongly connected. If, for all $v \in V$, there is a cycle $\mathcal{C}$ in $\mathcal{G}\backslash v$ from that a final vertex of $\mathcal{G}$ is reachable in $\mathcal{G}\backslash v$ then $\mathcal{G}$ is 1-complex.*

Proof. Consider a position in a play of the Entanglement game with final vertices on $\mathcal{G}$ when a cop enters $\mathcal{G}$ for the first time. Without loss of generality it is the first cop. The second cop is still outside $\mathcal{G}$ and it is the robber's move. Let the robber and the first cop stay on a vertex v_0. There is a cycle $\mathcal{C}_0$ in $\mathcal{G}\backslash v_0$. Because (V, E) is strongly connected, there is a (cop free) path from the vertex v_0 to any vertex in V. Particularly, the robber can run to some vertex in $\mathcal{C}_0$. The strategy of the robber is to do so and to wait in $\mathcal{C}_0$ for the first cop to follow her on a vertex v_1. Then she goes to another cycle $\mathcal{C}_1$ and so on. If the second cop never comes into the graph, the play is infinite and the Robber player wins. Otherwise let v be the vertex where the first cop is when the second one comes and let C_v be a cycle in $\mathcal{G}\backslash v$ from which there is a path to an exit point in $\mathcal{G}\backslash v$. After the arrival of the second cop the robber goes to C_v and from there to an exit point and wins. ☐

Let $\mathcal{G} = (V, E, F)$ be a graph with final vertices. We call a vertex $v \in V$ a *blocking vertex* of $\mathcal{G}$, if there is no strongly connected component of $\mathcal{G}\backslash v$ from which there is a path to a final vertex. We denote the set of blocking vertices $B(\mathcal{G})$ and define a binary relation $\rightarrow$ on $B(\mathcal{G})$:

$$\rightarrow := \{(v, w) \mid w \text{ is not on a cycle in } \mathcal{G}\backslash v\}.$$

Lemma 3.3.6. *If $\mathcal{G} = (V, E, F)$ is a 1-simple graph with final vertices then the relation $\rightarrow$ on $B(\mathcal{G})$ is a total preorder, i.e., it is transitive and total.*

Proof. For transitivity, let u, v, w be in $B(\mathcal{G})$ and assume that it is $u \rightarrow v$ and $v \rightarrow w$. It follows that all cycle on that w is contain v and all cycles with v contain u. It follows that all cycles with w contain u and w is not on a cycle in $\mathcal{G}\backslash u$.

It remains to show the totality of $\rightarrow$. Because the reflexivity is trivial, let v and w be distinct vertices in $B(\mathcal{G})$. Assume that neither $v \rightarrow w$ nor $w \rightarrow v$ holds, i.e. w is on a cycle $\mathcal{C}_v$ in $\mathcal{C}\backslash v$ and v is on a cycle $\mathcal{C}_w$ in $\mathcal{C}\backslash w$. Further, every path from $\mathcal{C}_v$ to an exit point lead through v, because v is blocking, and there is such a path, because $\mathcal{G}$ is strongly connected. Consider the part of this path from v to a final vertex. Together with $\mathcal{C}_w$ it witnesses that w is not blocking in contradiction to the choice of w. □

Note that $\rightarrow$ is not necessarily antisymmetric, so we define an equivalence relation $\sim$ on $B(\mathcal{G})$, for a graph with final vertices $\mathcal{G}$, to be $\sim := \{(v, w) \mid v \rightarrow w \text{ and } w \rightarrow v\}$. Further, we extend the relation $\rightarrow$ on $B(\mathcal{G})/_{\sim}$. Let $[v]$ denote the equivalence class of v with respect to $\sim$. The binary relation $\rightarrow_\sim$ is well defined by $\rightarrow_\sim := \{([v], [w]) \mid v \rightarrow w\}$.

Lemma 3.3.7. *If $\mathcal{G} = (V, E, F)$ is a 1-simple graph with final vertices then the relation $\rightarrow_\sim$ on $B(\mathcal{G})$ is a total order.*

Proof. The transitivity and the totality are inherited by $\rightarrow_\sim$ from $\rightarrow$, the antisymmetry is guaranteed by including all not antisymmetric pairs of elements into the same class. □

If we have, for vertices v and w in a graph with final vertices $\mathcal{G} = (V, E, F)$, that $v \rightarrow w$ then we say that vertex v *blocks* vertex w.

Lemma 3.3.8. *If $\mathcal{G} = (V, E, F)$ is a 1-simple graph with final vertices of entanglement 2 then there is a vertex v in V that blocks all vertices from $B(\mathcal{G})$.*

Proof. Consider the relation $\rightarrow_\sim$ on $B(\mathcal{G})/_\sim$. According to Lemma 3.3.7, it is a total order, so it has an element $[u]$ with $[u] \rightarrow_\sim [w]$, for all $[w] \in B(\mathcal{G})$. An arbitrary vertex $v \in [u]$ blocks all blocking vertices of $\mathcal{G}$. □

Now we are ready to give a structural characterisation of graphs that have entanglement two.

Definition 3.3.9. An *entanglement two-decomposition* of a strongly connected graph $\mathcal{G} = (V, E)$ is a triple $(\mathcal{T}, F, g)$, where $\mathcal{T}$ is a nontrivial directed tree $\mathcal{T} = (T, E_T)$ with root r and edges directed away from the root, and F and g are functions $F : T \rightarrow 2^V$ and $g : T \rightarrow V$ with the following properties:

1. $F(r) = V$,
2. $g(v) \in F(v)$ for all $v \in T$,

3. if $(v, w) \in E_T$ then $F(w) \subsetneq F(v)$,
4. if $(v, w_1) \in E_T$ and $(v, w_2) \in E_T$, then $F(w_1) \cap F(w_2) = \emptyset$, for $w_1 \neq w_2$,
5. if $(v, w) \in E_T$ then $\mathcal{G}[F(w)]$ is a strongly connected component of $\mathcal{G}[F(v) \backslash g(v)]$,
6. if $vE_T = \{w_1, \ldots, w_m\}$ then the subgraph of $\mathcal{G}$ induced by the vertex set $\Big(F(v) \backslash g(v)\Big) \backslash \big(\bigcup_{i=1}^{m} F(w_i)\big)$ is acyclic,
7. no exit point of $\mathcal{G}[F(v)]$ is on a cycle in $\mathcal{G}[F(v) \backslash g(v)]$,
8. if $v \in T$ is a leaf then $F(v) \backslash g(v)$ is acyclic.

We shall call tree vertices and (abusing the notation) their F-images *bags* and g-images *decomposition points*.

Observe that successors of a bag are partially ordered in the sense that, for each bag v, its successors $vE_T = \{w_1, \ldots, w_m\}$ form a DAG $\mathcal{D} = (vE_T, E_D)$ such that, for all $w_i, w_j \in vE_T$, w_j is reachable from w_i in $\mathcal{D}$ if and only if $F(w_j)$ is reachable from $F(w_i)$ in $\mathcal{G}[F(v) \backslash g(v)]$. An example of a graph and its entanglement two-decomposition is given in Figure 3.11.

Proposition 3.3.10. *A strongly connected graph $\mathcal{G} = (V, E)$ has entanglement at most 2 if and only if $\mathcal{G}$ has a entanglement two-decomposition.*

Proof. Having $\mathrm{ent}(G) = 2$ we construct the tree $\mathcal{T} = (T, E_T)$ and the functions F and g in a top-down manner. In each step we enlarge the tree adding to a bag v that is currently a leaf some successors $\{w_1, \ldots, w_m\}$ and define the functions F and g on them. We require that all $g(w_i)$-components of $\mathcal{G}[F(w_i)]$ are 1-simple.

