Characterizations and Algorithms for Special Digraph Classes

Inaugural-Dissertation

zur Erlangung des Doktorgrades der Mathematisch-Naturwissenschaftlichen Fakultät der Heinrich-Heine-Universität Düsseldorf

Vorgelegt von

Dominique Komander

aus Düsseldorf

16th December 2021

Aus dem Institut für Informatik
Mathematisch-Naturwissenschaftlichen Fakultät
Heinrich-Heine-Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät
Heinrich-Heine-Universität Düsseldorf

Berichterstatter:

1. PD Dr. Frank Gurski

2. Prof. Dr. Egon Wanke

Tag der mündlichen Prüfung: 01.03.2022

Acknowledgements

First, I would like to thank my supervisor and academic mentor PD Dr. Frank Gurski for all the support and teaching. As well, I would like to thank my second supervisor Prof. Dr. Egon Wanke for his good advices. In addition, I would like to thank Frank Gurski and Van Bang Le for the very useful discussions about directed versions of distance-hereditary graphs. Since parts of the research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation, I would also like to thank the DFG for the funding. And finally, I would like to thank my wonderful parents and my beloved sister who are my biggest fans and supporters, and at the same time, what I value most in life.

Abstract

The investigation of directed graph classes is highly motivated by the existence of NP-hard problems on directed graphs, which can be solved faster when restricted to special directed graph classes. But besides designing algorithms for hard problems, there are some open spots in the characterization of directed graph classes that are already solved for undirected graphs. The objective of this work is to expand the research in the field of directed graph classes and show on several examples how we can use the structures of special directed graphs to solve NP-hard problems, in some cases even in linear time. We deepen the understanding of several directed graph classes and show a new directed version of distance-hereditary graphs, namely twin-distance-hereditary graphs, fit them into the hierarchy and characterize them by a set of forbidden induced subdigraphs. We show some properties of this new directed graph class which are algorithmical useful for solving problems on this class. Furthermore, we study some directed graph parameters, compare them on semicomplete digraphs, and show an efficient computation of some directed width parameters on recursive defined directed graph classes as, e.g., directed co-graphs, and related classes. Moreover, we focus on different NP-hard digraph problems, namely the subset sum problem with (weak) digraph constraints, the directed Steiner path cover problem and several vertex and arc colorings on directed and oriented graphs. We show how to solve these problems on various recursive directed graph classes using dynamic programming by exploiting the underlying tree structure of the recursive digraph classes. Doing so, we achieve more efficient algorithms, in some cases we even show linear time solutions. Thus, with this work we take another small step to expand research in the rather young field of directed graph classes.

Contents

1 Introduction

Graphs are frequently used to model networks, problem settings or relations. These problems can be connectivity or network flow problems. In such a graph we have vertices which represent certain objects and arcs or edges that model the relation between the different objects. However, there are many cases in which this does not suffice as the structure of the problem is more complex. When modeling an infrastructure network for example, especially in urban areas we have many one-way streets that cannot be passed in both directions. Also, when modeling time sequences we need a model which represents that one event happens after an other, such that we need an order within the objects. In modeling relationships, there also could be one-way connections. Probably, every student at the university knows the director, but conversely she does not know every student at the university. This is how directed graphs come into play. In a directed graph, or digraph for short, we have vertices as our main elements, and directed edges (which are also called arcs), that set the vertices in relation to each other.

There are several NP-hard problems on directed graphs, such as different coloring problems, the directed Hamiltonian path problem or the directed path-width problem. At this point, the analysis of certain directed graph classes becomes interesting. Since NP-hard problems are not solvable in polynomial time, assuming that P$\neq$NP holds, we try to come closer to a solution by restricting our input digraph with several structural constraints. A set of digraphs that fulfills several restrictions, is called digraph class. Bang-Jensen and Gutin [BJG18] present an overview on several directed graphs classes such as tournaments or semicomplete digraphs and planar digraphs. Especially tournaments received significant attention in the past [CS11, KS15]. A frequently used digraph class is the set of directed acyclic graphs (DAGs), see [KN09, ALLM16]. In [VSR$^+$18] for example, a DAG is used to represent a vehicle sharability network in a special variation of the minimum fleet problem. Another possibility for restriction is to regard special rules with which the members of a special digraph class can be constructed. Therefore, we start with a very basic digraph as, e.g., with the one vertex digraph which consists of only one vertex. Then, we can follow certain instructions to build larger digraphs and insert edges. Such a construction of a digraph is called decomposition. An example of a digraph class with a recursive structure is the class of directed co-graphs. When extending this class by special rules, we come to extended directed co-graphs, which are a superclass of DAGs, the well-known class we mentioned before. Superclass means, that every directed acyclic graph can be constructed following instructions from the class of extended directed co-graphs.

The aim of this work is the deeper exploration of some directed graph classes and extend the research in this field. In addition, we show how to solve several NP-hard digraph problems more efficiently, when restricted to a special digraph class. But how could hard problems get easier of a sudden? We achieve this by exploiting the restrictions on the input digraphs and develop algorithms that use these structures to find a solution without checking every possible subsolution or combinations of those, which is often the reason for the high complexity of the problem. Using these restrictions, the input digraphs have a special tree-structure or it is possible to build a tree-like decomposition in a reasonable amount of time. On this tree-structure it is possible to work with dynamic programming, such that there are less subsolutions which must be considered within the computation. This is how we reduce the complexity of the problem and save time. Among the NP-hard problems we consider several directed graph parameters, multiple coloring problems, directed Steiner Path covers (which are related to the well known Hamiltonian path problem) and subset sum problems with digraph constraints. Also we investigate some new digraph classes and characterize them in different ways. This can help e.g., in solutions for solving recognition problems.

This work is structured as follows. In the second chapter, we start with some necessary notation and basic definitions of digraphs, directed graph parameters and some common directed graph classes.

Afterwards, we continue with recursive digraph classes, which play an important role in this work. We start with definitions of several well-known undirected graph classes such as co-graphs, series-parallel graphs and distance-hereditary graphs and continue with several directed versions of these classes. We aim to deepen the understanding about the properties of these classes. Thus, we characterize the classes in different ways, e.g., by restricted graph parameters or forbidding a set of induced subdigraphs. In this sense there is extensive research on undirected graph classes, meanwhile the research in the directed classes is not as advanced. There already exist several useful definitions of directed co-graphs, especially the characterization by forbidding eight induced subdigraphs by Crespelle and Paul [CP06]. We show similar characterizations for sub- and superclasses of directed co-graphs, each one of them motivated by the corresponding undirected class. Nevertheless, the set of forbidden induced subdigraphs of these classes looks very different than the undirected ones. Furthermore, we investigate a digraph class that is motivated by distance-hereditary graphs [How77]. These graphs maintain their distance heredity property when deleting vertices and the corresponding edges from the graph. The same attempt was made by [LS10] called distance-hereditary digraphs. Unfortunately, this directed version of distance-hereditary graphs does not fulfill some other useful properties of the undirected class. Among others, there is no recursive structure by twins and pendant vertices which defines a digraph, which is very useful in the undirected version for several algorithms. Moreover, we do not have a bound of the famous graph parameter directed clique-width. To overcome this drawback, we introduce another version, the so-called twin-distance-hereditary digraphs (twin-dh digraphs). For this class we present a definition by several directed versions of twins and pendant vertices, a characterization by a set of forbidden induced subdigraphs and we show a tight upper bound for directed clique-width.

In the fourth chapter, we talk about some results on directed graph parameters on several digraph classes. While tree-width and clique-width are the most famous parameters in undirected graph theory, at least directed tree width is not such a clear winner on digraphs. While the definition of directed clique-width [CO00] is very straightforward, there are several attempts to transfer the tree-width concept to directed graphs. Among these there are directed tree-width [JRST01b], DAG-width [BDHK06] or Kelly-width [HK08], as well as measures like directed path-width which are related to the well-known path-width for undirected graphs. And those are not all of them. The corresponding decision problems, whether a digraph has width k for some integer k, are at least NP-hard in general. But computing or bounding these parameters is very useful for developing parameterized algorithms. We give a comparison of some of these parameters for the directed graph class semicomplete digraphs, which are digraphs for which the underlying undirected graph is a clique. Especially this digraph class is interesting in this context, since the undirected parameters are often neither useful nor meaningful on a clique. However, directed parameters are not as trivial on semicomplete digraphs, which due to the different directions have a much more complex structure than an undirected clique. For directed co-graphs we show that we can compute some directed graph parameters with quite straightforward formulas using the construction rules. For some parameters, this is even possible on extended directed co-graphs, such that the results are transferable to the previously mentioned twin-dh digraphs.

In Chapter 5, 6 and 7 we show how to solve certain digraph problems on several digraph classes.

More concretely, in Chapter 5 we investigate the subset sum problem with (weak) digraph constraints (SSG and SSGW) on directed co-graphs and on series-parallel digraphs. Within the well-known subset sum problem we try to fit a subset of maximum size of items into a given capacity. Additionally, we have to fulfill certain digraph constraints. Since both problems are NP-hard even on oriented co-graphs as well as on minimal series-parallel digraphs (msp-digraphs), we give pseudo-polynomial[1] solutions. For these solutions, we use the recursive structure and compute the subsolutions along the decomposition of the digraph classes.

Further on, in Chapter 6 we regard the directed Steiner path cover problem. The (directed) Steiner path problem is a special variant of the (directed) Steiner tree problem in which the so-called terminal vertices must lie on one path of minimal cost. Since a (directed) Steiner path not always exists, it makes sense to look at a directed Steiner path cover. Given a directed graph G and a set of so-called terminal vertices, the problem is to find a directed Steiner path cover for G, which is a set of vertex-disjoint simple directed paths, that contain all terminal vertices as well as some non-terminal vertices if needed. The size of a directed Steiner path cover is defined as the number of Steiner paths, while the cost of a directed Steiner path cover is the minimum number of Steiner vertices in a directed Steiner path cover of minimum size. The associated decision problem is NP-hard such that we consider it on directed co-graphs, for which there is a linear time solution using the recursive structure and the forbidden induced subdigraphs.

In Chapter 7 we then talk about the coloring of digraphs. Although directed coloring problems such as oriented vertex and arc coloring as well as acyclic colorings are NP-

[1]Pseudo-polynomial means polynomial in some numerical value of the input instead of the length of the input.

hard problems, we provide tight upper bounds and linear time solutions on several recursive digraph classes such as minimal series-parallel digraphs and edge series-parallel digraphs (esp-digraphs), acyclic transitive digraphs and (oriented) directed co-graphs. The corresponding problem in acyclic coloring is called the Dichromatic number problem which is obviously motivated by one very fundamental problem in graph theory which is the Chromatic number problem. As already mentioned it is an NP-hard problem. Thus, we give a parameterized algorithm for solving this problem with respect to the parameter directed clique-width. This implies that there exists a polynomial time solution on digraphs of bounded directed clique-width, like for example on twin-dh digraphs, which we invented in the third chapter.

2 Basic Definitions

2.1 Notations

In this chapter we start with some terminology and notations which we use within this work. As a reference for definitions around digraphs we use Bang-Jensen and Gutin [BJG09]. We consider exclusively graphs without loops or multi-edges if not explicitly stated.

A graph G is a pair $G = (V, E)$ of vertices and edges $E \subseteq \{u, v\} \mid u, v \in V, u \neq v\}$. Correspondingly, a directed graph, digraph for short, is a pair $G = (V, E)$ of vertices and directed edges or arcs $E \subseteq \{(u, v) \mid u, v \in V, u \neq v\}$. At some points we use the following notation for edges and vertices for sake of simplicity: For a (directed) graph G, we denote by $V(G)$ the vertex set and $E(G)$ the edge set of G.

Let $und(G)$ be the underlying undirected graph of a digraph G, which arises by replacing every edge (u, v) by $\{u, v\}$ and deleting multi-edges. If (u, v) and (v, u) exist both, we call them *bidirectional edges* or *symmetric edges*. If an edge is not symmetric we call it *asymmetric*. The spanning subdigraph with only symmetric arcs is denoted by $sym(G)$, which is the symmetric part of digraph G. We define $asym(G)$ as the asymmetric part, respectively. There are different ways to give an undirected graph G a special orientation such that it becomes digraph D, see [BJG09]. We illustrate the different occasions in Figure 2.1. If we replace every edge $\{u, v\}$ of $E(G)$ by

- one of the arcs (u, v) and (v, u), we denote D as an *orientation* of G. Every digraph D that can be obtained by an orientation of an undirected graph G is called an *oriented graph*.

- one or both of the arcs (u, v) and (v, u), we denote D as a *biorientation* of G. A digraph D that we get by a biorientation of an undirected graph G is called a *bioriented graph*.

- both arcs (u, v) and (v, u), we denote D as a *complete biorientation* of G. Since in this case D is well defined by G we also denote it by $\overleftrightarrow{G}$. A digraph D that we obtain by a complete biorientation of an undirected graph G is called a *complete bioriented graph*.

A (di)graph $G' = (V', E')$ is a sub(di)graph of $G = (V, E)$ if we have $V' \subseteq V$ as well as $E' \subseteq E$. A vertex set $V' \subseteq V(G)$ *induces* a graph G', if $E(G') \subseteq E(G)$ and for every edge in G it holds that if $u, v \in V'$ then $(u, v) \in E(G)$ implies $(u, v) \in E(G')$. Then G' is called an induced sub(di)graph of G. For short we define with $G[V']$ the sub(di)graph induced by the

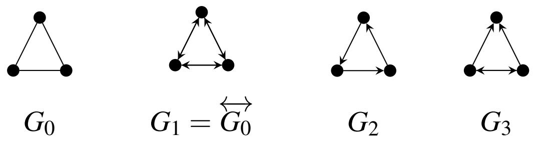


Figure 2.1: An undirected graph G_0, its complete biorientation G_1, G_2 which is an orientation of G_0 and G_3 which is a biorientation of G_0.

vertex set V'. Looking at a (di)graph class F we define $Free(F)$ as the set of all (di)graphs G such that there is no induced sub(di)graph of G isomorphic to a (di)graph of F.

An *independent set* in a (di)graph G is a subset $I \subseteq V(G)$ such that none of the vertices $v,u \in I$ are connected by an edge. In contrast, a *clique* in a graph G is a subset $C \subseteq V(G)$ such that for every vertices $v,u \in C$ with $u \neq v$ are connected by an edge. A *directed clique* in a digraph G is a subset $C' \subseteq V(G)$ such that for every vertices $v,u \in C'$ with $u \neq v$ are connected by bidirectional edges.

The *neighborhood* of a vertex v in an undirected graph G is denoted by $N(v) = \{u \mid \{u,v\} \in E(G)\}$. For a digraph G we distinguish between the set of predecessors $N_G^-(v) = \{u \mid (v,u) \in E(G)\}$, called *in-neighbors*, and the set of successors $N_G^+(v) = \{u \mid (v,u) \in E(G)\}$, called *out-neighbors*. A vertex is a *source* if it has no in-neighbors and a *sink* if it has no out-neighbors. The *outdegree* of v denoted by $\text{outdegree}_G(v) = |N_G^+(v)|$ is the number of successors of v and the *indegree* of v is the number of predecessors denoted by $\text{indegree}_G(v) = |N_G^-(v)|$. For the neighbors as well as for the degrees we omit indices if the graph under consideration is clear from the context. The *maximum outdegree* is denoted by $\Delta^+(G) = \max_{v \in V} \text{outdegree}(v)$ while the *maximum indegree* is $\Delta^-(G) = \max_{v \in V} \text{indegree}(v)$. The maximum (vertex) degree of digraph G is denoted by $\Delta(G) = \max_{v \in V(G)}\{\text{outdegree}(v) + \text{indegree}(v)\}$ and the *maximum semidegree* by $\Delta^0(G) = \max\{\Delta^-(G), \Delta^+(G)\}$.