There is a vertex $a_0 \in V$ such that all a_0-components of $\mathcal{G}$ are 1-simple (Lemma 3.3.2). Set $F(r) = a_0$ and $g(r) = V$. In general, for every bag v that is a leaf of the already constructed part of the tree, let $C_1, \ldots, C_m$ be all strongly connected components of $F(v) \backslash g(v)$. If there are no such components (i.e., $m = 0$), skip this bag and proceed with a next one, if there is any. If m is at least 1 create, for each $i \in \{1, \ldots, m\}$, a successor w_i of v and set $F(w_i) = C_i$. From the construction we know that each C_i induces a 1-simple $g(v)$-component. If it has a vertex a whose removal makes the

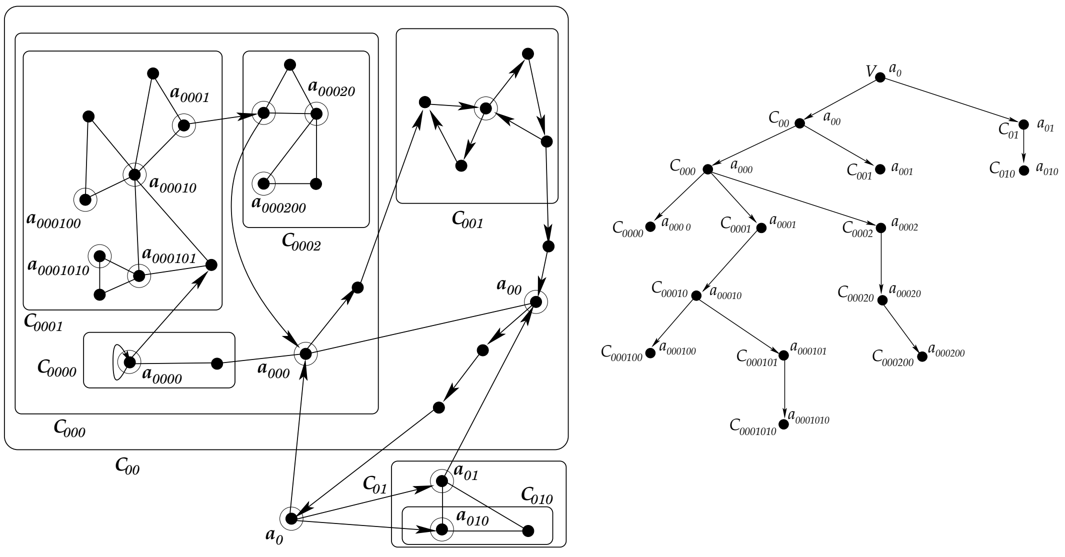

Figure 3.11: A typical graph of entanglement 2 and its entanglement two-decomposition. On the left, the components (images of function F) are shown as squares (only up to level 4), blocking points (images of function g) are shown in circles. On the right, the decomposition tree of the graph. The bags have images from functions F and g as labels.

component acyclic, i.e., one cop wins $\mathrm{EG}(\mathcal{G}[C_i])$, then set $g(w_i) = a$. If one cop cannot win $\mathrm{EG}(\mathcal{G}[C_i])$ then, according to the definition of a 1-simple component, he can block all exit points (to win with help of the other cop), i.e., he can place himself on a blocking vertex. Among all blocking vertices there is a vertex a that blocks all vertices in $B(\mathcal{G}[C_i])$, due to Lemma 3.3.8. Set $g(v) = a$. Then all a-components of $\mathcal{G}[F(w_i)]$ are 1-simple.

It follows from the construction that all requirements of the entanglement two-decomposition are fulfilled.

Conversely, a entanglement two-decomposition induces a winning strategy for two cops on G. Note that if a cop is on a vertex $g(v)$, for a bag v, and the robber is in a bag on a lower level of the tree, then the cop blocks the robber in the bags under v. Consider a vertex a of the graph with the robber on it. Let v be the bag with the smallest F-image (it is the lowest in the tree) among all with $a \in F(v)$ and let vE_T be $\{w_1, \ldots, w_m\}$, for $m \geq 0$ (if $m = 0$ then vE_T is empty). The Cop player waits for the robber to enter a component $\mathcal{G}[F(w_i)]$ or to go to $g(v)$. In the first case, he plays the same strategy with w_i instead of v. This descending along the tree is finite and on some level (without loss of generality already on that where v is) the robber visits $g(v)$. One cop goes there. If the robber proceeds to a component $\mathcal{G}[F(w_i)]$, the second cop continues to chase her using the same strategy until she reaches a leaf bag and is captured. If she leaves $F(v)$ and enters a brother bag v' of v, the cop from v follows her there and so on until the robber is forced to go to $g(u)$, where u is the predecessor of v. The first cop goes to $g(u)$ as well and chases the robber in this manner upwards. This process is finite and when the robber goes downwards, the second cop plays the described strategy with the difference that the robber cannot climb so high as before. Continuing in this way the cops finally capture the robber.

□

Now we describe a polynomial algorithm (that uses a recursive subprocedure) determining whether a given strongly connected graph has entanglement at most 2 and, if so, returning a winning strategy for the Cop player.

Algorithm 3.3.11. *Entanglement-Two*

Input: A strongly connected directed graph $\mathcal{G} = (V, E)$

Question: Is the entanglement of $\mathcal{G}$ at least 2?

```
1: for all $a \in V$ do
2:   compute $a$-components $\mathcal{C}_1, \dots, \mathcal{C}_m$ of $\mathcal{G} \backslash a$
3:   for all $i \in \{1, \dots, m\}$ do
4:     if $One\text{-}Cop\text{-}Wins(\mathcal{C}_i)$ returns "Cop wins" then
5:       continue with the next $i$
6:     if $B(\mathcal{C}_i) = \emptyset$ then
7:       return no
8:     else find a vertex $b$ that blocks all blocking vertices with procedure Find-Block-Exit
9:     Let $F$ be the set of exit points of $\mathcal{G}[C_i]$
10:    call $Final\text{-}Entanglement\text{-}Two((\mathcal{G}[C_i], F), b)$
11:    if $Final\text{-}Entanglement\text{-}Two$ returns "no" then
12:      return no
13: return yes
```

Algorithm 3.3.12. *Final-Entanglement-Two*

Input: A strongly connected graph with final vertices $\mathcal{G} = (V, E, F)$, $b \in V$

Question: Does the Cop player have a winning strategy in the Entanglement game with final vertices $\mathrm{EG}_2(\mathcal{G})$?

```
1: compute SCCs $\mathcal{C}_1, \dots, \mathcal{C}_m$ of $\mathcal{G} \backslash b$
2: for all $i \in \{1, \dots, m\}$ do
3:   if $One\text{-}Cop\text{-}Wins(\mathcal{C}_i)$ returns "Cop wins" then
4:     continue with the next $i$
5:   if $B(\mathcal{C}_i) = \emptyset$ then
6:     return no
7:   find a vertex $b' \in B(\mathcal{C}_i)$ that blocks all blocking vertices with prosedure Find-Block-Exit
```

```
8:   call $Final$-$Entanglement$-$Two(C_i, b')$
9:   if $Final$-$Entanglement$-$Two(C_i, b')$ returns “no” then
10:      return no
11: return yes
```

Complexity of Algorithm *Entanglement-Two*

Let n be the size of the given graph $\mathcal{G} = (V, E)$. The first line of the procedure *Entanglement-Two* is executed at most n times. In the second line, strongly connected components of $\mathcal{G}$ are computed, which can be done with Tarjan's algorithm ([54]) in time $\mathcal{O}(n)$. Checking whether a component has entanglement one in lines 3,4 and 5 is linear in n via procedure *One-Cop-Wins*. Note that deciding whether a given graph has vertex disjoint cycles, as needed in line 5 of *One-Cop-Wins*, can be done in linear time according to [22]. Algorithm *Find-Block-Exit* computes $B(\mathcal{C}_i)$ needed in line 6 in linear time with respect to the size of its input. The same algorithm finds a blocking vertex for line 8. Because the time complexity of the procedure *Final-Entanglement-Two* is greater than $\mathcal{O}(n)$, it is crucial for the total complexity.

The procedure *Final-Entanglement-Two* is similar to Algorithm *Entanglement-Two* and runs in linear time multiplied by the number of recursive calls in line 8. They occur in parallel on every level of the decomposition tree, except the on level with the lowest leaves. There are at most $n - 1$ such levels. In total, the time complexity of Algorithm *Entanglement-Two* is in $\mathcal{O}(n^3)$.

Algorithm 3.3.13. *One-Cop-Wins*

Input: A directed graph $\mathcal{G}$
Output: “Robber wins” or “Cop wins”

```
1: if $\mathcal{G}$ is acyclic then
2:   return “Cop wins”
3: compute SCCs $\{\mathcal{G}_1, \dots, \mathcal{G}_m\}$ of $\mathcal{G}$
4: for all $i \in \{1, d \dots, m\}$ do
5:   if there are vertex disjoint cycles in $\mathcal{C}_i$ then
6:     return “Robber wins”
7:   find a cycle $\mathcal{C} = (C, E_C)$ in $\mathcal{C}_i$
8:   enumerate vertices of $\mathcal{C}$ with $\{0, \dots, t-1\}$
9:   $Start \leftarrow 0$, $End \leftarrow (t-1)$
10:  $i \leftarrow Start$
11:  while $Start \leq i \leq End$ do
12:    make a DFS from $i$ in $\mathcal{C}_i \backslash C$ deleting edges while backtracking and checking:
13:    if an edge leads to a vertex $p \in C$ then
14:      $Start \leftarrow \min(i, p)$
15:      $End \leftarrow \max(i, p)$ {DFS is continued}
16:  delete from $\mathcal{C}$ vertex with smallest number between $Start$ and $End$
17:  if a cycle in $\mathcal{C}_i$ remains then
18:    return “Robber wins”
19: return “Cop wins”
```

Algorithm 3.3.14. *Find-Block-Exit*

Input: a strongly connected graph with final vertices $\mathcal{G} = (V, E, F)$
Output: vertex $v \in B(\mathcal{G})$ that blocks all blocking vertices or “No blocking vertex”

```
1: compute the set of exit points of $C$
2: trace paths to exit points in parallel backwards until a cycle is found at the beginning
   of each path
3: if the intersection of all cycles is empty, then return “No blocking vertex”
```

```
4: return any vertex from the intersection
```

Our next purpose is to establish a connection to the characterisations of graphs of entanglement 2 given by Belkhir and Santocanale in [7].

Every undirected graph $\mathcal{G} = (V, E)$ of entanglement at most two is a union of trees $\mathcal{T}$ that, in addition, for every edge $\{a, b\}$, can have vertices $v_1^{a,b}, \ldots, v_m^{a,b}$ with edges $\{a, v_i^{a,b}\}$ and $\{b, v_i^{a,b}\}$, for every $i \in \{1, \ldots, m\}$. For a entanglement two-decomposition of a strongly connected component of $\mathcal{G}$, consider the corresponding tree $\mathcal{T} = (V_T, E_T)$ that forms a component of $\mathcal{G}$. Add a new vertex r to the tree and a directed edge (r, v) to an arbitrary leaf $v \in V_T$. We get a decomposition tree after orienting all edges from E_T away from the root r and deleting all other leaves than v. The functions F and g can be defined as follows. Set $F(r)$ to be V_T and $g(r)$ to be r. In general, if, for a bag w, the functions F and g on w are already defined, let $\mathcal{C}$ be a strongly connected component of $\mathcal{G}[F(w)]\backslash g(w)$. Choose a vertex u in $\mathcal{C}$ with an edge between w and u and set $F(u) = \mathcal{C}$ and $g(u) = u$.

3.3.2 An Application of structural description

The structure of a graph with entanglement at most two that is given in Lemma 3.3.2 allows to compare the entanglement of such graphs with its Kelly-width. In [9] it is shown that if a graph has entanglement k then its DAG-width is at most $k + 1$.

Proposition 3.3.15. *For any graph $\mathcal{G}$, if $\operatorname{ent}(\mathcal{G}) \leq 2$, then both the DAG-width and the Kelly-width of $\mathcal{G}$ are at most 3.*

Proof. We first use the entanglement two decomposition to show that the Cop has a winning strategy in the DAG Game on graphs that have entanglement two and then adjust this strategy to the Kelly Game. Recall that the DAG-width (respectively the Kelly-width) of a graph is the minimal number of cops needed to capture the robber in the DAG Game (respectively, in the Kelly Game). Without loss of generality, $\mathcal{G}$ is acyclic. Consider a entanglement two-decomposition $(\mathcal{T}, F, g)$ of $\mathcal{G}$. The idea of the desired winning strategies of the Cop player in the DAG Game is, first, to place one cop on the vertex whose existence is guaranteed by Lemma 3.3.2. In general, assume that, for a bag v, a cop is on a blocking vertex $g(v)$ and the robber is on a vertex in $F(w)$, for a successor bag w of v. The component $F(w)$ has also a blocking vertex $g(w)$. Another cop (who is not on $g(v)$) goes to $g(w)$ and the third cop visits every vertex in $F(w)$ that is not in a strongly connected component of $F(w)$. Thus the robber is forced to move down the decomposition tree (i.e., she is blocked in a shrinking part of $\mathcal{G}$) and finally loses.

The strategy of the Cop player in the Kelly Game is similar. Assume that a cop is on a blocking vertex $g(v)$. The Cop player does not know where the robber is, so he decontaminates a strongly connected component of $F(v)\backslash g(v)$ as described for the DAG Game, moves a cop back on vertex $g(v)$ and continues with the next strongly connected component.

Note that both winning strategies are monotone. □

Proposition 3.3.15 gives the best possible upper bound for the number of cops needed to capture the robber in the same graph in the DAG Game and Kelly Game. Note that the third cop in the DAG Game and the Kelly Game is used to force the robber to move.

Proposition 3.3.16. *There is a graph $\mathcal{G}$ with entanglement two, but DAG-width and Kelly-width 3.*

Proof. Let $\mathcal{G}$ be the graph consisting of two directed cycles of length 4 connected by two undirected edges, formally: the graph $\mathcal{G}$ is $\mathcal{G} = (V, E)$ with $V = \{v_0, \ldots, v_3, w_0, \ldots, w_3\}$, and

$$\begin{aligned} E = \{&(v_i, v_{i+1(\text{mod } 4)}), (w_i, w_{i+1(\text{mod } 4)}) \mid i = 1, 2, 3\} \\ &\cup \{(v_0, w_0), (w_0, v_0), (v_2, w_2), (w_2, v_2)\}. \end{aligned}$$

See Figure 3.12 for an illustration. It is easy to verify that the entanglement of $\mathcal{G}$ is two and the DAG-width and the Kelly-width are three. □

3.4 Towards decomposing graphs of entanglement k

In order to prove a statement analogous to Proposition 3.3.3, we need to generalise some definitions. In a play of the Entanglement game, the Cop player would like to cut the graph $\mathcal{G}$ into several parts by placing some cops on vertices $v_1, \ldots, v_p$ rather than just one cop on a vertex as it is the case for two cops. The strongly connected components of

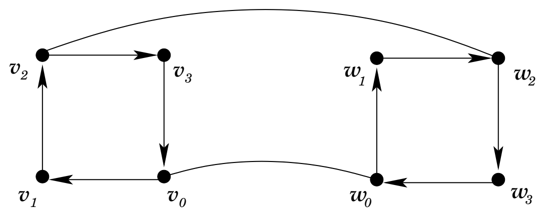

Figure 3.12: An example of a graph with entanglement 2 and DAG-width and Kelly-width 3.

$\mathcal{G} \backslash \{v_1, \ldots, v_p\}$ should be simple if the remainder cops can either win the Entanglement game on the strongly connected component or block some exit points to make some cops from $v_1, \ldots, v_p$ free from guarding the strongly connected component such that they can help to capture the robber in it.

It must not be the case that the cops in the component block all exit points as in Section 3.2. It can suffice if they block only some of exit points and a cop becomes free. Then together with this additional cop they block other exit points, another cop from outside comes into the component and so on until the robber is captured. Consider the following example.

Example 3.4.1. Let $\mathcal{G} = (V, E)$ be an undirected graph with $V = \{a_1, \ldots, a_m\} \cup \{b_1, \ldots, b_m\}$, for some $m > 1$ and edge relation $E = E_1 \cup E_2 \cup E_3$ where E_1 is $\{\{a_i, b_i\} \mid i = 1, \ldots, m\}$, E_2 is $\{\{a_m, b_i\} \mid i = 1, \ldots, m-1\}$, E_3 is $\{\{b_i, b_j\} \mid i, j = 1, \ldots, m\}$ (see Figure 3.13). Suppose that $m+1$ cops and a Robber play a Entanglement game with final vertices on a graph with final vertices $(\mathcal{G}, \{a_1, \ldots, a_m\})$ and the $(m+1)$-st cop has already moved. Let m cops guard the subgraph $\mathcal{G}' := \mathcal{G}[\{b_1, \ldots, b_m\}]$ on vertices $a_1, \ldots, a_m$. The Cop player can use only one cop additionally to those on a_i. This one cop is unable to capture the robber in $\mathcal{G}'$, so he wants to block an exit point b_i to make the cop on a_i free. For this, every b_i is good except b_m, because it is not an exit point. Therefore, it is reasonable for the robber to stay in b_m, but being expelled from there she visits another b_i and the additional cop blocks a_i. In the remainder of the play this cop will stay on b_i and the just described process repeats with the cop from a_i. The robber has a strategy to force the Cop player to use all his guarding cops in $\mathcal{G}'$, if he wants to win.