For a digraph G a *strongly connected component* is an induced subdigraph H of G such that for all vertices $u,v \in V(H)$, there is a directed walk from u to v in H as well as a directed walk from v to u in H. A *strong component* of G is a maximal strongly connected component of G, i.e., a strongly connected component H of G such that there is no vertex $v \in V(G) \setminus V(H)$ such that the induced subdigraph of G generated by $V(H) \cup \{v\}$ is a strongly connected component of G. Notice that, all strong components of a digraph are vertex-disjoint. A digraph G is *weakly connected* if $und(G)$ is connected, such that a *weakly connected component* of G is a maximal subdigraph, such that the corresponding underlying graph is connected (maximal in terms of non-extensibility). A vertex v in digraph G is called a *bioriented leaf* if there is $(u,v),(v,u) \in E(G)$ and if v is a leaf in $und(G)$, such that in $und(G)$ it holds that $|N_{und(G)}(v)| = 1$. A digraph G is *transitive* if for every two edges $(u,v) \in E(G)$ and $(v,w) \in E(G)$ it holds that also $(u,w) \in E(G)$. The digraph $tc(G)$ is the *transitive closure* of G if $V(tc(G)) = V(G)$ and for two distinct vertices u,v there is an edge $(u,v) \in E(tc(G))$ if and only if vertex v is reachable from u in digraph G. A (di)graph G is *bipartite* if we can divide $V(G)$ into two distinct possibly empty independent sets $A \subseteq V(G)$ and $B \subseteq V(G)$. For an illustration of an example of an oriented bipartite graph see Figure 2.2.

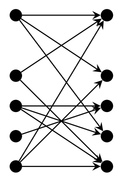


Figure 2.2: An oriented bipartite graph.

We continue with members of some well known (di)graph classes where n is the number of vertices of the respective (di)graph.

- An oriented path is $\overrightarrow{P_n} = (\{v_1,\dots,v_n\},\{(v_1,v_2),\dots,(v_{n-1},v_n)\})$ with $n \geq 2$.
- A path is $P_n = (\{v_1,\dots,v_n\},\{\{v_1,v_2\},\dots,\{v_{n-1},v_n\}\})$ with $n \geq 2$.
- An oriented cycle is $\overrightarrow{C_n} = (\{v_1,\dots,v_n\},\{(v_1,v_2),\dots,(v_{n-1},v_n),(v_n,v_1)\})$ with $n \geq 2$.
- A cycle is $C_n = (\{v_1,\dots,v_n\},\{\{v_1,v_2\},\dots,\{v_{n-1},v_n\},\{v_n,v_1\}\})$ with $n \geq 2$.
- A bidirectional complete digraph is $\overleftrightarrow{K_n} = (\{v_1,\dots,v_n\},\{(v_i,v_j) \mid 1 \leq i \neq j \leq n\})$ with $n \geq 1$. This is also called bioriented clique.
- A complete graph is $K_n = (\{v_1,\dots,v_n\},\{\{v_i,v_j\} \mid 1 \leq i \neq j \leq n\})$ with $n \geq 1$. Thus, the vertices of these graphs form a clique.
- An oriented complete bipartite digraph is $\overrightarrow{K_{n,m}} = (\{v_1,\dots,v_n,w_1,\dots,w_m\},\{(v_i,w_j) \mid 1 \leq i \leq n, 1 \leq j \leq m\})$.
- A complete bipartite graph is $K_{n,m} = (\{v_1,\dots,v_n,w_1,\dots,w_m\},\{\{v_i,w_j\} \mid 1 \leq i \leq n, 1 \leq j \leq m\})$.
- An edgeless (di)graph is $I_n = (\{v_1,\dots,v_n\},\{\})$.

A *DAG* (directed acyclic graph) is a digraph that does not contain a $\overrightarrow{C_n}$, with $n \geq 2$, as subdigraph. A tree is a connected undirected graph without cycles while a forest is the union of trees, such that forests are the class of cycle free undirected graphs. We have a distinguished *root* in a possibly directed tree T and a vertex v with $|N_{und(T)}(v)| = 1$ is called *leave*. An *oriented forest (tree)* is any orientation of a forest (tree). The class of oriented trees is denoted by OT. An *out-tree* (*in-tree*) is an orientation of a tree with a root with indegree (outdegree) zero such that all arcs are directed away from (to) the root. In some literature an out-tree is called an *arborescence*. We have a deeper look into directed graph classes and their characterization in Chapter 3.

Vertex v is *reachable* from vertex u in G, if there exists a directed path in G from u to v. Further, we call G *odd cycle free*, if it contains no $\overrightarrow{C_n}$ for an odd number n as subdigraph. We call a digraph *even*, if it contains a directed cycle of even total weight for every 0-1-weighting of the edges. A vertex $v \in V$ is *out-dominating* (*in-dominated*) if it is adjacent to every other vertex in V and is a source (a sink, respectively). We call a linear order of the vertices of a digraph such that for every edge (u,v), vertex u is before vertex v in the order a *topological ordering* of this digraph.

For a directed graph $G = (V,E)$ its *complement digraph* is defined by

$$\text{co-}G = (V, \{(u,v) \mid (u,v) \notin E, u,v \in V, u \neq v\})$$

or $\overline{G}$ for short and its *converse digraph* is defined by

$$G^c = (V, \{(u,v) \mid (v,u) \in E, u,v \in V, u \neq v\}).$$

The complement of an undirected graph is

$$\text{co-}G = (V, \{\{u,v\} \mid \{u,v\} \notin E, u,v \in V, u \neq v\}).$$

For a digraph class X we define by co-$X = \{\text{co-}G \mid G \in X\}$. For a digraph G and an integer d let dG be the disjoint union of d copies of G.

The notations and results below are from [KL15, Chapter 2] where they are introduced for undirected graphs. They can be transferred, as they also hold for directed graphs. This part can also be found in [GKR21c].

Classes of (di)graphs which are closed under taking induced sub(di)graphs are called *hereditary*. Given a (di)graph class F, then Free(F) is the set of all (di)graphs G such that no induced sub(di)graph of G is isomorphic to a (di)graph in set F.

Theorem 2.1.1 ([KL15])**.** *For a class of (di)graphs X, this class is hereditary if and only if there is a set F for that holds that Free$(F) = X$.*

A (di)graph G is a *minimal forbidden induced sub(di)graph* for a hereditary class X if G does not belong to X and if also every proper induced sub(di)graph of G is a member of X. Given a hereditary (di)graph class X let Forb(X) be the set of all minimal forbidden induced sub(di)graphs of X.

Theorem 2.1.2 ([KL15])**.** *For a hereditary class of (di)graphs X it holds that $X =$ Free(Forb(X)), and the set Forb(X) is unique and of minimum size.*

Theorem 2.1.3 ([KL15])**.** *It holds that Free$(F_1) \subseteq$ Free(F_2) if and only if for every (di)graph $G \in F_2$ there exists a (di)graph $H \in F_1$ where H is an induced sub(di)graph of G.*

Lemma 2.1.4 ([KL15])**.** *Let $X =$ Free(F_1) and $Y =$ Free(F_2) be hereditary (di)graph classes. Then, it holds that $X \cap Y =$ Free$(F_1 \cup F_2)$ and co-$X =$ Free(co-F_1).*

Observation 2.1.5 ([GKR21c])**.** *Let G be a digraph with $G \in$ Free(X) for a hereditary class of digraphs Free(X). Further, there exists a digraph $X^* \in X$ such that every biorientation of und(X^*) is in Free(X). Then, it holds that und$(G) \in$ Free(und(X^*)).*

Observation 2.1.6 ([GKR21c])**.** *Let G be a digraph such that und$(G) \in$ Free(X) for a hereditary class of digraphs Free(X), then it holds for all $X^* \in X$ and all biorientations $b(X^*)$ of X^* that $G \in$ Free$(b(X^*))$.*

We continue with a short excursion into computational complexity theory. In complexity theory we classify problems into certain *complexity classes* such that two problems in a class have the same complexity in the sense that they have the same upper bound. Two of the major classes here are *P* and *NP*, especially as it is still not proven if *P* is a proper subset of *NP* or if the two classes are equal. All decision problems in *P* can be solved in polynomial time while decision problems in *NP* can be solved in non-deterministic polynomial time (in polynomial time by a non-deterministic Turing machine). As already mentioned in the introduction, we assume that $P \neq NP$ applies. For a deeper look into complexity theory we refer to the famous work of Garey and Johnson [GJ79].

In complexity theory there is a part called parameterized complexity. Within parameterized complexity we classify problems according to their hardness with respect to different parameters of the input. By this parameterizations we want to solve NP-hard problems efficiently in practice. Intuitively, we want to exchange the high number of vertices within the exponent by a smaller fixed parameter. XP is the class of all parameterized problems which can be solved by algorithms that are polynomial if the parameter is considered to be a constant. FPT is the class of all parameterized problems that can be solved by algorithms that are exponential only in the size of a fixed parameter k while polynomial in the size of the input x, thus in time $f(k)|x|^{O(1)}$ for a computable function f. For more details about both classes see [DF13]. Parameters that can be used in this field are presented in the next Section.

2.2 Graph Parameters

A *(directed) graph parameter* of a (di)graph G is a function α that maps from (di)graph G to an integer. We call two graph parameters α and β *equivalent*, if there exist functions f, g such that for every (directed) graph $\alpha(G) \leq f(\beta(G))$ and $\beta(G) \leq g(\alpha(G))$. A (directed) width parameter or *(digraph) width measure* is as well a graph parameter which often has an underlying structure or decomposition.

In this section we give a more formal definition for some directed width parameters especially by so-called decompositions. Many of these tree-width inspired directed graph parameters correspond to different variants of *cops and robber games*, which we also define later in this section. The content from this section is mostly from [GKR21b] and [GKRW21].

Undirected path-width and tree-width

The concept of tree-width was developed, among others, by Robertson and Seymour [RS86]. The idea of tree-width is straightforward as it describes how tree-like a graph is. Thus, forests are exactly the graphs with tree-width 1.

Definition 2.2.1 (Tree-width)**.** For a graph G a *tree-decomposition* is a tuple $(\mathcal{X}, T)$ where T is a tree and $\mathcal{X} = \{X_u \mid X_u \subseteq V(G), u \in V(T)\}$ is a set of subsets of $V(G)$ which satisfies the following conditions:

1. $\bigcup_{u \in V(T)} X_u = V(G)$.

2. For two distinct vertices $u, v \in V(G)$ and every edge $\{u, v\} \in E(G)$ there is a vertex $t \in V(T)$ with both vertices $u, v \in X_t$.
3. For every vertex $v \in V(G)$ the corresponding tree vertices $u \in T$ with $v \in X_u$ induce a connected subtree.

The *width* of a tree-decomposition $(\mathcal{X}, T)$ is

$$\max_{u \in V(T)} |X_u| - 1.$$

The *tree-width* of a graph G, denoted by $\text{tw}(G)$, is the minimum width of all possible tree-decompositions for G.

The decision whether a graph G has tree-width of at most k is an NP-complete problem [ACP87]. If k is a fixed constant, the recognition of a graph of tree-width k as well as the construction of a corresponding tree-decomposition of at most width k is possible in linear time [Bod96].

If we restrict the tree in a tree-decomposition to be a path, we get the definition of path-width. We take an equivalent definition from [RS83].

Definition 2.2.2 (Path-width)**.** A sequence of subsets $(X_1, \dots, X_r)$ with $X_i \subseteq V(G)$ and $1 \leq i \leq r$, is a *path-decomposition* for graph G if the following conditions are satisfied.

1. $X_1 \cup \cdots \cup X_r \;=\; V(G)$.
2. For every edge $\{u, v\} \in E(G)$ there exist a X_i, $1 \leq i \leq r$, such that $u, v \in X_i$.
3. If $u \in X_i$ and $u \in X_j$ for an $u \in V(G)$ and two indices i, j with $i \leq j$, then $u \in X_\ell$ for all indices ℓ with $i \leq \ell \leq j$.

The *width* of a path-decomposition is

$$\max_{1 \leq i \leq r} |X_i| - 1.$$

The *path-width* of G, denoted by $\text{pw}(G)$ is the minimum width over all possible path-decompositions for G.

The decision whether a graph G has path-width of at most k is an NP-complete problem [ACP87].

Directed path-width

The notion of directed path-width was introduced by Reed, Seymour, and Thomas in the 1990s and leads a restriction of directed tree-width which is defined by Johnson, Robertson, Seymour, and Thomas in [JRST01b]. The following subsections are taken from [GKR21b].

Definition 2.2.3 (Directed path-width)**.** A *directed path-decomposition* of a directed graph G is a sequence $(X_1, \dots, X_r)$ of subsets of $V(G)$, called *bags*, such that the following three conditions hold.

(dpw-1) $X_1 \cup \cdots \cup X_r = V(G)$.

(dpw-2) For each $(u,v) \in E(G)$ there is a pair $i \leq j$ such that $u \in X_i$ and $v \in X_j$.

(dpw-3) If $u \in X_i$ and $u \in X_j$ for an $u \in V(G)$ and two indices i, j with $i \leq j$, then $u \in X_\ell$ for all indices ℓ with $i \leq \ell \leq j$.

The *width* of a directed path-decomposition $\mathcal{X} = (X_1, \ldots, X_r)$ is

$$\max_{1 \leq i \leq r} |X_i| - 1.$$

The *directed path-width* of G, d-pw(G) for short, is the smallest integer k such that there is a directed path-decomposition of G of width k.

The following illustrations which exemplify some of the digraph parameters are taken from [GKR21b].

Example 2.2.4. In Figure 2.4 we show an illustration of a directed path-decomposition for a digraph G, see Figure 2.3.

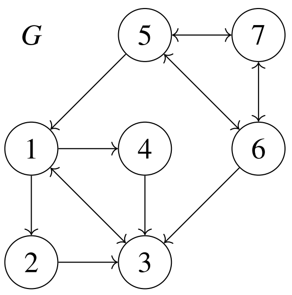


Figure 2.3: A digraph G which is used in Examples 2.2.4, 2.2.9, 2.2.15 and 2.2.19 to illustrate a decompositions of the corresponding width measures.

Every DAG has directed path-width 0. Furthermore, several types of tree-like digraphs have directed path-width 1 [GR19b].

There are several definitions of directed path-width, e.g. there exists a cops and robber variant for directed path-width [Bar06]. Additionally, there is the well known concept of the directed vertex separation number which is equivalent to the directed path-width [YC08].

Definition 2.2.5 (Directed vertex separation number, [YC08]). Let G be a digraph and $L : V(G) \to \{1, \ldots, |V(G)|\}$ a *linear layout* of G. Let further $DV_L(i) = \{x \in V(G) | \exists y \in V(G)$ such that $(y,x) \in E(G)$ and $L(x) \leq i$ and $L(y) > i\}$. The *directed vertex separation* of G w.r.t. L ($dvs_L(G)$) is defined as $dvs_L(G) = \max\{|DV_L(i)| : 1 \leq i \leq |V(G)|\}$. The directed vertex separation of G is the minimum over all layouts $dvs(G) = \min\{dvs_L(G) :$ L is a linear layout of $G\}$.

Lemma 2.2.6 ([YC08]). *Let G be a digraph, then d-pw$(G) \leq pw(und(G))$.*

The proofs shown in [YC08] use the notation of directed vertex separation number.

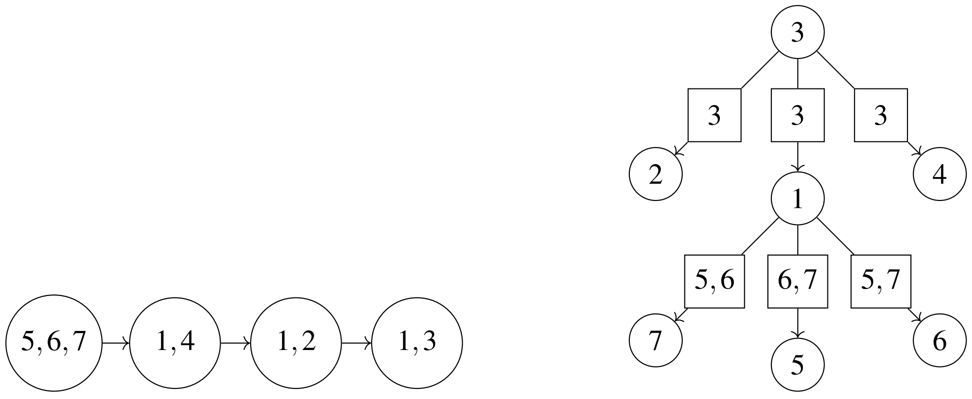


Figure 2.4: An illustration of a directed path-decomposition (left) of width 2 and an arboreal tree-decomposition (right) of width 3 for digraph G, see Figure 2.3. In the aboreal tree-decomposition, the X_e set is represented as a square on the respective edge e, while the W sets are represented as round vertices of the out-tree. For sake of simplicity the set braces are omitted.

Lemma 2.2.7 ([Bar06]). *Let G be a complete bioriented graph, then d-$pw(G) = pw(und(G))$.*

The proof is straightforward since for a complete bioriented graph G a directed path-decomposition of width k is a path-decomposition of width k for $und(G)$, and vice versa.

Directed tree-width

We use the directed tree-width introduced by Johnson et al. [JRST01b, JRST01a]. There are different directed tree-width definitions such as forbidding and allowing empty sets W_r in [JRST01a, JRST01b], using sets W_r of size one only for the leaves of T in [Ree99] and using strong components within (dtw-2) in [DES14, Chapter 6]. Further, in works of Courcelle et al. [Cou18, CE12, CO00] the directed tree-width of a digraph G is defined by the tree-width of the underlying undirected graph.

A *directed walk* in digraph $G = (V,E)$ is an alternating sequence $W = (u_1, e_1, u_2, e_2, u_3, \ldots, e_{k-1}, u_k)$ of vertices $v_i \in V$, $1 \leq i \leq k$, and edges $e_i \in E$, $1 \leq i \leq k-1$, such that $e_i = (u_i, u_{i+1})$, $1 \leq i \leq k-1$, if the vertices of the directed walk W are distinct, then W is a *directed path*.

For two vertices u, v of an out-tree T the notation $u \leq v$ means that there is a directed path on ≥ 0 arcs from u to v and $u < v$ means that there is a directed path on ≥ 1 arcs from u to v.

Let $G = (V,E)$ be a digraph and $Z \subseteq V$. The digraph $G[V - Z]$ which is obtained from G by deleting Z will be denoted by $G - Z$. A vertex set $S \subseteq V \setminus Z$ is *Z-normal* if there is no directed walk in $G - Z$ with first and last vertices in S that uses a vertex of $G - (Z \cup S)$. In other words, a set $S \subseteq V$ is Z-normal, if every directed walk which leaves and again enters S must contain a vertex from Z.

As $W_{\geq v} = \bigcup_{\tilde{v} \geq v} W_{\tilde{v}}$ we define the union of the sets $W_{\tilde{v}}$ of all (indirect) successors $\tilde{v}$ of v including W_v.

Definition 2.2.8 (directed tree-width)**.** A (-n *arboreal) tree-decomposition* (or directed tree-decomposition) of a digraph $G = (V_G, E_G)$ is a triple $(T, \mathcal{X}, \mathcal{W})$. Here $T = (V_T, E_T)$ is an out-tree, $\mathcal{X} = \{X_e \mid e \in E_T\}$ and $\mathcal{W} = \{W_r \mid r \in V_T\}$ are sets of subsets of V_G, such that the following two conditions hold.

(dtw-1) $\mathcal{W} = \{W_r \mid r \in V_T\}$ is a partition of V_G into non-empty subsets.[1]

(dtw-2) For every $(u,v) \in E_T$ the set $W_{\geq v}$ is $X_{(u,v)}$-normal.

The *width* of a (-n arboreal) tree-decomposition $(T, \mathcal{X}, \mathcal{W})$ is

$$\max_{r \in V_T} |W_r \cup \bigcup_{e \sim r} X_e| - 1.$$

Here $e \sim r$ means that r is one of the two vertices of arc e. The *directed tree-width* of G, d-tw(G) for short, is the smallest integer k such that there is a (-n arboreal) tree-decomposition $(T, \mathcal{X}, \mathcal{W})$ of G of width k.

Example 2.2.9. In Figure 2.4 we show an illustration of an arboreal tree-decomposition for a digraph G, see Figure 2.3.

Every DAG has directed tree-width 0. Furthermore, several types of tree-like digraphs have directed tree-width 1 [GR19b].

Remark 2.2.10 (Z-normality). Notice that, our above used definition of Z-normality slightly differs from the following definition in [JRST01b] where S and Z are disjoint. A vertex set $S \subseteq V \setminus Z$ is *Z-normal*, if there is no directed walk in $G - Z$ with first and last vertices in S that uses a vertex of $G - (Z \cup S)$. Or, a set $S \subseteq V \setminus Z$ is Z-normal, if every directed walk which leaves and again enters S must contain a vertex from Z, see [BJG09]. Every set $S \subseteq V \setminus Z$ which is is Z-normal w.r.t. the definition in [JRST01b] is also Z-normal w.r.t. our definition. Further, a set $S \subseteq V$ which is Z-normal w.r.t. our definition, is also $Z \setminus S$-normal w.r.t. the definition in [JRST01b]. Thus, the directed tree-width of a digraph is equal for both definitions of Z-normality.

Lemma 2.2.11 ([JRST01b])**.** *Let G be a digraph, then it holds that d-tw$(G) \leq$ tw$(und(G))$.*

Lemma 2.2.12 ([JRST01b])**.** *Let G be a complete bioriented graph, then it holds that d-tw$(G) =$ tw$(und(G))$.*

Directed cops and robbers games

We continue with a small insertion as we now slide in with a notion of cops and robber games. This part is taken from [GKRW21].

A *cops and robbers game* on a (directed) graph is a pursuit-evasion game with two teams of players, the cops, which can move without any restriction to every vertex with their helicopters and the robbers moving from vertex to vertex along the arcs/edges of a graph. The

[1] A remarkable difference to the undirected tree-width (Definition 2.2.1) is that the bags, which are the sets W_r here, have to be disjoint and non-empty.

cops try to “catch” the robbers by moving onto the vertices where the robbers are positioned, while the robbers try to evade this capture.

Let $G = (V,E)$ be a directed graph with one robber and a set of cops. A position in the game is a pair (C,r) where $C \subseteq V$ is the current position of the cops and $r \in V$ is the current position of the robber. Initially, there is no cop on the graph, i.e., $C_0 = \emptyset$ and in the first round the robber can choose a start position r_0. In every round $i+1$, (C_i,r_i) is the current position of the cops and robber. The game is then played as follows: The cops announce their new position C_{i+1}. Then the robber can chose any vertex r_{i+1} as a new position, that is reachable from r_i in the graph $G-(C_i \cap C_{i+1})$. There are two variations of reachability: In strong component searching, the robber can move to every vertex in the same strong component of $G-(C_i \cap C_{i+1})$. In reachability searching, the robber can move to any vertex r_{i+1} such that there is a directed walk from r_i to r_{i+1}.

If $r_i \in C_i$ after any round i, then the cops capture the robber and win the game. Otherwise, the game never ends and the robber wins the game. Clearly, the game can always be won by the cops, by positioning a cop on every vertex of G. However, an interesting question is, how many cops are needed for a graph G, such that there is always a winning strategy for the cops.

By varying the rules, many different cops and robber games can be defined. The best known modification is, if the cops know the current robber position (visible CnR-Game) or do not know the current robber position (invisible CnR-Game). Another variant is a so-called inert robber: This robber is only allowed to move, if $r_i \in C_{i+1}$, i.e., the robber would be captured in the next round.

A winning strategy of the cops is called *robber monotone* if for every sequence of cop moves $C_1, C_2, \dots$ and all possible resulting moves of the robber, the strong components of the digraph without C_i are a non-increasing sequence [JRST01b].

There is a strong link between a variant of the cops and robber game and directed tree-width, which we defined in the previous subsection. Indeed, this link played an important role in finding the definition of directed tree-width in the first place.

Proposition 2.2.13 ([JRST01b])**.** *If G has directed tree-width of at most k, then $k+1$ cops have a robber monotone winning strategy in the visible strong component cops and robber game on G. If k cops have a winning strategy in this game, then the directed tree-width of G is at most $3k+2$.*

DAG-width

This part is taken from [GKRW21]. The DAG-width has been defined in [BDHK06, BDH$^+$12, Obd06] and can be intuitively understood as a measure for how DAG-alike a digraph is. The main difference between directed tree-width and DAG-width is that the separations in an arboreal decomposition only destroy strong connectivity, while those in a DAG-decomposition block all directed paths leaving the bags of a sub-DAG. Since directed separations are more restricted than strong separations, the model graph which is used for the decomposition needs to be relaxed from an arborescence to a DAG. We recall the definition of [BDHK06].

Let $G = (V_G, E_G)$ be an acyclic digraph. The partial order $\preccurlyeq_G$ on G is the reflexive, transitive closure of E_G. A source or root of a set $X \subseteq V_G$ is a $\preccurlyeq_G$-minimal element of X, that is, $r \in X$ is a root of X if there is no $y \in X$, such that $y \preccurlyeq_G r$ and $y \neq x$. Analogously, a sink or leaf of a set $X \subseteq V_G$ is a $\preccurlyeq_G$-maximal element.

Let $V' \subseteq V_G$, then a set $W \subseteq V_G$ *guards* V' if for all $(u,v) \in E_G$ it holds that if $u \in V'$ then $v \in V' \cup W$.

Definition 2.2.14 (DAG-width). A *DAG-decomposition* of a digraph $G = (V_G, E_G)$ is a pair $(D, \mathcal{X})$ where $D = (V_D, E_D)$ is a directed acyclic graph (DAG) and $\mathcal{X} = \{X_u \mid X_u \subseteq V_G, u \in V_D\}$ is a family of subsets of V_G such that:

(dagw-1) $\bigcup_{u \in V_D} X_u = V_G$.

(dagw-2) For all vertices $u, v, w \in V_D$ with $u \succcurlyeq_D v \succcurlyeq_D w$, it holds that $X_u \cap X_w \subseteq X_v$.

(dagw-3) For all edges $(u,v) \in E_D$ it holds that $X_u \cap X_v$ guards $X_{\succcurlyeq_v} \setminus X_u$, where $X_{\succcurlyeq_v} = \cup_{v \succcurlyeq_D w} X_w$. For any source u, $X_{\succcurlyeq_u}$ is guarded by $\emptyset$.

The *width* of a DAG-decomposition $(D, \mathcal{X})$ is the number

$$\max_{u \in V_D} |X_u|.$$

The *DAG-width* of a digraph G, $\text{dagw}(G)$ for short, is the smallest width of all possible DAG-decompositions for G.

It is straightforward that a DAG-decomposition where D is a path can also be seen as a directed path decomposition, as it meets the same conditions.

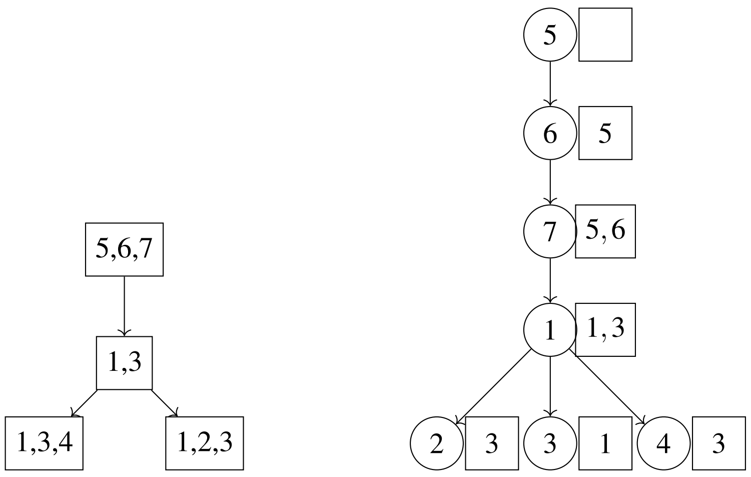


Figure 2.5: An illustration of a DAG-decomposition of DAG-width 3 (left) and a Kelly decomposition of width 3 (right) for digraph G, see Figure 2.3. In the Kelly decomposition, the round vertices represent the W sets, while the squares next to them represent the corresponding X sets. For sake of simplicity the set braces are omitted.

Example 2.2.15. In Figure 2.5 we show an illustration of a DAG-decomposition for a digraph G, see Figure 2.3.

One can restrict to a special structure of the decomposition, called nice DAG-decompositions from [BDH$^+$12]. We define $A \triangle B$ as the symmetric difference.

Definition 2.2.16 (Nice DAG-decomposition, [BDH$^+$12])**.** A DAG-decomposition $(D, \mathcal{X})$ of a digraph G is *nice*, if the following properties are fulfilled.

1. D has exactly one root r.
2. Every vertex in D has at most two successors.
3. If vertex d has two successors d' and d'', then it holds that $X_d = X_{d'} = X_{d''}$.
4. If vertex d has one successors d', then it holds that $|X_d \triangle X_{d'}| = 1$.

Lemma 2.2.17 ([BDH$^+$12])**.** *If digraph G has a DAG-decomposition of width k, it also has a nice DAG-decomposition of width k.*

The complexity of DAG-width has been studied in [AKR16]. They showed that there are digraphs on n vertices whose optimal DAG-decompositions have super-polynomially many bags w.r.t n. Thus, it has been shown that deciding whether the DAG-width of a given digraph is at most a given value is PSPACE-complete.

Kelly-width

We now come to Kelly-width, which has originally been introduced in [HK08]. This part is taken from [GKRW21]. The original definition of Kelly-width bears some resemblance to the definition of DAG-width, but it is more technical. Its definition is based on the existence of a special DAG. While a DAG-decomposition has one vertex set for every vertex of the decomposition, within a Kelly-decomposition there are two vertex sets for every vertex of the decomposition. In [HK08] it was conjectured that Kelly-width and DAG-width are indeed parametrically equivalent, but so far only one of the two relations has been shown [AKK$^+$15].