For a natural number $m \leq 1$, an *graph with m-guarded final vertices* is a structure $\mathcal{G} = (V, E, I, F, block)$ with edges $E \subseteq V^2$, initial vertices $I \subseteq V$, final vertices $F \subseteq V$ and a function $block : \{1, \dots, m\} \to \mathcal{P}(F)$. The intuition is that $\mathcal{G}$ is a subgraph of a graph $\mathcal{G}'$, and V is guarded by a set $w_1, \dots, w_m$. Let the robber start in I and let m cops guard the exit points of $\mathcal{G}$ on $w_1, \dots, w_m$. A final vertex v is in $block(i)$, if there is a path from v to w_i in $\big((\mathcal{G}' \backslash \mathcal{G}) \cup \{v\}\big) \backslash \{w_1, \dots, w_{i-1}, w_{i+1} \dots, w_m\}$. Informally, the function *block* describes which exit points must be blocked by a cop to make the cop on a guarding vertex w_i free.

Definition 3.4.2. Let $\mathcal{G} = (V, E, I, F, block)$ be a graph with m-guarded final vertices. The *Entnglement game with m-guarded final vertices* $\mathrm{EG}_{\mathrm{k}}^{\mathrm{m}}(\mathcal{G})$ is played on $\mathcal{G}$ as follows. There are two players: the Cop player who controls $(m + k)$ numbered cops, and the Robber player who controls a robber. At the beginning, all cops stay outside the graph and the Robber chooses a vertex in I to place the robber there in the first move. The players move in turn. The Cop player can take one cop from his vertex or from outside and place him on the position where the robber currently is, or he can let his cops idle. The robber must move to a vertex where an edge from her current vertex leads to, no matter whether a cop occupied her current vertex in the previous move or not. The target vertex must be free of cops. If the robber is unable to do that, the play ends and she loses. If the play lasts infinitely long, the Robber player wins.

In addition, there is another winning condition for the Robber: if the robber reaches the vertex v with $v \in block(i)$ after the Cop player has moved the $(k + i)$-th cop for $1 \leq i \leq m$ (i.e., has brought him from outside into the graph), then the Robber player wins as well.

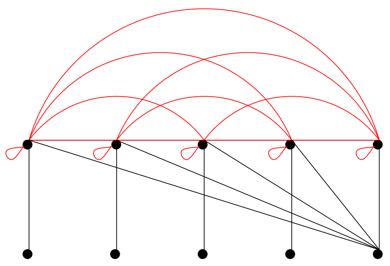

Figure 3.13: New cops come consequently in the strongly connected component

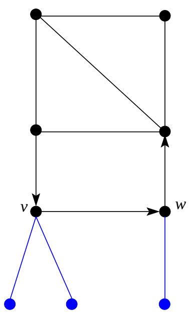

Figure 3.14: Three cops on the lowest vertices guard the rest part of the graph. There is one additional cop for it. If the robber starts on v, she loses, if she starts on w, she wins.

A graph with m-guarded final vertices $\mathcal{G} = (V, E, I, F, block)$ is *k-m-complex*, if the robber has a winning strategy in the Entnglement game with m-guarded final vertices $\mathrm{EG}_{\mathrm{k}}^{\mathrm{m}}(\mathcal{G})$, otherwise the graph is *k-m-simple*.

We call cops that guard a subgraph *guarding* and the others *active*.

The next example shows that an analogon of Lemma 3.2.2 does not hold.

Example 3.4.3. Consider the graph in Figure 3.14. Three cops on the lowest vertices guard the subgraph resulting from deleting these vertices. Assume that the Cop player has one active cop for the rest part. The robber chooses to start on a vertex different from v and wins, because the Cop player can make at most one cop free (if the robber starts on w), which is not sufficient for the upper part. But if the robber starts on v the active cop goes to v too, becomes guarding, but makes two cops free of guarding and active. These two cops capture the robber.

Let $\mathcal{G} = (V, E, I, F, block)$ be an graph with m-guarded final vertices. Assume that the Cop player has $k + m$ cops. We say that a strategy σ_C *allows an introduction* of m_0 cops, if for all strategies σ_R for the Robber, in the play consistent with σ_C and σ_R the Cop player brings $k + m_0$ cops into the graph.

Lemma 3.4.4. *Let, for some $k, m > 0$, $\mathcal{G}_v = (V, E, \{v\}, F, block)$ be a k-m-simple graph with m-guarded final vertices and let $\mathcal{G}_w = (V, E, \{w\}, F, block)$ be a k-m-complex graph with m-guarded final vertices. If (V, E) is strongly connected then every Cop's winning* *strategy for the game* $\mathrm{EG}^{\mathrm{m}}_{\mathrm{k}}(\mathcal{G}_v)$ *allows an introduction of more cops than any Cop's strategy for the game* $\mathrm{EG}^{\mathrm{m}}_{\mathrm{k}}(\mathcal{G}_w)$*.*

The proof is analogous to the proof of Lemma 3.2.2.

We were not able to prove an analogon of Proposition 3.3.3. However, we conjecture that it holds. We need another definition to formulate the conjecture.

Definition 3.4.5. Let $\mathcal{G} = (V, E)$ be a strongly connected directed graph. Let $v_0, \dots, v_{m-1}$ be a sequence of distinct vertices. Let $\mathcal{C} = (C, E_C, I, F, \mathit{block})$ be a Entnglement game with m-guarded final vertices. We say that $\mathcal{C}$ is a $(v_0, \dots, v_{m-1})$-component of $\mathcal{G}$, if (C, E_C) is a strongly connected component of $\mathcal{G}$, $I = \{v \in C \mid \text{ there is a vertex } w \in V \backslash C \text{ with } (w, v) \in E\}$, $F = \{v \in C \mid \text{ there is a vertex } w \in V \backslash C \text{ with } (v, w) \in E\}$, and $\mathit{block} : \{1, \dots, m\} \to 2^F$ with $v \in \mathit{block}(i)$ if and only if there is a path from v to v_{i-1} in $\mathcal{G}$.

Conjecture 3.4.6. *A directed graph* $\mathcal{G} = (V, E)$ *has entanglement at most* k *if and only if every its strongly connected component has* $m \leq k$ *vertices* $v_0, \dots, v_{m-1}$ *such that every* $(v_0, \dots, v_{m-1})$*-component is* k*-*m*-simple.*

Proposition 3.3.15 uses decompositions. Note that a decomposition implies *monotone* Cops' strategies in games that characterise the DAG-width and the Kelly-width. This means that if we can prove a generalisation of the decomposition characterisation for arbitrary entanglement, we can also show that DAG-width and Kelly-width of a graph are at most its entanglement plus one.

Chapter 4

Comparing Measures

In this chapter we summarise results about relations between different measures. An schematic overview can be found in Appendix A. We are interested how does one measure bounded on a graph class, bound another measure on this class.

We start with a usefull statement that allows to rise a non-boundness result for a graph, to a class of graphs.

Definition 4.0.7 ([34])**.** Let $\mathcal{G}_1 = (V_1, E_1)$ and $\mathcal{G}_2 = (V_2, E_2)$ be directed graphs. The *lexicographic product* $\mathcal{G}_1 \bullet \mathcal{G}_2$ of $\mathcal{G}_1$ and $\mathcal{G}_2$ is the directed graph $\mathcal{G} = (V, E)$ with $V = V_1 \times V_2$ and $E = \{((x, y), (x', y')) \mid (x, x') \in E_1$, or $x = x'$ and $(y, y') \in E_2\}$.