Definition 2.2.18 (Kelly-width)**.** A *Kelly decomposition* of a digraph $G = (V_G, E_G)$ is a triple $(\mathcal{W}, \mathcal{X}, D)$ where D is a directed acyclic graph, $\mathcal{X} = \{X_u \mid X_u \subseteq V_G, u \in V_D\}$ and $\mathcal{W} = \{W_u \mid W_u \subseteq V_G, u \in V_D\}$ are families of subsets of V_G such that:

(kw-1) $\mathcal{W}$ is a partition for V_G.

(kw-2) For all vertices $v \in V_G$, X_v guards $W_{\succcurlyeq v}$.

(kw-3) For all vertices $v \in V_G$, there is a linear order $u_1, \ldots, u_s$ on the successors of v such that for every u_i it holds that $X_{u_i} \subseteq W_i \cup X_i \cup \bigcup_{j<i} W_{\succcurlyeq u_j}$. Similarly, there is a linear order $r_1, r_2, \ldots$ on the roots of D such that for each root r_i it holds that $W_{r_i} \subseteq \bigcup_{j<i} W_{\succcurlyeq r_j}$.

The *width* of a Kelly decomposition $(\mathcal{W}, \mathcal{X}, D)$ is the number

$$\max_{u \in V_D} |X_u| + |W_u|.$$

The *Kelly-width* of a digraph G, denoted with $\mathrm{kw}(G)$, is the smallest width of all possible Kelly decompositions for G.

Example 2.2.19. In Figure 2.5 we show an illustration of a Kelly decomposition for a digraph G, see Figure 2.3.

Since the definition via a Kelly decomposition is not exactly intuitive, we introduce the directed elimination ordering, see [HK08]. Therefore we use the following notation.

Definition 2.2.20 (Directed Elimination Ordering)**.** Let $G = (V,E)$ be a digraph. A directed elimination ordering $\lhd$ on G is a linear ordering on V. For $\lhd = (v_0, v_1, \ldots, v_{n-1})$ we define

- $G_0^{\lhd} = G$.
- $G_{i+1}^{\lhd} = (V_{i+1}^{\lhd}, E_{i+1}^{\lhd})$ with $V_{i+1}^{\lhd} = V_i^{\lhd} \setminus \{v_i\}$ and $E_{i+1}^{\lhd} = \{(u,v) \mid (u,v) \in E_i^{\lhd} \text{ and } u,v \neq v_i \text{ or } (u,v_i),(v_i,v) \in E_i^{\lhd}, u \neq v\}$.

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

Lemma 2.2.21 ([HK08])**.** *For digraph G the following statements are equivalent:*

1. *G has Kelly-width at most $k+1$.*
2. *G has a directed elimination ordering of width $\leq k$.*

Kelly-width can also be characterized by a certain variant of cops and robber games.

Proposition 2.2.22 ([HK08])**.** *A digraph G has Kelly-width of at most $k+1$ if and only if $k+1$ cops have a winning strategy to capture an invisible and inert robber in the reachability searching game.*

(Directed) (linear) clique-width

For undirected graphs the clique-width [CO00] is one of the most important parameters. Clique-width measures how difficult it is to decompose the graph into a special tree-structure. It can also intuitively be understood as as measure of how different the neighborhoods inside a graph are. From an algorithmic point of view, only tree-width [RS86] is a more studied graph parameter. But clique-width is more general than tree-width since graphs of bounded tree-width have also bounded clique-width [CR05]. Meanwhile, tree-width can only be bounded by the clique-width under certain conditions [GW00]. Many NP-hard graph problems admit polynomial-time solutions when restricted to graphs of bounded tree-width or graphs of bounded clique-width. This part is from [GKR21a]. Just like clique-width, directed clique-width was defined by Courcelle and Olariu in [CO00].

Definition 2.2.23 (Directed clique-width [CO00])**.** The *directed clique-width* of a vertex labeled digraph G, d-cw(G) for short, is the minimum number of labels needed to define G using the following four operations:

1. Creation of a new vertex with label a (denoted by $\bullet_a$).
2. Disjoint union of two labeled digraphs G and H (denoted by $G \oplus H$).
3. Inserting an arc from every vertex with label a to every vertex with label b ($a \neq b$, denoted by $\alpha_{a,b}$).
4. Change every label a into label b (denoted by $\rho_{a \to b}$).

The *directed clique-width* of an unlabeled digraph $G = (V,E)$, d-cw(G) for short, is the smallest integer k, such that there is a mapping lab : $V \to \{1,\dots,k\}$ such that the labeled digraph (V,E,lab) has directed clique-width at most k.

An expression X built with the operations defined above using k labels is called a *directed clique-width k-expression*. Let digraph(X) be the digraph defined by k-expression X. By the given definition every graph of directed clique-width at most k can be represented by a tree structure, denoted as *k-expression-tree*. The leaves of the k-expression-tree represent the vertices of the digraph and the inner nodes of the k-expression-tree correspond to the operations applied to the subexpressions defined by the subtrees. Using the k-expression-tree many hard problems have been shown to be solvable in polynomial time when restricted to graphs of bounded directed clique-width [GHK+14, GWY16]. The following example which is partly from [KR21] shows an k-expression.

Example 2.2.24. A 3-expression for the $\overrightarrow{P_3}$ is

$$\alpha_{2,3}(\alpha_{1,2}(\bullet_1 \oplus \bullet_2) \oplus \bullet_3).$$

A 3-expression for the $\overrightarrow{P_4}$ is

$$\alpha_{2,3}((\rho_{2\to 1}(\alpha_{2,3}(\alpha_{1,2}(\bullet_1 \oplus \bullet_2)) \oplus \bullet_3)) \oplus \bullet_2).$$

The linear clique-width for undirected graphs was introduced in [GW05] as a parameter that restricts the clique-width to an underlying path-structure. Directed linear clique-width can be obtained, when the disjoint union operation is only allowed for one digraph and one labeled vertex, i.e., in the Definition 2.2.23, the graph H contains exactly one vertex, as also showed in Example 2.2.24.

Further directed width measures

The remaining part of this subsection is taken from [GKR21b]. The directed feedback vertex set number is probably the oldest of the measures considered here and was already considered by Karp [Kar72], where it is shown that the corresponding decision problem is NP-complete.

Definition 2.2.25 (Directed feedback vertex set number)**.** The *directed feedback vertex set number* of a digraph $G = (V,E)$, denoted by fvs(G), is the minimum cardinality of a set $S \subset V$ such that $G[V \setminus S]$ is a DAG.

We come to the directed feedback arc set. Finding the directed feedback arc set number is a very fundamental problem and has applications in layered graph drawing [EL89].

Definition 2.2.26 (Directed feedback arc set number)**.** The *directed feedback arc set number* of a digraph $G = (V,E)$, denoted by $\mathrm{fas}(G)$, is the minimum cardinality of a set $S \subset E$ such that $(V, E \setminus S)$ is a DAG.

The corresponding decision problem of computing the directed feedback arc set number is NP-hard since it is one of the 21 NP-complete problems of Karp, see [Kar72].

Cycle rank was introduced in [Egg63] and also appeared in [Coh68] and [McN69].

Definition 2.2.27 (Cycle rank)**.** The *cycle rank* of a digraph $G = (V,E)$, denoted by $\mathrm{cr}(G)$, is defined as follows.

- If G is acyclic, $\mathrm{cr}(G) = 0$.
- If G is strongly connected, then $\mathrm{cr}(G) = 1 + \min_{v \in V} \mathrm{cr}(G - \{v\})$.
- Otherwise, the cycle rank of G is the maximum cycle rank of any strongly connected component of G.

Results on the cycle rank can be found in [Gru12]. In this papers Gruber proved the hardness of computing cycle rank, even for sparse digraphs of maximum outdegree at most 2.

If we compare the definitions of the directed feedback vertex set number and the cycle rank the two width measures seem very similar. However, the values can differ significantly from each other for some digraphs. The reasons for the difference is that the cycle rank is about the maximum number of cycles in the biggest component while the directed feedback vertex set number is more about the number of all contained cycles in every component. Assume digraph $G = \overleftrightarrow{K_2} \oplus \overleftrightarrow{K_2} \oplus \overleftrightarrow{K_2} \oplus \overleftrightarrow{K_2}$, where $\oplus$ is the disjoint union of digraphs. Then, the cycle rank of G is 1, as we look at one component, while the directed feedback vertex set number is 4.

The DAG-depth of a digraph was introduced in [GHK$^+$09] motivated by tree-depth for undirected graphs, given in [NdM06].

For a digraph $G = (V,E)$ and $v \in V$, let G_v denote the subdigraph of G induced by the vertices which are reachable from v. The maximal elements in the partially ordered set $\{G_v \mid v \in V\}$ w.r.t. the digraph inclusion order (subdigraph) are the reachable fragments of G and will be denoted by $R(G)$. In the undirected case, reachable fragments coincide with connected components.

Definition 2.2.28 (DAG-depth)**.** Let $G = (V,E)$ be a digraph. The DAG-depth of G, denoted by $\mathrm{ddp}(G)$, is defined as follows.

- If $|V| = 1$, then $\mathrm{ddp}(G) = 1$.
- If G has a single reachable fragment, then $\mathrm{ddp}(G) = 1 + \min_{v \in V} \mathrm{ddp}(G - \{v\})$.
- Otherwise, $\mathrm{ddp}(G)$ equals the maximum over the DAG-depth of the reachable fragments of G.

We introduce a decomposition for DAG-depth, which is very similar to the one for cycle rank in [Gru12, McN69].

Definition 2.2.29 (Directed Elimination Forest)**.** A *directed elimination tree* for a digraph $G = (V,E)$ with $|R(G)| = 1$ reachable fragment is a rooted tree $T = (V_T, E_T)$ having the following properties.

1. $V_T \subseteq V \times 2^V$ and if $(x,X) \in V_T$, then $x \in X$.
2. The root of T is (v,V) for some $v \in V$.
3. If there is some vertex $(x,X) \in V_T$, then there is no vertex $(y,X) \in V_T$ for $x \neq y$.
4. If there is some vertex $(x,X) \in V_T$, and $G[X] - \{x\}$ has j reachable fragments $G_1 = (X_1,E_1),\ldots,G_j = (X_j,E_j)$, then (x,X) has exactly j children $(x_1,X_1),\ldots,(x_j,X_j)$ for $x_1,\ldots,x_j \in V$.

A *directed elimination forest* for some digraph G with $|R(G)| = j$ reachable fragments $G_1,\ldots,G_j$, is a rooted forest consisting of directed elimination trees for $G_1,\ldots,G_j$.

For some rooted tree T the *height* $h(T)$ is the number of edges on a longest path between the root and a leaf. For some forest F of rooted trees the height $h(F)$ is the maximum height of its trees.

Observation 2.2.30 ([GKR21b])**.** *For a digraph G the DAG-depth can be determined as follows:*

$$ddp(G) = 1 + \min\{h(F) \mid F \textit{ is a directed elimination forest for } G\}.$$

2.3 Directed Graph Classes

Directed graph classes, or digraph classes for short, are special subsets of the set of all digraphs, that fulfill certain conditions. We already introduced some well-known digraph classes as DAGs and directed paths, now we continue with the class of semicomplete digraphs. In a *semicomplete* digraph there is at least one edge between each two distinct vertices. A special property of semicomplete digraphs is that many problems are polynomial time solvable on their underlying undirected graph, which is very natural since it is a clique. This makes this class particularly interesting in the context of NP-hard problems on digraphs, as it is not possible to transfer the problem directly and apply the same methods as for solving the corresponding problem on the undirected graph. More about semicomplete digraphs and related classes can be read in [BJG09].

The class of semicomplete digraphs is a superclass of the so-called *tournaments* which received significant attention in the past [CS11, KS15]. In a *tournament* there is exactly one edge between each two distinct vertices. If G is transitive and a tournament, it is a *transitive tournament*. The class of transitive tournaments is denoted by TT and a member of this class with n vertices is denoted by $\overrightarrow{T_n}$. Transitive tournaments can be characterized in several ways, see [Gou12, Chapter 9]. A (directed) hamiltonian path in (directed) graph G is a (directed) path that contains each vertex of $V(G)$ exactly once.

Lemma 2.3.1 ([Gou12]). *For every digraph G the following statements are equivalent.*

1. *G is a transitive tournament.*
2. *G is an acyclic tournament.*
3. *G is a tournament with exactly one directed Hamiltonian path.*
4. *G is a tournament and every vertex in G has a different outdegree, i.e., $\{outdegree(v) \mid v \in V(G)\} = \{0, \ldots, |V| - 1\}$.*
5. *G can be constructed from the one-vertex graph by repeatedly adding an out-dominating vertex.*
6. *G can be constructed from the one-vertex graph repeatedly adding an in-dominated vertex.*

A grid is a (di)graph with k rows of m vertices such that each vertex is only connected to its neighbor directly to the left, to the right, the one on the top and below such that it looks very matrix-alike, see Figure 2.6.

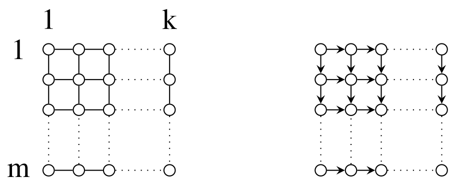


Figure 2.6: An undirected grid with $m \times k$ vertices (left) and an acyclic orientation of a grid (right).

3 Recursive Digraph Classes: Characterization and Hierarchy

3.1 Introduction

Recursive (directed) graph classes are classes of digraphs which can be defined recursively by several instructions. We normally start with the smallest graph, which is member of the class and then follow some rules. These classes are also called decomposable graphs, as many of them can be represented by a special decomposition. These structures can be used in algorithms, as showed e.g. in [BPT09]. Some examples of these classes will be introduced in this chapter.

3.2 Undirected Graph Classes

3.2.1 Co-graphs

Some NP-hard graph problems, e.g. the maximum clique or the minimum vertex cover problem can be solved efficiently or in polynomial time on so-called complement reducible graphs, co-graphs for short, such that they are an important graph class in the current field of research. Most of the solving algorithms use the tree structure which defines a co-graph and which is described further below.

For the definition of co-graphs and subclasses we need the following operations. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two vertex-disjoint graphs.

- The *disjoint union* of G_1 and G_2, denoted by $G_1 \oplus G_2$, is the graph with vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2$.
- The *join* of G_1 and G_2, denoted by $G_1 \otimes G_2$, is the graph with vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2 \cup \{\{v_1, v_2\} \mid v_1 \in V_1, v_2 \in V_2\}$.

With these operations we define the class of co-graphs, which have been introduced independently by different authors, for example in [Jun78], [Ler71], [Sei74] or [Sum74].

Definition 3.2.1 (Co-Graphs [CLSB81])**.** The class of *co-graphs* is recursively defined as follows.

(i) A graph on a single vertex $(\{v\},\emptyset)$, denoted by $\bullet$, is a co-graph.

(ii) If G_1 and G_2 are two vertex-disjoint co-graphs, then (a) $G_1 \oplus G_2$ and (b) $G_1 \otimes G_2$ are co-graphs.

We denote the class of co-graphs by C.

As mentioned before, the recursive structure of co-graphs allows certain problems to be solved in linear time, e.g. see [CLSB81]. Further, this structure can be used to compute the path-width, as well as the tree-width of co-graphs in linear time [BM93].

A co-graph G can be represented by a *co-tree* T in which the leaves represent the vertices of G, while the inner nodes of T correspond to the operations which are applied on the respective subtrees.

Following [CLSB81], co-graphs can be defined equivalently by different properties.