Proposition 4.0.8 ([34])**.** *Let $\mathcal{G}$ be a directed graph and let $\mathcal{K}_n$ be the complete graph on n vertices. Then at least k cops have a winning strategy on $\mathcal{G}$ in the DAG Game (in the Kelly Game) if and only if at lest $n \cdot k$ cops have a winning strategy on $\mathcal{G} \bullet \mathcal{K}_n$ in the DAG Game (in the Kelly Game).*

For convenience, we define some more measures that correspond to the DAG Game, the Kelly Game and the directed treewidth (but are different from the DAG-width, the Kelly-widthand the directed treewidth, see Propositions 1.6.7, 1.8.6, 2.1.7, and 2.1.8).

Definition 4.0.9. Let $\mathcal{G}$ be a directed graph. The *non-monotone DAG-width* $\mathrm{nmDAGw}(\mathcal{G})$ is the minimal number of cops needed to capture a robber in the DAG Game on $\mathcal{G}$. The *non-monotone Kelly-width* $\mathrm{nmKw}(\mathcal{G})$ is the minimal number of cops needed to capture a robber in the Kelly Game on $\mathcal{G}$. The *directed treewidth game measure* $\mathrm{dtwG}(\mathcal{G})$ is the minimal number of cops needed to capture a robber in the Directed treewidth Game on $\mathcal{G}$.

We do not consider non-monotone variants of the Directed treewidth Game, because we do not know any results that relate them to measures other than the directed treewidth. The relations are, however, described in Section 2.1.

The last measure we take into consideration is defined by Safari in [51] and is is a restricted version of the D-width.

Definition 4.0.10 ([51])**.** Let $\mathcal{G} = (V, E)$ be a directed graph. A set $X \subseteq V$ is a *good separator*, if, for every minimal strongly connected set $H \subseteq V$, $H \subseteq X \cup C$, where C is the vertex set of a strongly connected component $\mathcal{C}$ of $\mathcal{G} \backslash X$.

Further, we say that $\mathcal{G}$ satisfies the *augmenting condition* if, for any bramble $\mathcal{B}$ and any cover $X \subseteq V$ of $\mathcal{B}$, there exists a directed graph $\mathcal{G}'$ such that

- $\mathcal{G}$ and $\mathcal{G}'$ have havens of the same orders,
- X is a good separator in $\mathcal{G}'$, and every strongly connected set of $\mathcal{G}$ is a strongly connected set of $\mathcal{G}$.

Now we are ready to discuss relations between different measures. We start with the directed treewidth. Recall the definition of extended directed treewidth from Section 1.5. The following result follows trivially from definitions. Note that an arboreal decomposition is also an arboreal pre-decomposition.

Proposition 4.0.11. *For any directed graph $\mathcal{G}$, we have* $\mathrm{extdtw}(\mathcal{G}) \leq \mathrm{dtw}(\mathcal{G})$.

Proposition 4.0.12 ([9, Proposition 5.4])**.** *For any directed graph $\mathcal{G}$, we have* $\mathrm{extdtw}(\mathcal{G}) \leq \mathrm{DAGw}(\mathcal{G})$.

Proposition 4.0.13 ([9, Corollary 1])**.** *For any directed graph $\mathcal{G}$, we have* $\mathrm{dtw}(\mathcal{G}) \leq \mathrm{dw}(\mathcal{G})$.

For graphs satisfying the augmenting condition, we have the following statement proved in [51].

Proposition 4.0.14 ([51, Corollary 3])**.** *For any directed graph $\mathcal{G}$ that satisfies the augmenting condition, we have* $\mathrm{dw}(\mathcal{G}) = \mathrm{tw}(\mathcal{G})$.

While, as stated in Proposition 1.4.8, there are classes of graphs with treewidth one and unbounded pathwidth, the former can be used to establish an upper bound for the latter.

Proposition 4.0.15. *[16] Let $\mathcal{G}$ be an undirected graph with n vertices. Then we have* $\mathrm{pw}(\mathcal{G}) \leq \mathcal{O}(\mathrm{tw}(\mathcal{G}) \cdot \log n)$.

Recall Proposition 2.2.4 saying that in the Directed treewidth Game Robber's winning strategies against less than k cops correspond to havens of order k. The following proposition limits the use of havens in the Directed treewidth Game.

Proposition 4.0.16 ([1])**.** *There is a directed graph $\mathcal{G}$ with directed treewidth at least* 4*, but no haven of order* 5*.*

Corollary 4.0.17. *There is a directed graph $\mathcal{G}$ with* $\mathrm{dtw}(\mathcal{G}) \geq 4$*, but* $\mathrm{dtwG}(\mathcal{G}) \leq 4$*.*

In the Directed treewidth Game, the robber can move only to a vertex from that she can go back to her original vertex. This constraint is not present in the DAG Game, which is the only difference between the games. So, the next property follows.

Proposition 4.0.18. *For any directed graph $\mathcal{G}$, we have* $\mathrm{dtwG}(\mathcal{G}) \leq \mathrm{nmDAGw}(\mathcal{G})$*.*

We now turn to the DAG-width and related measures. We give the next proposition from [9] in another form than the authors who relate the DAG-width to the directed treewidth rather than to the directed treewidth game measure.

Proposition 4.0.19 ([9, Proposition 5.3])**.** *For any directed graph $\mathcal{G}$, we have* $\mathrm{dtwG}(\mathcal{G}) \leq \mathrm{DAGw}(\mathcal{G})$*.*

The converse does not hold. Consider graphs $(\mathcal{T}_k^1)^{op}$ defined in Example 1.6.16. For any $k \geq 3$, the DAG-width of $(\mathcal{T}_k^1)^{op}$ is $k+1$, but it is easy to see how 2 cops win the Directed treewidth Game. (See [9] for a detailed proof.)

Proposition 4.0.20 ([9, Proposition 5.3])**.** *There exists a family of graphs* $\{\mathcal{G}_k \mid k \geq 2\}$ *with* $\mathrm{DAGw}(\mathcal{G}_k) = k+1$ *and* $\mathrm{dtwG}(\mathcal{G}_k) = 2$*.*

A monotone winning strategy is in particular, of course, a winning strategy, so the next statement is trivial.

Proposition 4.0.21. *For any directed graph $\mathcal{G}$, we have* $\mathrm{nmDAGw}(\mathcal{G}) \leq \mathrm{DAGw}(\mathcal{G})$.

The converse does not hold, recall Propositions 1.6.7 and 1.8.6.

Similar to the fact that the treewidth generalises the pathwidth, the DAG-width generalises the directed pathwidth. Again, as in the undirected case, the converse fails.

Proposition 4.0.22 ([9, Proposition 5.5])**.**

- *For any directed graph $\mathcal{G}$, we have* $\mathrm{DAGw}(\mathcal{G}) \leq \mathrm{dpw}(\mathcal{G}) + 1$.
- *There exists a family of graphs* $\{\mathcal{G}_k \mid k \geq 3\}$ *with* $\mathrm{DAGw}(\mathcal{G}_k) = 2$ *and* $\mathrm{dpw}(\mathcal{G}_k) = k$.

Proposition 4.0.23 ([34, Theorem 29])**.** *For any directed graph $\mathcal{G}$, we have* $\mathrm{DAGw}(\mathcal{G}) \leq \mathrm{nmKw}(\mathcal{G})$.

We turn to the Kelly-width and related measures.

Proposition 4.0.24 ([34, Theorem 20])**.** *For any directed graph $\mathcal{G}$, we have*

$$2 \cdot \mathrm{nmDAGw}(\mathcal{G}) - 1 \leq \mathrm{Kw}(\mathcal{G}).$$

Proposition 4.0.25 ([34, Theorem 22])**.** *For any k, there is a directed graph $\mathcal{G}_k$ such that* $\mathrm{Kw}(\mathcal{G}_k) \leq 3k$ *and* $\mathrm{nmDAGw}(\mathcal{G}_k) \geq 4k$.

The next proposition is analogous to 4.0.20.

Proposition 4.0.26 ([34, Theorem 28])**.** *There exists a family of graphs with unbounded Kelly-width and directed treewidth one.*

Proposition 4.0.27. *For any directed graph $\mathcal{G}$, we have* $\mathrm{nmKw}(\mathcal{G}) \leq \mathrm{Kw}(\mathcal{G})$.

We finally consider connections between the entanglement and some other measures. Before comparing the entanglement with the treewidth, we define the treewidth of a directed graph $\mathcal{G}$ to be the treewidth of the graph $\bar{\mathcal{G}}$. First, we note a relation between the treewidth and the DAG-width.

Proposition 4.0.28 ([9, Proposition 5.1],[42])**.**

- *For any directed graph $\mathcal{G}$, we have* $\mathrm{DAGw}(\mathcal{G}) \leq \mathrm{tw}(\mathcal{G}) + 1$.

- *There exists a family of graphs with unbounded treewidth and DAG-width one.*

Proposition 4.0.29 ([10, Proposition 8])**.** *There exists a family of graphs with unbounded entanglement and treewidth two.*

Despite this discrepancy, we can bound the entanglement in terms of the treewidth if we consider the number of vertices in the graph.

Proposition 4.0.30 ([10, Proposition 7])**.** *Let $\mathcal{G}$ be a directed graph with n vertices. Then we have* $\mathrm{ent}(\mathcal{G}) = \mathcal{O}\big((\mathrm{tw}(\mathcal{G}) + 1) \cdot \log n\big)$.

An analogous connection can be established between the entanglement and the DAG-width and the non-monotone DAG-width.

Proposition 4.0.31 ([9, Proposition 5.6])**.**

- *For any directed graph $\mathcal{G}$, we have* $\mathrm{nmDAGw}(\mathcal{G}) \leq \mathrm{ent}(\mathcal{G}) + 1$.
- *There exists a family of graphs with unbounded entanglement and DAG-width two.*
- *Let $\mathcal{G}$ be a directed graph with n vertices. Then we have* $\mathrm{ent}(\mathcal{G}) = \mathcal{O}\big((\mathrm{DAGw}(\mathcal{G}) + 1) \cdot \log n\big)$.

Recall the definition of the cycle rank from Section 1.9.

Proposition 4.0.32. *For any directed graph $\mathcal{G}$, we have* $\mathrm{ent}(\mathcal{G}) \leq \mathrm{cr}(\mathcal{G})$.

Proof. Without loss of generality, let $\mathcal{G}$ be strongly connected. Let $\mathrm{cr}(\mathcal{G})$ be k. Then k cops have a winning strategy in the Cycle rank Game . In the Entanglement Game, k cops have the following winning strategy. In every round i of a play the Cop player uses $k - i$ cops to force the robber to go to a vertex v_i, where the i-th cop in the Cycle rank Game would place himself. Then the i-th cop in the Entanglement Game goes there. From that time on, he remains on the vertex till the end of the play and the i-th round is over.

It remains to show that $k - i + 1$ cops can drive the robber to v_i in the i-th round, or, to say it differently, to win the i-th round. We do this inductively backwards on the number of the round. Note, that if the Cop player has a strategy to win every

j-th round with $k-j+1$ cops, for all $j > i$, then he just plays this strategy until the robber either loses or goes to v_i. In the last round, there is a vertex v_k in the current strongly connected component, whose removal makes it acyclic. This means, the robber will finally visit this vertex. □

Taking as $\mathcal{G}_n$ the complete graph of size $n+1$ we get the next proposition.

Proposition 4.0.33. *For every n, there exists a directed graph $\mathcal{G}_n$ of size n with* $\mathrm{ent}(\mathcal{G}_n) = \mathrm{cr}(\mathcal{G}_n)$.

Proposition 4.0.34. *There exist a class of graphs with entanglement two and unbounded cycle rank.*

Proof. Consider undirected paths $\mathcal{P}_n = (\{v_1, v_2, \ldots, v_n\}, \{\{v_i, v_{i+1}\} \mid 1 \leq i \leq n-1\})$ of length n, for $n \geq 3$. For simplicity, we confuse $\mathcal{P}_n$ and $\overleftrightarrow{\mathcal{P}_n}$. The entanglement of $\mathcal{P}_n$ is 2 for all n. But the cycle rank of $\mathcal{P}_n$ is $\log n$. A winning strategy for $\log n$ cops in the Cycle rank Game on $\mathcal{P}_n$ is to place the first cop on a "middle" vertex of the path and then to play in the same way on the subpath where the robber went. On the other hand, $(\log n) - 1$ cops lose against the robber, because she can always choose the larger strongly connected component of the remaining graph. □

Remark 4.0.35. Instead of $\mathcal{P}_n$ we can give even a graph without undirected edges. Consider the directed path with shortcuts of Proposition 2.1.12 (denote it by $\mathcal{P}'_n$). We have seen that the entanglement of it is two. Analogously to 4.0.34 it can be shown that $\mathrm{cr}(\mathcal{P}'_n)$ is also unbounded.

For the proof of the next proposition it suffices to take the directed path with n vertices that has additionally a self-loop on every vertex.

Proposition 4.0.36. *There exists a class of graphs $\mathcal{G}_n = (V, E)$ with $|V| = n$ such that* $\mathrm{cr}(\mathcal{G}_n) = n$ *and directed treewidth, DAG-width, Kelly-width, entanglement one.*

Recall the definition of clique-width from Section 1.9.

Proposition 4.0.37 ([26])**.** *Let $\mathcal{G}$ be a directed graph and let k be a natural number. If* $2^{2 \cdot tw(\overline{\mathcal{G}})} \leq k$ *then* $cw(\mathcal{G}) \leq k$.

Finally we compare entanglement with pathwidth and directed pathwidth.

Proposition 4.0.38. *Let $\mathcal{G}$ be a directed graph. Then* $\mathrm{ent}(\mathcal{G}) \leq \mathrm{dpw}(\mathcal{G})$.

Proof. Let $(\mathcal{P}, \mathcal{X}, f)$ be a path decomposition of $\mathcal{G}$ of minimal width k with $\mathcal{P} = (P, E_P)$ and $P = (p_0, \ldots, p_m)$. Then the largest bag has size $k + 1$. The strategy of the Cop player is to expel the robber from $f(p_0)$, then from $f(p_1)$ and so on, every time until the robber proceeds to a bag with a larger index. For that at most k cops are needed (remember that the robber has to move when it is her turn). She is then captured in the last bag. □

In fact, this bound is tight: we take as $\mathcal{G}_m$ the complete graph on m vertices.

Proposition 4.0.39. *For all natural $m > 0$, there exists a directed graph $\mathcal{G}_m$ with* $\mathrm{ent}(\mathcal{G}_m) = \mathrm{dpw}(\mathcal{G}_m) = m$.

Proposition 4.0.40. *There exist a class of undirected graphs $\mathcal{G}_m$ such that, for every $m > 1$, $\mathrm{ent}(\overleftrightarrow{\mathcal{G}_m}) = m$ and $\mathrm{pw}(\mathcal{G}_m) = \mathrm{dpw}(\overleftrightarrow{\mathcal{G}}) = 2m$.*

Proof. Let $\mathcal{G}_m$ be the graph $\mathcal{G}_m = (V, E)$ consisting of $2m + 2$ cliques K_i, for $1 \leq i \leq 2m + 2$, each of size m (we call K_{2m+2} central and the others peripheral), and $2m + 1$ connecting vertices $\{v_1, \ldots, v_{2m+1}\}$ such that every v_i is connected to every vertex in K_{2m+2} and to every vertex in K_i. First, we show that $\mathrm{ent}(\mathcal{G}_m) = m$.

$\mathrm{ent}(\mathcal{G}_m) \leq m$. The strategy for the Cop player is the following.

1. Force the robber to the central clique: if she is on a peripheral clique, force her to leave it using m cops and wait for her to enter the central clique.

2. Place cops on the central clique until the robber goes to a connecting vertex v_i. Place a new cop on it. If she comes back to the centre, fill it further, unless $m - 1$ cops are already there (then the robber is captured). If the robber goes to K_i, leave the cop on v_i block the way to the centre and use $m - 1$ other cops to capture the robber in K_i.

$\mathrm{ent}(\mathcal{G}_m) \geq m$. The cliques $K_i \cup \{v_i\}$ are $(m + 1)$-cliques, so the robber can escape $m - 1$ cops.

For the second part of the proof, recall that the pathwidth of an undirected graph is the minimum number of cops needed to capture a robber in the Pathwidth Game minus one. The strategy for $2m+1$ cops to capture the robber is to place m of them on the central clique and then, successively for all $1 \leq i \leq 2m+1$, to place a cop on v_i and m cops on K_i. (This strategy is both cop-monotone and robber-monotone.)