Observation 3.2.2 ([CLSB81])**.** *The following statements about graph G are equivalent.*

1. *G is a co-graph.*
2. *co-G is a co-graph.*
3. *G contains no P_4 as induced subgraph.*
4. *G has clique-width of at most* 2*.*

3.2.2 Distance-Hereditary Graphs

This subsection is taken from [KR21]. Distance-hereditary graphs have been introduced by Howorka in 1977 [How77]. They are exactly those graphs that are distance-hereditary for their connected induced subgraphs, which means that if any two vertices u and v belong to a connected induced subgraph H of a graph G, then some shortest path between u and v in G has to be a subgraph of H. But this is not the only definition of distance-hereditary graphs. Most important from an algorithmic perspective are the definition by forbidden induced subgraphs as well as the recursive construction by so-called twins and pendant vertices.

We take different definitions of undirected distance-hereditary graphs from [BM86, Oum05]. Let G be an undirected, connected graph.

- A vertex $v \in V(G)$ is called *pendant* if there is $u \in V(G)$ such that $\{u,v\} \in E(G)$ and for all other $w \in V(G)$, $w \neq u$ it holds that $\{w,v\} \notin E(G)$.
- A vertex $v \in V(G)$ is called *twin* of $u \in V(G)$, if $N(x) \setminus \{y\} = N(y) \setminus \{x\}$.
- It is called *true twin* if $\{u,v\} \in E(G)$, otherwise *false twin*.

Observation 3.2.3 ([BM86, Oum05])**.** *The following conditions are equivalent.*

1. *G is distance-hereditary, $G \in DH$ for short, that is, for every two vertices u and v, all induced u,v-paths have the same length.*

2. *For every two vertices u and v that have distance* 2 *to each other, there is no induced path between u and v of length greater than* 2*.*

3. *The house, holes, domino, and gem (see Figure 3.13) are not induced subgraphs of G.*

4. *G has rank-width* [1] *at most one.*

5. *G can be defined recursively from a single vertex by adding twins and pendant vertices.*

The recursive structure emerging by adding twins and pendant vertices can be represented by the so-called pruning sequence, which is algorithmically useful to compute along this structure. For a distance-hereditary graph G, let $\sigma(G) = (v_0, \dots v_{n-1})$ be an ordering on $V(G)$. Then, $S(G) = (s_1, \dots, s_{n-1})$ is the pruning sequence of G, where for every $1 \leq j \leq i \leq n-1$, s_i is one of the following:

- (x_i, P, x_j) if x_i is a pendant vertex of x_j.
- (x_i, TT, x_j) if x_i is a true twin of x_j.
- (x_i, FT, x_j) if x_i is a false twin of x_j.

The relation between co-graphs and distance-hereditary graphs can be followed by a result from [CLSB81], where twins are called *siblings*.

Observation 3.2.4 ([CLSB81])**.** *A co-graph can be obtained starting with a single vertex, by adding true and false twins.*

This can also be shown as we can build a co-tree from a given pruning sequence that contains only true and false twins and the other way around. The intuitive idea behind this is the following: A leaf in a subtree T' in a co-tree T is always a twin of all the other leaves in T' with respect to the leaves in the remaining part of T, since they all have the same neighborhood to these vertices in T.

Concerning graph parameters, for undirected distance-hereditary graphs the tree-width is computable in linear time [BDK00]. The clique-width of any distance-hereditary graph is at most 3 [GR99]. Further, linear rank-width of distance-hereditary graphs is computable in polynomial time [AKK17], but path-width is hard even on bipartite distance-hereditary graphs [KBMK93].

3.2.3 Series-Parallel Graphs

Undirected series-parallel graphs are formed recursively by parallel and series composition [BLS99, Section 11.2], which are defined as follows.

Let G_1 and G_2 be vertex-disjoint graphs such that each of the two has exactly one source and one sink.

- the *parallel composition* $G_1 \cup G_2$ identifies the source of G_1 with the source of G_2 and the sink of G_1 with the sink of G_2.

[1]Rank-width has been introduced by Oum and Seymour, see [OS06] for a definition.

- the *series composition* $G_1 \times G_2$ identifies the sink of G_1 with the source of G_2.

Definition 3.2.5 (Series-Parallel Graphs)**.** The class of *series-parallel graphs* consists of graphs with two distinguished vertices called terminals and is recursively defined as follows.

(i) A graph with two distinct vertices joined by a single edge $(\{u,v\},\{\{u,v\}\})$, denoted by $\{u,v\}$, is a *series-parallel graph*. We call vertex u the *source* and vertex v the *sink*.

(ii) If G_1 and G_2 are vertex-disjoint series-parallel graphs, then

  (a) the parallel composition $G_1 \cup G_2$ is a *series-parallel graph* and

  (b) the series composition $G_1 \times G_2$ is a *series-parallel graph*.

From a practical point of view these graphs have interesting applications in modeling series and parallel electric circuits. Additionally, this class play an important role in theoretical computer science, as they have tree-width of at most 2 and they are K_4-minor free graphs [Bod98].

3.2.4 Subclasses of Directed Co-graphs

In Figure 3.2 we summarize co-graphs and some well-known subclasses. All of them can be defined recursively by the operations showed in the third column of the shown table, starting with a single vertex denoted by •, a clique K, or an edgeless graph I. As shown in the table, it is also possible to characterize them by a set of forbidden induced subgraphs, see [CLSB81, Gol78, CH77, NP11, HMP11].

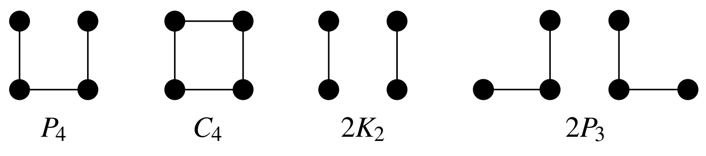


Figure 3.1: Some special undirected graphs.

In Figure 3.3 we compare the above graph classes to each other and show the hierarchy of the subclasses of co-graphs, the corresponding classes are defined in Figure 3.2.

3.3 Directed Co-graphs

3.3.1 Definition of Directed Co-graphs

Directed co-graphs are a recursive digraph class which are the directed version of the already introduced class of complement reducible graphs. In [GWY16] the set of directed co-graphs is characterized by excluding two digraphs and at the same time being a proper subset of the set of all graphs of directed clique-width 2.

class X	notation	operations			Forb(X)
co-graphs	C	$\bullet$	$G_1 \oplus G_2$	$G_1 \otimes G_2$	P_4
quasi threshold/trivially perfect graphs	TP	$\bullet$	$G_1 \oplus G_2$	$G_1 \otimes \bullet$	P_4, C_4
co-quasi threshold/co-trivially perfect graphs	CTP	$\bullet$	$G_1 \oplus \bullet$	$G_1 \otimes G_2$	P_4, $2K_2$
threshold graphs	T	$\bullet$	$G_1 \oplus \bullet$	$G_1 \otimes \bullet$	P_4, C_4, $2K_2$
simple co-graphs	SC	$\bullet$	$G_1 \oplus I$	$G_1 \otimes I$	P_4, co-$2P_3$, $2K_2$
co-simple co-graphs	CSC	$\bullet$	$G_1 \oplus K$	$G_1 \otimes K$	P_4, $2P_3$, C_4
weakly quasi threshold graphs	WQT	I	$G_1 \oplus G_2$	$G_1 \otimes I$	P_4, co-$2P_3$
co-weakly quasi threshold graphs	CWQT	K	$G_1 \oplus K$	$G_1 \otimes G_2$	P_4, $2P_3$

Figure 3.2: [GKR21c] Overview on subclasses of co-graphs. By G_1 and G_2 we denote graphs of the class X, by I we denote an edgeless graph and by K we denote a complete graph. The given forbidden sets in the last column are known from the existing literature [CLSB81, Gol78, CH77, NP11, HMP11]. The corresponding graphs can be found in Figure 3.1.

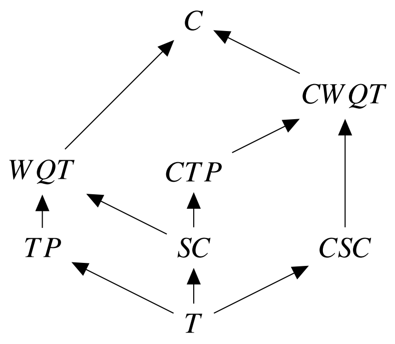

$$\begin{tikzcd}
 & & C & \\
 & & & CWQT \arrow[ul] \\
WQT \arrow[uurr] & CTP \arrow[urr] & & \\
TP \arrow[u] & SC \arrow[ul] \arrow[u] & & CSC \arrow[uu] \\
 & T \arrow[ul] \arrow[u] \arrow[urr] & &
\end{tikzcd}$$

Figure 3.3: [GKR21c] Relations between the subclasses of co-graphs. If there is a path from A to B, then it holds that $A \subset B$. The classes, that are not connected by a directed path are incomparable.

At first, we introduce operations that are used in the definition of directed co-graphs from [BdGR97]. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two vertex-disjoint digraphs.[2]

- The *disjoint union* of G_1 and G_2, denoted by $G_1 \oplus G_2$, is the digraph with vertex set $V_1 \cup V_2$ and arc set $E_1 \cup E_2$.
- The *series composition* of G_1 and G_2, denoted by $G_1 \otimes G_2$, is the digraph with vertex set $V_1 \cup V_2$ and arc set $E_1 \cup E_2 \cup \{(u,v),(v,u) \mid u \in V_1, v \in V_2\}$.
- The *order composition* of G_1 and G_2, denoted by $G_1 \oslash G_2$, is the digraph with vertex set $V_1 \cup V_2$ and arc set $E_1 \cup E_2 \cup \{(u,v) \mid u \in V_1, v \in V_2\}$.

[2]We use the same symbols for the disjoint union and join between undirected and digraphs. Although the meaning becomes clear from the context we want to emphasize this fact.

Additionally, the directed union transformation was introduced by Johnson et al. in [JRST01b]. It generalizes the operations disjoint union and order composition.

- A graph G is obtained by a *directed union* of G_1 and G_2, denoted by $G_1 \ominus G_2$, if G is a subdigraph of the order composition of $G_1 \oslash G_2$ and contains the disjoint union $G_1 \oplus G_2$ as a subdigraph.

Notice that, the directed union is not unique and thus no operation. Every graph which can be obtained by the operations introduced above, is constructible by a tree structure or even a sequence, as we see for undirected co-graphs and threshold graphs. These tree structures or sequences can be used for algorithmic properties of those graphs.

Definition 3.3.1 (Directed co-graphs, [CP06])**.** The class of *directed co-graphs*, DC for short, is recursively defined as follows.

(i) Every digraph on a single vertex $(\{v\},\emptyset)$, denoted by $\bullet$, is a *directed co-graph.*

(ii) If G_1 and G_2 are vertex-disjoint directed co-graphs, then

  (a) the disjoint union $G_1 \oplus G_2$,

  (b) the series composition $G_1 \otimes G_2$, and

  (c) the order composition $G_1 \oslash G_2$ are directed co-graphs.

The recursive definition of directed and undirected co-graphs leads to the following observation.

Observation 3.3.2. *For every directed co-graph G the underlying undirected graph $und(G)$ is a co-graph.*

The reverse direction only holds under certain conditions, see Theorem 3.3.6.

Obviously, for every directed co-graph we can define a tree structure, denoted as *(binary) di-co-tree*. We take a definition from [GKR20b].

Definition 3.3.3 (Di-co-tree)**.** The *di-co-tree* for a directed co-graph G is recursively defined as follows.

- The di-co-tree T for di-co-expression v consists of a single vertex r (the root of T) labeled by v.

- The di-co-tree T for di-co-expression $G_1 \oplus G_2$ consists of a copy T_1 of the di-co-tree for G_1, a copy T_2 of the di-co-tree for G_2, an additional vertex r (the root of T) labeled by $\oplus$ and two additional arcs from vertex r to the roots of T_1 and T_2. The root of T_1 is the *first child* of r and the root of T_2 is the *second child* of r.

- The di-co-tree T for di-co-expressions $G_1 \otimes G_2$ and $G_1 \oslash G_2$ are defined analogously to $G_1 \oplus G_2$.

While the leaves of the di-co-tree represent the vertices of the graph, the inner nodes of the di-co-tree correspond to the operations applied on the subexpressions which are defined by the associated subtrees. In a binary di-co-tree is a di-co-tree in which each inner vertex has only one or two successors. We can convert every non-binary di-co-tree in a binary one in linear time, as well as constructing a di-co-tree.

Theorem 3.3.4 ([CP06]). *A binary di-co-tree T can be computed in $O(n+m)$ time from a directed co-graph with n vertices and m arcs.*

Crespelle and Paul [CP06] also developed a recognition algorithm for directed co-graphs. Giving a digraph, their algorithm decides in linear time whether the digraph is a directed co-graph and if not, it returns a certificate. This is possible as the class has certain forbidden induced subdigraphs, see Theorem 3.3.6.

Directed co-graphs are interesting from an algorithmic point of view since several hard graph problems can be solved in polynomial time by dynamic programming along the tree structure of the input digraph, see [BJM14, Gur17, GR18]. Additionally, we use this property in Chapter 4,5, 6 and 7.

The property of complement reducibility is also found in directed co-graphs.

Lemma 3.3.5 ([GKR21b]). *For a digraph G the following properties hold.*

1. *Digraph G is a directed co-graph if and only if digraph co-G is a directed co-graph.*
2. *Digraph G is a directed co-graph if and only if digraph G^c is a directed co-graph.*

Directed co-graphs are closed under taking induced subdigraphs and can be characterized by excluding forbidden subdigraphs, see [CP06, Figure 2].

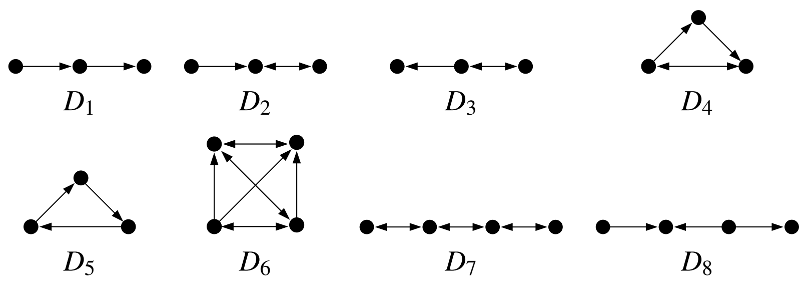


Figure 3.4: The eight forbidden induced subdigraphs for directed co-graphs (see [CP06]).

The following properties hold for directed co-graphs:

Theorem 3.3.6 ([GKR21c]). *Let G be a digraph. The following properties are equivalent:*

1. *G is a directed co-graph.*
2. *$G \in Free(\{D_1,\ldots,D_8\})$.*

3. *$G \in Free(\{D_1, \dots, D_6\})$ and $und(G) \in Free(\{P_4\})$.*

4. *$G \in Free(\{D_1, \dots, D_6\})$ and $und(G)$ is a co-graph.*

5. *G has directed NLC-width 1*[3].

6. *G has directed clique-width at most 2 and $G \in Free(\{D_2, D_3\})$.*

The proof can be read in [GKR21c], the statements follow from literature [CP06] and [GWY16]. Since the complement digraphs of $\{D_1, \dots, D_8\}$ are also included in $\{D_1, \dots, D_8\}$, it holds the following.

Proposition 3.3.7. *$DC = co\text{-}DC$.*

3.3.2 Oriented Co-graphs

Definition 3.3.8 (Oriented Co-Graphs)**.** The class of *oriented co-graphs*, *OC* for short, is recursively defined as follows.

1. Every digraph on a single vertex $(\{v\}, \emptyset)$, denoted by $\bullet$, is an oriented co-graph.

2. If G_1, G_2 are two vertex-disjoint oriented co-graphs, then

   (a) $G_1 \oplus G_2$ and

   (b) $G_1 \oslash G_2$ are oriented co-graphs.

Obviously, we can find a di-co-tree in linear time, as for directed co-graphs, see Theorem 3.3.4. The property of recursiveness of oriented and undirected co-graphs leads us to the following observation.

Observation 3.3.9. *For every oriented co-graph G the underlying undirected graph $und(G)$ is a co-graph.*

The reverse direction only holds under certain conditions, see Theorem 3.3.10 below. The class of oriented co-graphs is closed under taking induced subdigraphs.

Theorem 3.3.10 ([GKR19c])**.** *Let G be a digraph. The following properties are equivalent:*

1. *G is an oriented co-graph*

2. *$G \in Free(\{D_1, D_5, D_8, \overleftrightarrow{K_2}\})$.*

3. *$G \in Free(\{D_1, D_5, \overleftrightarrow{K_2}\})$ and $und(G) \in Free(\{P_4\})$.*

4. *$G \in Free(\{D_1, D_5, \overleftrightarrow{K_2}\})$ and $und(G)$ is a co-graph.*

5. *G has directed clique-width at most 2 and $G \in Free(\{\overleftrightarrow{K_2}\})$.*

[3]For a definition of directed NLC-width, see [GWY16].

6. *G is transitive and $G \in Free(\{\overleftrightarrow{K_2}, D_8\})$.*

The corresponding digraphs can be found in Figure 3.1 and 3.4. Following the notations of [VTL82] we denote the orientation of a P_4 which produced the D_8, see Figure 3.4 or 3.5, as the N graph. The class of oriented co-graphs has been analyzed by Lawler [Law76] and in [CLSB81, Section 5] with the notation of *transitive series-parallel (TSP) digraphs*.

Every oriented co-graph is obviously a DAG, since no cycles can emerge by the allowed operations. An superclass of oriented co-graphs is the class of transitive DAGs. As an example for a transitive DAG which is not an oriented co-graph we can just consider the N-graph ($= D_8$), see Figure 3.5.

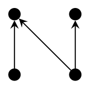


Figure 3.5: The N-graph.

3.3.3 Extended Directed Co-graphs

Since the directed union generalizes the disjoint union and also the order composition, we can define the class of extended directed co-graphs, which forms a superclass of directed and oriented co-graphs. The content is mostly from [GKR21b].

Definition 3.3.11 (Extended directed co-graphs)**.** The class of *extended directed co-graphs*, *EDC* for short, is recursively defined as follows.

(i) Every digraph on a single vertex $(\{v\}, \emptyset)$, denoted by $\bullet$, is an *extended directed co-graph.*

(ii) If G_1 and G_2 are vertex-disjoint extended directed co-graphs, then

  (a) every directed union $G_1 \ominus G_2$ and

  (b) the series composition $G_1 \otimes G_2$ are *extended directed co-graphs.*

As well as for directed co-graphs, for every extended directed co-graph we can define a tree structure, denoted as *ex-di-co-tree*. Like in a di-co-tree, the leaves of the ex-di-co-tree represent the vertices of the graph and the inner nodes of the ex-di-co-tree correspond to the operations applied on the subexpressions defined by the subtrees. For the class of extended directed co-graphs it remains open how to compute an ex-di-co-tree for a given digraph G. Since in the directed union it is not clear which edges arise, a special ex-di-co-tree represents several different extended directed co-graphs. Regarding the properties of of extended directed co-graphs, we see that they are not closed under complementation. By applying the directed union, which is not a disjoint union or an order composition, we can obtain digraphs whose complement digraphs are not extended directed co-graphs. An example for this is the directed path on 3 vertices $\overrightarrow{P_3}$. Thus, we only can carry over one of the two results shown in Lemma 3.3.5 to the class of extended directed co-graphs, see [GKR21b].

Lemma 3.3.12. *A digraph G is an extended directed co-graph if and only if digraph G^c is an extended directed co-graph.*

Proof. Let X be an expression using extended directed co-graph operations for G. We can get G^c given by an expression X' by modifying X as follows. For every $X_1 \ominus X_2$ in X, we change this into $X_2 \ominus X_1$ in X'. The rest remains as in X. □

The class of extended directed co-graphs is much more powerful than the class of directed co-graphs. An interesting fact is that the oriented extended directed co-graphs, thus extended directed co-graphs without series operations, are exactly the class of all DAGs.

Observation 3.3.13. *G is an acyclic extended directed co-graph if and only if G is a DAG.*

Proof. If G is a DAG with n vertices, G is a subgraph of the transitive tournament with n vertices. We can construct a transitive tournament by applying the order composition. To get G we just replace the order composition by a directed union and leave out all the edges $e \notin E(G)$. Every acyclic digraph G is a DAG, thus this direction trivially holds. □

This can also be proven by using a topological ordering.

3.3.4 Isomorphism Problem on Oriented Co-graphs

Using the tree-structure of a directed co-graph or the bounded directed clique-width of the class, there are many NP-hard problems which are solved on directed co-graphs in linear or at least polynomial time e.g. the Hamiltonian path, Hamiltonian cycle, regular subdigraph, and directed cut problem are polynomial on directed co-graphs, see [Gur17]. Directed co-graphs were also used together with with pomset logic in [Ret98]. From [BJM14] we know as well that the weak k-linkage problem is solvable in polynomial time on the class of directed co-graphs. Further, the recursive structure leads us to the existence of linear time dynamic programming algorithms for the computation of the size of a largest independent set, the size of a largest directed clique, the size of a largest subdigraph that is a tournament and the size of a largest semicomplete subdigraph of a directed co-graph. In the following we take a look at the Isomorphism problem. On undirected co-graphs the isomorphism problem is solvable in polynomial time, see [CLSB81]. This result can be improved by the following instructions. The content of this subsection is taken from [GKR19c].

For an oriented co-graph, a di-co-tree T is *canonical* if on every path from the root to the leaves of T, the disjoint union and order operation strictly alternate. As the disjoint union $\oplus$ and the order operations $\oslash$ are associative, we get the following lemma for an ordered structure in a di-co-tree.

Lemma 3.3.14 ([GKR19c])**.** *Let G be an oriented co-graph and T be a di-co-tree for G, then, T can be transformed into a canonical di-co-tree for G in linear time.*

Two undirected co-graphs G_1 and G_1 are isomorphic if and only if their corresponding canonical co-trees T_1 and T_2 are isomorphic. We get a canonical co-tree of a co-graph in linear time. Thus, by applying a linear time isomorphism test for rooted labeled trees (cf. [AHU74], Section 3.2) on canonical co-trees, we decide in linear time whether G_1 and G_2

are isomorphic for undirected co-graphs. Formally, the isomorphism problem on oriented co-graphs is defined as follows:

Name: Oriented co-graph isomorphism problem
Instance: Two oriented co-graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$.
Question: Are G_1 and G_2 isomorphic, i.e., is there a bijection $b : V_1 \to V_2$ such that for all $u, v \in V_1$ it holds that $(u, v) \in E_1$ if and only if $(b(u), b(v)) \in E_2$?

This method of using an isomorphism test for rooted labeled trees on the co-trees can not directly be used for oriented co-graphs, as we have to preserve the order of the vertices, which are representing the order operations in the di-co-tree. So we give a procedure in Algorithm 1 that provides a solution for di-co-trees. Our solution is obtained by a modification of the method given in [AHU74, Section 3.2]. There is also explained in detail, how the labeling of the tree is working.

Theorem 3.3.15. *Let G_1 and G_2 be two oriented co-graphs, then oriented co-graph isomorphism for G_1 and G_2 can be solved in linear time.*

Proof. Let G_1 and G_2 be two oriented co-graphs with the corresponding di-co-trees T_1 and T_2, which can be found in linear time with Theorem 3.3.4. Moreover we can assume, that the di-co-trees are canonical by Lemma 3.3.14. If two graphs are isomorphic, the two canonical di-co-trees must be isomorphic, too. W.l.o.g. assume that the height and the roots of T_1 and T_2 are equal. Then, if two trees are isomorphic, there must be a bijection from the vertices of T_1 of level ℓ to the vertices of T_2 of level ℓ. We look at the procedure from Algorithm 1. Under the given conditions, the operation of the vertices of level ℓ are either order compositions or disjoint unions for both trees. If the operation on level ℓ is a directed union, the labels of the children of each node on level ℓ are sorted. Otherwise, it is an order composition, where the order of the children cannot be changed, such that the labels of the children will stay in the same order. After visiting every vertex on level ℓ, the vectors with the labels of the children are sorted in the sequences S_1 and S_2. With the method given in [AHU74] (Section 3.2) the sorting can be done in linear time with respect to the number of edges from each vertex to its children. If both sequences are equal, the algorithm continues, since the isomorphism is satisfied for level $\ell + 1$. If it is not, the ordered sequences will be different, such that the algorithm terminates and returns false. This is repeated for every level of both trees, except for level 0, which is the root, where the operations are assumed to be equal, and level h, which is the first level the algorithm goes through. When the leaves on this level are labeled, there is nothing more to do, since these vertices have no children. The isomorphism of level h is checked on level $h - 1$. Let n be the number of vertices in T_1 and T_2 and m the number of edges. Then, the algorithm needs $2n$ steps for looking at every vertex of both trees, additional to $2m$ steps for looking at the children of each vertex. Thus, it runs in linear time. $\square$

3.4 Subclasses of Directed Co-graphs

In order to obtain new insights for the study of directed graph classes we investigate further recursive digraph classes. After showing a superclass and a subclass of directed co-graphs

Algorithm 1 [GKR19c] Testing graph isomorphism for two oriented co-graphs given by canonical di-co-trees.

```
procedure TEST($T_1, T_2$)

let $h$ be the height of $T_1$ and $T_2$
for $\ell = h$ downto 0 do
    for all vertices $v$ on level $\ell$ in $T_1$ from left to right do
        if ($v$ is a leaf)
            label[$v$] $= 0$
        else
            let $v_1, \ldots, v_r$ be the children of $v$
            label[$v$] $=$ (label[$v_1$], ..., label[$v_r$])
            if ($v$ corresponds to a union operation)
                sort vector label[$v$] ascending
    let $S_1$ be the sequence of all label[$v$] for all $v$ on level $\ell$ in $T_1$
    for all vertices $v$ on level $\ell$ in $T_2$ from left to right do
        if ($v$ is a leaf)
            label[$v$] $= 0$
        else
            let $v_1, \ldots, v_r$ be the children of $v$
            label[$v$] $=$ (label[$v_1$], ..., label[$v_r$])
            if ($v$ corresponds to a union operation)
                sort vector label[$v$] ascending
    let $S_2$ be the sequence of all label[$v$] for all $v$ on level $\ell$ in $T_2$
    sort $S_1$ to obtain $S'_1$ and sort $S_2$ to obtain $S'_2$
    if ($S'_1 \neq S'_2$)
        return $false$
    let $V_\ell$ be the set of all vectors on level $\ell$ in $T_1$
    find a bijection $b : V_\ell \to \{1, \ldots, |V_\ell|\}$
    for all vertices $v$ on level $\ell$ in $T_1$ do
        label[$v$] $= b(v)$;
    for all vertices $v$ on level $\ell$ in $T_2$ do
        label[$v$] $= b(v)$;
}
return $true$;
```

we now continue with further subclasses, which are motivated by corresponding undirected versions of these classes. The content of this section is taken from [GKR21c].

3.4.1 Oriented Threshold Graphs

The class of oriented threshold graphs has been introduced in [Boe18] as follows.

Definition 3.4.1 (Oriented threshold graphs [Boe18])**.** A graph G is a threshold graph if there exists an injective weight function $w : V(G) \to \mathbb{R}$ and a threshold value $t \in \mathbb{R}$ such that $(x,y) \in E(G)$ if and only if $|w(x)| + |w(y)| \geq t$ and $w(x) > w(y)$.

Theorem 3.4.2 ([Boe18])**.** *Let G be an oriented graph. The following properties are equivalent:*

1. *G is an oriented threshold graph.*
2. *$G \in Free(\{D_1, D_5\})$ and $und(G) \in Free(\{2K_2, C_4, P_4\})$.*
3. *G is a transitive orientation of a threshold graph.*
4. *G can be constructed from the one vertex empty graph by successively adding an isolated vertex, an out-dominating vertex or an in-dominated vertex.*

Using Theorem 3.4.2 we obtain the following definition, which is equivalent to Definition 3.4.1:

Definition 3.4.3 (Oriented threshold graphs)**.** The class of *oriented threshold graphs*, OTD for short, is recursively defined as follows.

(i) Every digraph on a single vertex $(\{v\}, \emptyset)$, denoted by $\bullet$, is an oriented threshold graph.

(ii) If G is an oriented threshold graph, then (a) $G \oplus \bullet$, (b) $G \oslash \bullet$, and (c) $\bullet \oslash G$ are oriented threshold graphs.

The recursive definition of oriented and undirected threshold graphs lead to the following observation.

Observation 3.4.4. *For every oriented threshold graph G the underlying undirected graph $und(G)$ is a co-graph.*

This class can also be defined by forbidden induced subdigraphs. As it was possible for oriented co-graphs and oriented trivially perfect graphs, we can use the fact that oriented threshold graphs are exactly the directed threshold graphs not containing an induced $\overleftrightarrow{K_2}$:

Theorem 3.4.5. *Let G be a digraph. The following properties are equivalent:*

1. *G is an oriented threshold graph.*
2. *G is an oriented co-trivially perfect graph.*

3. *$G \in Free(\{D_1, D_5, D_8, D_{12}, 2\overrightarrow{P_2}, \overleftrightarrow{K_2}\})$.*

4. *$G \in Free(\{D_1, D_5, \overleftrightarrow{K_2}\})$ and $und(G) \in Free(\{2K_2, C_4, P_4\})$.*

5. *$G \in Free(\{D_1, D_5, \overleftrightarrow{K_2}\})$ and $und(G)$ is a threshold graph.*

6. *$G \in Free(\{D_1, D_5, D_{12}, \overleftrightarrow{K_2}\})$ and $und(G) \in Free(\{P_4, 2K_2\})$.*

7. *$G \in Free(\{D_1, D_5, D_{12}, \overleftrightarrow{K_2}\})$ and $und(G)$ is a co-trivially perfect graph.*

8. *$G \in Free(\{D_1, D_5, D_{12}, 2\overrightarrow{P_2}, \overleftrightarrow{K_2}\})$ and $und(G) \in Free(\{P_4\})$.*

9. *$G \in Free(\{D_1, D_5, D_{12}, 2\overrightarrow{P_2}, \overleftrightarrow{K_2}\})$ and $und(G)$ is a co-graph.*

10. *G is transitive and $G \in Free(\{D_8, D_{12}, 2\overrightarrow{P_2}, \overleftrightarrow{K_2}\})$.*

The digraphs can be found in Figure 3.1, 3.4 and 3.8.