Why is the Cop player unable to win with $2m$ cops? We want to show that he has no robber-monotone strategy, what would be sufficient ([37]). At some time the Cop player will have to place m cops on the central clique. At this moment at least $m+1$ of the peripheral cliques, without lost of generality, $K_1, \dots, K_{m+1}$ are still contaminated. The cops first have to block every v_i, for $1 \leq i \leq m+1$, before they can use the cops from the central clique, because otherwise the robber can return to K_{2m+2} which contradicts the robber-monotonicity of the Cop's strategy. But the Cop player has at the moment only m free cops, so he is unable to do so. □

Conclusions

In Section 3.1 we studied entanglement on tournaments and gave lower and upper bounds for the difference between the size of tournaments and their entanglements.

In Section 3.3.1 we set Kelly-width in relation to entanglement for values at most two.

We compared directed pathwidth and entanglement in Chapter 4 and proved that entanglement of a graph is always at most its directed pathwidth.

Our main result presented in Sections 3.2 – 3.3.1 are two characterisations of directed graphs of entanglement two. One of them is a description of a structure (a decomposition) that the graph should have. From this decomposition we derived an $\mathcal{O}(n^3)$-time algorithm to decide whether a given graph has entanglement at most two.

In Section 3.4 we proposed definitions that generalise those used in the characterisations in Section 3.2.

We conjectured that at least the first characterisation can be generalised to the case of an arbitrary k. We stated in Section 3.3.1 that a generalisation of the decomposition characterisation of entanglement would imply that Kelly-width of a graph are at most its entanglement plus one.

Bibliography

[1] Isolde Adler. Directed tree-width examples. *J. Comb. Theory, Ser. B*, 97(5):718–725, 2007.

[2] Stefan Arnborg. Efficient algorithms for combinatorial problems with bounded decomposability - a survey. *BIT*, 25(1):2–23, 1985.

[3] Stefan Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic and Discrete Methods*, 8:277–284, 1987.

[4] Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for np-hard problems restricted to partial k-trees. *Discrete Appl. Math.*, 23(1):11–24, 1989.

[5] Emgad H. Bachoore and Hans L. Bodlaender. A branch and bound algorithm for exact, upper, and lower bounds on treewidth. In Siu-Wing Cheng and Chung Keung Poon, editors, *AAIM*, volume 4041 of *Lecture Notes in Computer Science*, pages 255–266. Springer, 2006.

[6] Janos Barat. Directed path-width and monotonicity in digraph searching. *Graphs and Combinatorics*, 22:161–172(12), June 2006.

[7] Walid Belkhir and Luigi Santocanale. Undirected graphs of entanglement 2. In Vikraman Arvind and Sanjiva Prasad, editors, *FSTTCS*, volume 4855 of *Lecture Notes in Computer Science*, pages 508–519. Springer, 2007.

[8] M. W. Bern, E. L. Lawler, and A. L. Wong. Linear-time computation of optimal subgraphs of decomposable graphs. *J. Algorithms*, 8(2):216–235, 1987.

[9] Dietmar Berwanger, Anuj Dawar, Paul Hunter, and Stephan Kreutzer. Dag-width and parity games. In *Proceedings of the 23rd Annual Symposium on Theoretical*

Aspects of Computer Science, STACS 2006, volume 3884 of *LNCS*, pages 524–536. Springer-Verlag, 2006.

[10] Dietmar Berwanger and Erich Grädel. Entangelemet - a meausre for the complexity of directed graphs with applications to logic and games. In *Proceedings of LPAR 2004, Montevideo, Uruguay*, volume 3452 of *LNCS*, pages 209–223. Springer-Verlag, 2005.

[11] Dietmar Berwanger, Erich Grädel, and Giacomo Lenzi. The variable hierarchy of the μ-calculus is strict. *Theory of Computing Systems*, 40:437–466, 2007.

[12] Dietmar Berwanger and Giacomo Lenzi. The variable hierarchy of the μ-calculus is strict. In *STACS 2005, Proceedings of the 22nd Symposium on Theoretical Aspects of Computer Science*, volume 3404 of *LNCS*, pages 97–109. Springer-Verlag, 2005.

[13] H. L. Bodlaender. A tourist guide through treewidth. Technical Report RUU-CS-92-12, Department of Information and Computing Sciences, Utrecht University, 1992.

[14] Hans L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *STOC*, pages 226–234, 1993.

[15] Hans L. Bodlaender. Treewidth: Algorithmic techniques and results. In Igor Privara and Peter Ruzicka, editors, *Proceedings 22nd International Symposium on Mathematical Foundations of Computer Science, MFCS'97, Lecture Notes in Computer Science, volume 1295*, pages 19–36, Berlin, 1997. Springer-Verlag.

[16] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998.

[17] Hans L. Bodlaender. Treewidth: Structure and algorithms. In Giuseppe Prencipe and Shmuel Zaks, editors, *SIROCCO*, volume 4474 of *Lecture Notes in Computer Science*, pages 11–25. Springer, 2007.

[18] Hans L. Bodlaender, Alexander Grigoriev, and Arie M. C. A. Koster. Treewidth lower bounds with brambles. In Gerth Stølting Brodal and Stefano Leonardi, ed-

itors, *ESA*, volume 3669 of *Lecture Notes in Computer Science*, pages 391–402. Springer, 2005.

[19] Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21(2):358–402, 1996.

[20] Hans L. Bodlaender and Babette van Antwerpen-de Fluiter. Parallel algorithms for series parallel graphs and graphs with treewidth two. *Algorithmica*, 29(4):534–559, 2001.

[21] H.L. Bodlaender. Classes of graphs with bounded tree-width. Technical Report RUU-CS-86-22, Department of Information and Computing Sciences, Utrecht University, 1986.

[22] H.L. Bodlaender. On disjoint cycles in graphs. Technical Report RUU-CS-90-29, Department of Information and Computing Sciences, Utrecht University, 1990.

[23] Daniela Bronner and Bernard Ries. An introduction to treewidth. Technical report, 2006.

[24] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

[25] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000.

[26] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.

[27] Erik D. Demaine and MohammadTaghi Hajiaghayi. Graphs excluding a fixed minor have grids as large as treewidth, with combinatorial and algorithmic applications through bidimensionality. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 682–689, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[28] L. C. Eggan. Transition graphs and the star-height of regular events. *Michigan Math. J.*, 10:385–397, 1963.

[29] Vibhav Gogate and Rina Dechter. A complete anytime algorithm for treewidth. In *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 201–208, Arlington, Virginia, United States, 2004. AUAI Press.

[30] Hermann Gruber and Markus Holzer. Finite automata, digraph connectivity, and regular expression size. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 39–50. Springer, 2008.

[31] D. M. Thilikos H. L. Bodlaender. Treewidth and small separators for graphs with small chordality. Technical Report UU-CS-1995-02, Department of Information and Computing Sciences, Utrecht University, 1995.

[32] R. Halin. s-functions for graphs. *Journal of Geometry*, 8(1):171–186, 1976.

[33] Paul Hunter. *Complexity and Infinite Games on Finite Graphs.* PhD thesis, Computer Laboratory, University of Cambridge, 2007.

[34] Paul Hunter and Stephan Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *SODA*, pages 637–644. SIAM, 2007.

[35] Thor Johnson, Neil Robertson, Paul D. Seymour, and Robin Thomas. Directed tree-width. *J. Comb. Theory, Ser. B*, 82(1):138–154, 2001.

[36] Lefteris M. Kirousis and Christos H. Papadimitriou. Interval graphs and seatching. *Discrete Mathematics*, 55(2):181–184, 1985.

[37] M. Kirousis and C. H. Papadimitriou. Searching and pebbling. *Theor. Comput. Sci.*, 47(2):205–218, 1986.

[38] Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science.* Springer, 1994.

[39] Stephan Kreutzer and Sebastian Ordyniak. Digraph decompositions and monotonicity in digraph searching. *CoRR*, abs/0802.2228, 2008.

[40] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *J. ACM*, 35(1):18–44, 1988.