Proof. (1) $\Leftrightarrow$ (2) By the recursive definition of the classes, which arises from the restriction of the directed classes to oriented graphs.
(1) $\Rightarrow$ (3) If G is an oriented threshold graph, then G is a directed threshold graph and thus, it holds that $G \in \text{Free}(\{D_1, \ldots, D_{15}, \text{co-}D_{11}, \text{co-}D_{10}, \text{co-}D_9\})$, see [GKR21c, Theorem 18]. Further $G \in \text{Free}(\{\overleftrightarrow{K_2}\})$ because of the missing series composition. This leads to $G \in \text{Free}(\{D_1, D_5, D_8, D_{12}, \overleftrightarrow{K_2}, 2\overrightarrow{P_2}\})$.
(3) $\Rightarrow$ (1) If $G \in \text{Free}(\{D_1, D_5, D_8, D_{12}, \overleftrightarrow{K_2}, 2\overrightarrow{P_2}\})$, then $G \in \text{Free}(\{D_1, \ldots, D_{15}, \text{co-}D_{11}, \text{co-}D_{10}, \text{co-}D_9\})$ and is a directed threshold graph. Since $G \in \text{Free}(\{\overleftrightarrow{K_2}\})$ there is no series operation in any construction of G which implies that G is an oriented threshold graph.
(4) $\Leftrightarrow$ (5) Since Forb(T) $= \{C_4, P_4, 2K_2\}$ (Figure 3.10).
(6) $\Leftrightarrow$ (7) Since Forb(CTP) $= \{P_4, 2K_2\}$ (Figure 3.10).
(8) $\Leftrightarrow$ (9) Since Forb(C) $= \{P_4\}$ (Figure 3.10).
(3) $\Rightarrow$ (5), (3) $\Rightarrow$ (7), and (3) $\Rightarrow$ (9) By Observation 3.4.4.
(4) $\Rightarrow$ (3), (6) $\Rightarrow$ (3), and (8) $\Rightarrow$ (3) By Observation 2.1.6
(3) $\Rightarrow$ (10) By [GKR21c, Lemma 3] we know that G is transitive.
(10) $\Rightarrow$ (3) If G is transitive, then $G \in \text{Free}(\{D_1, D_5\})$. □

A very famous subclass of oriented threshold graphs is the previously defined class of transitive tournaments, which can be easily constructed by repeating $G \oslash \bullet$.

The proof for the next Theorem can be done very similar to the proof of [GKR21c, Theorem 10].

Theorem 3.4.6 ([GKR21c])**.** *A graph G is a threshold graph if and only if there exists an orientation G' of G such that G' is an oriented threshold graph.*

Observation 3.4.7 ([GKR21c])**.** *If $G \in DT$, then the underlying undirected graph of the symmetric part of G is a threshold graph and the asymmetric part of G is an oriented threshold graph.*

This holds since the asymmetric part is constructed according to the same rules as threshold graphs and the asymmetric part is constructed according to the rules of OTD. Similar to directed threshold graphs every oriented co-graph can be defined by a sequence with only three operations, which can be used to give a linear time recognition algorithm.

3.4.2 Threshold Digraphs and Ferres Digraphs

The following idea of defining a directed version of threshold graphs, i.e., threshold digraphs, is from [CLMS14] where they define them by a set of forbidden subdigraphs. The content is taken from [GKR21c].

A 2-*switch* is a vertex set $\{w,x,y,z\}$ such that there exist the edges (w,x) and (y,z) but not the edges (w,z) and (y,x), see Figure 3.6. Examples for a 2-switch are co-D_{10}, co-D_9, co-D_{11} and $\overrightarrow{P_4}$.

Observation 3.4.8 ([GKR21c]). *Let G be a directed threshold graph, then G does not contain a* 2*-switch.*

Definition 3.4.9 (Threshold digraphs [CLMS14]). A digraph G is a *threshold digraph* if it does not contain a 2-switch nor a $\overrightarrow{C_3}$ as induced subdigraph. The class of threshold digraphs is denoted by TD.

This class is not very useful for the directed co-graph hierarchy, as they are incomparable to most of the graph classes in it, though it is a superclass of directed threshold graphs.

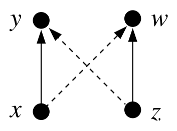


Figure 3.6: A 2-switch. All vertices are distinct. Solid arcs must appear in the digraph and dashed arcs must not appear in the digraph. If an arc is not given, then it may or may not be present.

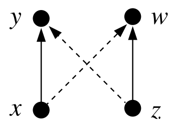


Figure 3.7: An alternating 4-anticircuit. The vertices are not necessarily distinct but $x \neq z$ and $y \neq w$. The solid arrows represent the presence of an arc and a dashed arrow its absence.

A well studied class of digraphs are Ferres digraphs, see [MP95, Chapter 2] for a survey. Ferres digraphs are introduced by Riguet in [Rig51]. In their first definition, Ferres digraphs were defined on digraphs including loops. As the subclasses of directed co-graphs do not use loops, only Ferres digraphs without loops will be used here. An *alternating 4-anticircuit* consists of vertices x,y,z,w, not necessarily distinct but $x \neq z$ and $y \neq w$, satisfying $(x,y),(z,w) \in A$ and $(x,w),(z,y) \notin A$ (cf. Figure 3.7).

Definition 3.4.10 (Ferres digraphs)**.** A digraph is a *Ferres digraph* if it does not contain an alternating 4-anticircuit. The class of Ferres digraphs is denoted by FD.

By considering all possible equalities of vertices in an alternating 4-anticircuit, an equivalent characterization is obtained by $G \in \text{Free}(\{D_1, \overleftrightarrow{K_2}\})$ and G does not contain a 2-switch, see [MP95, Figure 2.2] additional to the restriction to digraphs without loops. This class is comparable to oriented threshold graphs, but not to any other graph class in the directed co-graph hierarchy, as we will see in the overview.

3.4.3 Further Subclasses of Directed Co-graphs

In Figure 3.10 we summarize directed co-graphs and some subclasses. Since directed co-graphs and all defined subclasses are hereditary, by Theorem 2.1.2 there exist sets of minimal forbidden induced subdigraphs. Figure 3.10 shows the finite sets of minimal forbidden induced subdigraphs for the different digraph classes. These characterizations lead to polynomial time recognition algorithms for the corresponding digraph classes. The table and the forbidden subdigraphs are taken from [GKR21c] and also occur in [Kom19].

3.5 Twin-distance-hereditary Digraphs

Since the class of distance-hereditary graphs is a very useful one in undirected graph theory we try to consider this concept for directed graphs. Attempting to define a directed version of distance-hereditary graphs, it is necessary to decide which of these definitions modified to a directed definition is most promising to give a useful digraph class. We consider this matter concerning some digraph parameters. In [LS10] the authors use a straightforward way in generalizing the property of distance heredity on digraphs as described in the next subsection.

3.5.1 Distance-Hereditary Digraphs

As already mentioned, there has been previously an attempt to define a directed version of distance-hereditary graphs. A straightforward idea given by the name of the graph class is to say that a digraph G is called *distance-hereditary*, if for every induced subdigraph H of G and for every vertices u, v in H, the shortest directed path between u and v in H has the same length as the shortest directed path between u and v in G. This idea has been pursued in [LS10] but only for oriented graphs without bidirectional edges [Sch21]. They also show an equivalence to a characterization by a set of forbidden induced subdigraph as they stated that a digraph G is a distance-hereditary digraph if and only if G does not contain an induced subdigraph that is a so-called *skew bipath*, which is a special digraph structure with two paths where certain arcs which connect these two paths and therefore shorten them are forbidden [4]. This class of distance-hereditary digraphs is abbreviated as DHD in the following. The directed clique-width of this digraph class is not bounded, as we can see by the following example.

[4]See [LS10] for a formal definition.

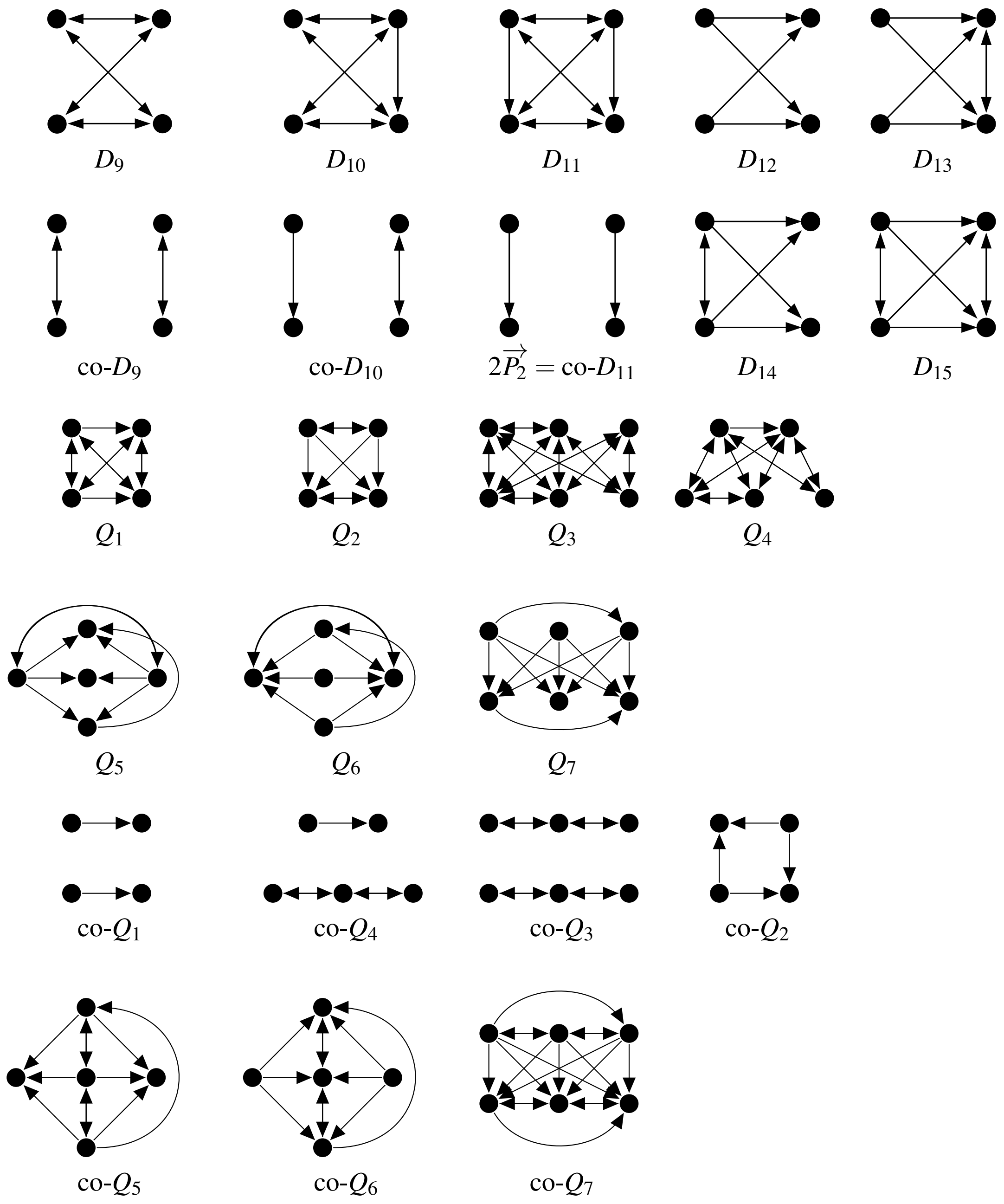


Figure 3.8: Digraphs from Figure 3.10

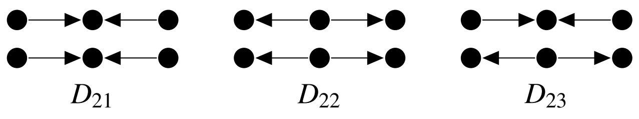


Figure 3.9: Forbidden induced subdigraphs of OCWQT.

class X	notation	operations					Forb(X)
directed co-graphs	DC	$\bullet$	$G_1 \oplus G_2$	$G_1 \oslash G_2$		$G_1 \otimes G_2$	$D_1, \dots, D_8$
oriented co-graphs	OC	$\bullet$	$G_1 \oplus G_2$	$G_1 \oslash G_2$			$D_1, D_5, D_8, \overleftrightarrow{K_2}$
directed trivially perfect	DTP	$\bullet$	$G_1 \oplus G_2$	$G_1 \oslash \bullet$	$\bullet \oslash G_2$	$G_1 \otimes \bullet$	$D_1, \dots, D_{15}$
oriented trivially perfect	OTP	$\bullet$	$G_1 \oplus G_2$	$G_1 \oslash \bullet$	$\bullet \oslash G_2$		$D_1, D_5, D_8, \overleftrightarrow{K_2}, D_{12}$
directed co-trivially perfect	DCTP	$\bullet$	$G_1 \oplus \bullet$	$G_1 \oslash \bullet$	$\bullet \oslash G_2$	$G_1 \otimes G_2$	$D_1, \dots, D_8, D_{12}, \dots, D_{15}$ co-D_{11}, co-D_{10}, co-D_9
oriented co-trivially perfect $*$	OCTP	$\bullet$	$G_1 \oplus \bullet$	$G_1 \oslash \bullet$	$\bullet \oslash G_2$		D_1, D_5, D_8, D_{12}, co-D_{11}, $\overleftrightarrow{K_2}$
directed weakly quasi threshold	DWQT	I	$G_1 \oplus G_2$	$G_1 \oslash I$	$I \oslash G_2$	$G_1 \otimes I$	$D_1, \dots, D_8, Q_1, \dots, Q_7$
oriented weakly quasi threshold	OWQT	I	$G_1 \oplus G_2$	$G_1 \oslash I$	$I \oslash G_2$		$D_1, D_5, D_8, \overleftrightarrow{K_2}, Q_7$
directed co-weakly quasi thresh.	DCWQT	K	$G_1 \oplus K$	$G_1 \oslash K$	$K \oslash G_2$	$G_1 \otimes G_2$	D_1, D_8, co-$Q_1, \dots,$ co-Q_7
oriented co-weakly quasi threshold	OCWQT	T	$G_1 \oplus T$	$G_1 \oslash T$	$T \oslash G_2$		$D_1, D_5, D_8, \overleftrightarrow{K_2}$, $D_{12}, D_{21}, D_{22}, D_{23}$
directed simple co-graphs	DSC	$\bullet$	$G_1 \oplus I$	$G_1 \oslash I$	$I \oslash G_2$	$G_1 \otimes I$	$D_1, \dots, D_8, Q_1, \dots, Q_7$, co-$D_9$, co-$D_{10}$, co-$D_{11}$
oriented simple co-graphs	OSC	$\bullet$	$G_1 \oplus I$	$G_1 \oslash I$	$I \oslash G_2$		$D_1, D_5, D_8, \overleftrightarrow{K_2}, Q_7$, co-$D_{11}$
directed co-simple co-graphs	DCSC	$\bullet$	$G_1 \oplus K$	$G_1 \oslash K$	$K \oslash G_2$	$G_1 \otimes K$	$D_1, \dots, D_8, Q_1$, co-$Q_1, \dots,$ co-Q_7, D_9, D_{10}
oriented co-simple co-graphs	OCSC	$\bullet$	$G_1 \oplus T$	$G_1 \oslash T$	$T \oslash G_2$		$D_1, D_5, D_8, \overleftrightarrow{K_2}$, $D_{12}, D_{21}, D_{22}, D_{23}$
directed threshold graphs	DT	$\bullet$	$G_1 \oplus \bullet$	$G_1 \oslash \bullet$	$\bullet \oslash G_2$	$G_1 \otimes \bullet$	$D_1, \dots, D_{15}$ co-D_{11}, co-D_{10}, co-D_9
oriented threshold graphs $*$	OTD	$\bullet$	$G_1 \oplus \bullet$	$G_1 \oslash \bullet$	$\bullet \oslash G_2$		$D_1, D_5, D_8, \overleftrightarrow{K_2}, D_{12}$, co-$D_{11}$

Figure 3.10: [GKR21c] Overview on subclasses of directed co-graphs, see Figure 3.4, 3.8 and 3.9 for illustrations of the forbidden induced subdigraphs. By G_1 and G_2 we denote graphs of the class X, by I we denote an edgeless graph, by K we denote a bidirectional complete digraph, and by T we denote a transitive tournament.