[41] Daniel Meister, Jan Arne Telle, and Martin Vatshelle. Characterization and recognition of digraphs of bounded kelly-width. In Andreas Brandstädt, Dieter Kratsch, and Haiko Müller, editors, *WG*, volume 4769 of *Lecture Notes in Computer Science*, pages 270–279. Springer, 2007.

[42] Jan Obdržálek. Dag-width: connectivity measure for directed graphs. In *SODA*, pages 814–821. ACM Press, 2006.

[43] Jan Obdrzálek. Clique-width and parity games. In Jacques Duparc and Thomas A. Henzinger, editors, *CSL*, volume 4646 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 2007.

[44] B. Reed. Introducing directed tree width. In H. J. Broersma, U. Faigle, C. Hoede, and J. L. Hurink, editors, *Electronic Notes in Discrete Mathematics*, volume 3. Elsevier, 2000.

[45] B. A. Reed. Tree width and tangles: a new connectivity measure and some applications. In *Surveys in combinatorics, 1997 (London)*, volume 241 of *London Math. Soc. Lecture Note Ser.*, pages 87–162. Cambridge Univ. Press, Cambridge, 1997.

[46] Bruce A. Reed. Finding approximate separators and computing tree width quickly. In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 221–228, New York, NY, USA, 1992. ACM.

[47] Neil Robertson and P. D. Seymour. Graph minors. xiii: the disjoint paths problem. *J. Comb. Theory Ser. B*, 63(1):65–110, 1995.

[48] Neil Robertson and Paul D. Seymour. Graph minors. i. excluding a forest. *J. Comb. Theory, Ser. B*, 35(1):39–61, 1983.

[49] Neil Robertson and Paul D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.

[50] Neil Robertson, Paul D. Seymour, and Robin Thomas. Quickly excluding a planar graph. *J. Comb. Theory, Ser. B*, 62(2):323–348, 1994.

[51] Mohammad Ali Safari. D-width: A more natural measure for directed tree width. In Joanna Jedrzejowicz and Andrzej Szepietowski, editors, *MFCS*, volume 3618 of *Lecture Notes in Computer Science*, pages 745–756. Springer, 2005.

[52] Mohammad Ali Safari. *D-width, metric embedding, and their connections.* PhD thesis, Vancouver, BC, Canada, Canada, 2007.

[53] Paul D. Seymour and Robin Thomas. Graph searching and a min-max theorem for tree-width. *J. Comb. Theory, Ser. B*, 58(1):22–33, 1993.

[54] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

Appendix A

Relations between complexity measures of graphs

We give a schematic overview of relations between complexity measures we discuss. In Figure A the notation is the following. Here A and B denote measures, $f, g : \omega \rightarrow \omega$ are functions, k is a natural number and $\mathcal{G}$ is a graph.

Notation	**Explanation**
$A \;{}_{f(x)}\!\leqslant^{g(x)}\; B$	for all $\mathcal{G}$, $f(A(\mathcal{G})) \leqslant g(B(\mathcal{G}))$
$A \;{}_{f(x)}\!\geqslant^{g(x)}\; B$	for all $\mathcal{G}$, $f(A(\mathcal{G})) \geqslant g(B(\mathcal{G}))$
$A \;{}_{f(x)}\!\nleqslant^{g(x)}\; B$	for all k there exists $\mathcal{G}$ with $f(A(\mathcal{G})) > k \cdot g(B(\mathcal{G}))$
$A \;{}_{f(x)}\!\ngeqslant^{g(x)}\; B$	for all k there exists $\mathcal{G}$ with $f(A(\mathcal{G})) < k \cdot g(B(\mathcal{G}))$
n	number of vertices in $\mathcal{G}$
$Th.$	Theorem
$P.$	Proposition
$Cor.$	Corollary

At the subscripts given in the table above we write a reference to a source in the Bibliography in squared brackets. References on propositions in the text are given without brackets. Basic directed measures are given in blue colour. Abbreviations used in the following figure are given in the next table.

Notation	**Explanation**
extdtw	extended directed treewidth
augD-width	augmented D-width
DAGw	DAG-width
nmKw	the least number of cops needed to non-monotonely win Kelly Game
pw	pathwidth
nmDAGw	the least number of cops needed to non-monotonely win DAG Game
tw	treewidth
dpw	directed pathwidth
D-width	D-width
ent	entanglement
dtwG	the least number of cops to win Directed treewidth Game
cr	cycle rank
dtw	directed treewidth
Kw	Kelly-width
cw	clique-width

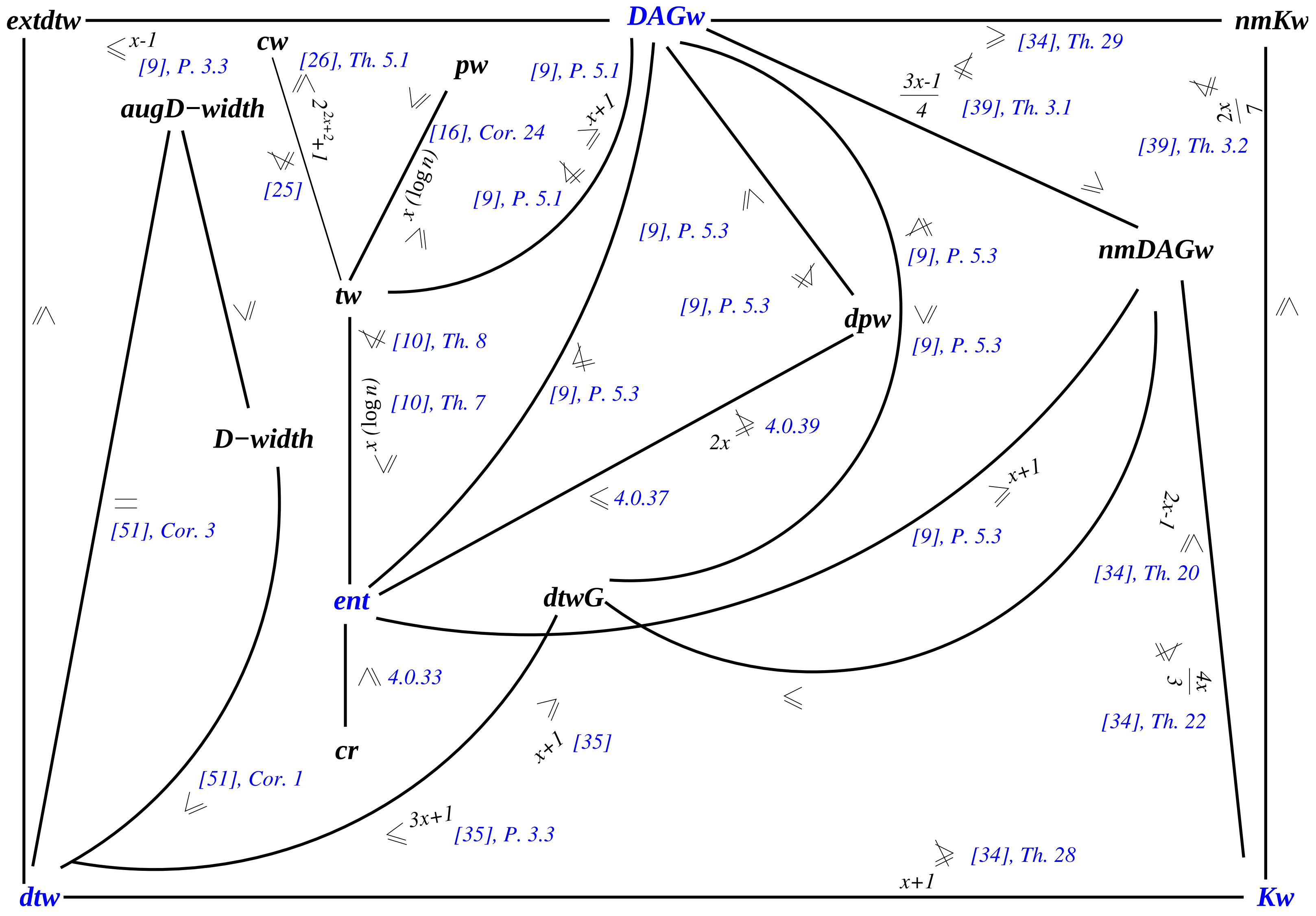

Figure A.1: Relations between graph complexity measures of graphs.