Example 3.5.1. Let G be an oriented $n \times m$ grid such that all arcs are oriented from left to right as from the top to the bottom, see Figure 2.6. G is a distance-hereditary digraph, since the path from each vertex to each other vertex has always the same length due to the grid structure. However, the clique-width increases with n or m, such that the directed clique-width is not bounded.

3.5.2 Motivation of Defining a New Class

There are several arguments that motivate a different definition of directed distance-hereditary graphs. A very obvious reason is that not all digraphs are covered by the former definition, since no bidirectional edges are allowed. Further, we cannot get a directed form of a pruning sequence for these digraphs. They give a relation to some kind of directed twins, but it is not possible to give a sequence which describes the digraph since several edges not necessarily exist. Thus, in a new definition we would have the concept of a directed pruning sequence and consequently, a recursive structure which precisely describes the digraph. The property of distance heredity is very reasonable such that it would be nice to keep that attribute. Since distance-hereditary graphs have bounded clique-width, it would be preferable to get a digraph class of bounded directed clique-width. This would also offer the possibility for parameterized solutions. Additionally, since the tree-width of distance-hereditary graphs is computable in linear time, we would like to have a class for which we also get efficient solutions for directed width parameters related to tree-width. Moreover, the new class should fit into the hierarchy, in such a manner that that directed co-graphs are a subclass, just as co-graphs are a subclass of distance-hereditary graphs in undirected graphs.

The content of this subsection is taken from [KR21]. There are at least three different definitions of twins in digraphs. In [KR09], twins are used to obtain distance-hereditary digraphs in context of directed rank-width and split decomposition. Thus, [KR09] can be seen as an attempt to extend undirected distance-hereditary graphs to directed distance-hereditary graphs. In [FHP19], twins have been defined to obtain results about domination and location-domination, and in [GY02] (see also [BJG18, p. 282]) in studying diameter in digraphs. In [LS10] twins are introduced in context of a distance based directed version of distance-hereditary graphs, but as said before, they do not lead to a characterization of this graph class. Further, in their twin construction there are optional arcs contained and no bidirectional arcs are allowed.

We define directed twins and pendant vertices in digraphs as follows.

Definition 3.5.2. Let G be a digraph.

- Vertices $x, y \in V(G)$ are *directed twins*[5] if $N^-(x) \setminus \{y\} = N^-(y) \setminus \{x\}$ and $N^+(x) \setminus \{y\} = N^+(y) \setminus \{x\}$. We distinguish between
  - x is a (directed) *false twin* ($\circ$) of y if $(x,y),(y,x) \notin E(G)$.
  - x is a *true out-twin* ($\leftarrow$) of y if $(y,x) \in E$, $(x,y) \notin E(G)$.
  - x is a *true in-twin* ($\rightarrow$) of y if $(x,y) \in E$, $(y,x) \notin E(G)$.

[5]We say twins for short, but the meaning is directed twins if the context is a digraph.

  - x is a *bioriented true twin* ($\leftrightarrow$) of y if $(x,y),(y,x) \in E(G)$.

- A vertex $v \in V(G)$ is called *pendant* if $|N^+(v)| + |N^-(v)| = 1$. We distinguish between

  - v is a *pendant plus* vertex ($+$) if $|N^+(v)| = 1$ and $|N^-(v)| = 0$.
  - v *pendant minus* vertex ($-$) if $|N^+(v)| = 0$ and $|N^-(v)| = 1$.

This leads to the definition of a recursively defined digraph class which fulfills the properties required above. We denote this class of digraphs as directed twin-distance-hereditary graphs.

Definition 3.5.3 (directed twin-distance-hereditary graphs)**.** A digraph $G = (V,E)$ is a *directed twin-distance-hereditary graph*, twin-dh digraph for short, if it can be constructed recursively by taking the disjoint union, adding twins and pendant vertices, starting from a single vertex. The class of twin-dh digraphs is denoted by DDH.

A *directed pruning sequence* for a twin-dh digraph G is a sequence $S(G) = (s_1,\ldots,s_{n-1})$, where $\sigma(G) = (v_0,\ldots,v_{n-1})$ is an ordering of $V(G)$ and every s_i is one of the following triples:

- $(v_i,+,v_{a_i})$ if v_i is a pendant plus vertex of v_{a_i} in $G[\{v_0,\ldots,v_i\}]$.
- $(v_i,-,v_{a_i})$ if v_i is a pendant minus vertex of v_{a_i} in $G[\{v_0,\ldots,v_i\}]$.
- $(v_i,\circ,v_{a_i})$ if v_i is a false twin of v_{a_i} in $G[\{v_0,\ldots,v_i\}]$.
- $(v_i,\leftarrow,v_{a_i})$ if v_i is a true out-twin of v_{a_i} in $G[\{v_0,\ldots,v_i\}]$.
- $(v_i,\rightarrow,v_{a_i})$ if v_i is a true in-twin of v_{a_i} in $G[\{v_0,\ldots,v_i\}]$.
- $(v_i,\leftrightarrow,v_{a_i})$ if v_i is a bioriented true twin of v_{a_i} in $G[\{v_0,\ldots,v_i\}]$.

In general, we denote $s_i = (v_i,op_i,v_{a_i})$ and say for vertex v_i, that op_i is the *operation* and v_{a_i} the *anchor vertex* of s_i.

Intuitively one can ask for the reason why there is not the concept of a bioriented pendant vertex. The inclusion of this operation would come with a couple of drawbacks. Most of all this would destroy the property that the class is a subclass of extended directed co-graphs and thus, lose all properties which we can transfer from this superclass. Due to this we only include oriented pendant vertices in our construction.

Example 3.5.4. The directed pruning sequence $S(G)$ below with $\sigma(G) = (v_0,\ldots,v_5)$ creates the twin-dh digraph G in Figure 3.11.

$$S = ((v_1,\leftrightarrow,v_0),(v_2,+,v_1),(v_3,\circ,v_1),(v_4,-,v_2),(v_5,\leftarrow,v_2))$$

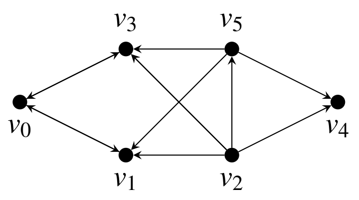


Figure 3.11: The twin-dh-digraph G from Example 3.5.4.

The algorithm used in [DHP01] can recognize a distance-hereditary graph and give its pruning sequence in linear time w.r.t. the number of vertices and edges. Unfortunately, this cannot directly be transferred to a recognition algorithm of twin-dh digraphs. To get the directed pruning sequence of a twin-dh digraph, we need a transformation of algorithm 3 from [DHP01]. Therefore, we need to define a directed version of a distance layout, which cannot be easily translated. But like in the undirected case, for a given twin-dh digraph, it is easy to get a directed pruning sequence.

Proposition 3.5.5. *Let G be a twin-distance-hereditary digraph. Then, a directed pruning sequence of G can be computed in polynomial time.*

Therefore, we check the pendant vertices in a first step of a loop and remove them. Then, we compare the neighborhoods of the different vertices and remove twins. After removing a twin we go on with removing pendant vertices again. If a vertex is removed, it is added to the directed pruning sequence with the corresponding operation.

3.5.3 Properties

The class of directed twin-distance-hereditary graphs is closed under the connected induced subdigraph operation. The content of this subsection is taken from [KR21].

Lemma 3.5.6. *Let G be a twin-dh digraph and let H be a weakly connected induced subdigraph of G. Then H is a twin-dh digraph.*

Proof. Let $G \in \text{DDH}$ with $V(G) = \{v_0, \dots, v_{n-1}\}$ and let $S(G) = (s_1, \dots, s_{n-1})$ with $\sigma(G) = (v_0, \dots, v_{n-1})$ be a directed pruning sequence of G. Let $H = G \setminus \{v\}$ be the weakly connected induced subdigraph H of G which emerges when deleting vertex v and all corresponding edges from G. We then create a directed pruning sequence $S(H)$ with ordering $\sigma(H)$ with the following procedures for the three different cases.

1. If $v = v_0$, we just delete s_1 from $S(G)$ to obtain $S(H)$ and adjust the indies, now v_1 is the first vertex in $\sigma(H)$.
2. If there exists $(v, op_i, a_i) \in S(G)$ and no (u_j, op_j, v) with $i < j$:
   (After generating v in $S(G)$, v never occurs as an anchor vertex.)
   In this case we get $S(H)$ by deleting (v, op_i, a_i) from $S(G)$ and adjust the indices.

3. If there exists $(v, op_i, a_i) \in S(G)$ and also $(u_{j1}, op_{j1}, v), ..., (u_{jk}, op_{jk}, v) \in S(G)$ with $i < j$ and $k, j \leq m-1$:
(After generating v in $S(G)$, v occurs at least once as an anchor vertex.)
Since the emerging digraph must be weakly connected, it holds that op_{jk} must be a directed twin operation. We get a temporary $S'(H) = (s'_1, \dots s'_{m-2})$ by the following steps.

   - For $t = 1, \dots, i-1$ let $s'_t = s_t$. We keep the directed pruning sequence until v is generated.
   - For $t = i$ we set $s'_i = (v', op_i, a_i)$ where v' is the vertex such that $s_h = (v', op_h, v)$ with op_h is a directed twin operation and $\nexists s_p = (v'', op_p, v)$ with $p > h$ and op_h is a directed twin operation. Thus, v' is the last twin of v with respect to $S(G)$. Now we replace v by v' as an anchor in all following occurrences. As v' is the latest twin of v w.r.t. $S(G)$, every operation applied on v is also applied on v'.
   - For $t = i+1, \dots, \ell$ with $i+1 \leq \ell \leq m-1$ and $v = a_t$ first check if $v' = u_t$ in $s_t = (u_t, op_t, a_t)$. If this situation arrives, we delete this s_t from the directed pruning sequence, such that we set $s'_t = (,,)$. Vertex v' is now generated earlier in the directed pruning sequence and we do not need this step anymore. We will delete this empty triple at the very end, such that we don't have counting issues in the following procedure. As long as $v' \neq u_t$ we set $s'_t = s_t$ if $v \neq a_t$ and we set $s'_t = (u_t, op_t, v')$ if $v = a_t$ for $s_t = (u_t, op_t, a_t)$.
   - For the remaining $t = \ell+1, \dots, m-1$ we set $s'_t = s_t$.

   At the end of this procedure, we delete the empty entry $s_c = (,,)$ from $S'(H)$, adjust the indices and get a directed pruning sequence $S(H)$ for H.

This holds for every weakly connected subdigraph H, since we can repeat this procedure for every vertex which is in G but not in H. Thus, we can always get a directed pruning sequence $S(H)$ and H is a twin-dh digraph. □

As every directed pruning sequence can easily be transformed into a pruning sequence, the relation to undirected distance-hereditary graphs follows immediately.

Proposition 3.5.7. *If G is a twin-dh digraph, then $und(G)$ is distance-hereditary.*

Twin-dh digraphs are distance-hereditary

Though for the definition we used the approach of a recursive construction by twins and pendant vertices, directed twin-distance-hereditary graphs still fulfill the distance heredity property.

Theorem 3.5.8. *Every twin-distance-hereditary digraph G is distance-hereditary, i.e., for every two vertices u and v in $V(G)$, all induced u,v-paths have the same length.*

In [LS10] the authors claim that for pendant vertices, for (slightly different, but more general defined) oriented twins and for false twins the distance-hereditary property remains fulfilled. However, the result that every path between two distinct vertices is of length one does not hold in general when including bidirectional edges. This is why we need the following lemma, which leads us directly to the theorem above. Notice that, the proof could be shortened using Theorem 4 of [LS10].

Lemma 3.5.9. *For two twins u, v in a directed twin-distance-hereditary graph G it holds that if there exists a path from u to v then the length of the shortest path in every induced subdigraph G' of G is at most* 2*.*

Proof. Let $u, v \in V(G)$ be twins in G. If they are bioriented twins, the distance between them is trivially 1. If u, v are oriented twins let w.l.o.g. be $(u, v) \in E(G)$. Then the distance from u to v is also 1, but this is not the case for the other direction. So let u and v be oriented twins with $(v, u) \in E(G)$ or false twins. In order to proof the lemma by contradiction, we assume that there is a shortest path from u to v of length ≥ 3 in an induced subdigraph G' of G. Let this path be $P = (u, v_1, \ldots, v_k, v)$. Since $N^-_{G'}(v) = N^-_{G'}(u)$ and $N^+_{G'}(v) = N^+_{G'}(u)$ it holds that $(v, v_1) \in E(G')$ and $(v_k, u) \in E(G')$. Then there is a cycle $(u, v_1, \ldots, v_k, u)$ of length at least 3. If the length is 3 with there must be at least two bidirectional edges in this cycle, otherwise this cycle is not constructible by directed twins. But then, one of the bidirectional edges goes to u and since v is a twin, we could have taken this shorter path (u, v_i, v) of length 2 from the beginning, which is a contradiction to the assumption of length 3. Let's assume the shortest path is > 3. Then there is a cycle $u, v_1, v_2, \ldots, v_k, u$ with the same argumentation as before. Since $und(G)$ is distance-hereditary, there cannot be any holes, thus induced cycles of length ≥ 5. Thus, the cycles must contain edges in between. If these edges are forward edges along the cycle they would shorten the path from u to v which is a contradiction. If these edges are backward edges along the cycle, they would again build smaller induced cycles, up to a $\overrightarrow{C_3}$ which is not constructible by a directed pruning sequence. Backward edges are only possible, if the outer edges from the cycle are bioriented. But this would build an induced subdigraph H_{19} or H_{16} (Figure 3.13), which are not constructible with a directed pruning sequence and thus are not directed twin-distance-hereditary. Thus, such a path cannot exists and the shortest path is always of length ≤ 2. $\square$

Since we include bidirectional edges in the new class and distance-hereditary digraphs can leave out certain edges, twin-dh digraphs are neither a superclass nor a subclass of distance-hereditary digraphs, although they are distance-hereditary. With the same example as for extended co-graphs, the class of distance-hereditary digraphs has unbounded directed clique-width, see Example 3.5.1. Here we see a certain advantage of the class of twin-dh digraphs which justifies to take a closer look.

3.5.4 Sub- and Superclasses

Directed co-graphs

The first part of this subsection is taken from [KR21]. In the undirected case, distance-hereditary graphs can be classified into the hierarchy with other graph classes. Especially, they are a superclass of co-graphs by the definition of co-graphs using twins. We now show, that this is also possible in the directed case.

Proposition 3.5.10. *Every directed co-graph with at least two vertices has directed twins.*

Proof. Let G be a directed co-graph with at least two vertices. If G has exactly two vertices, then these are twins. So, let G have more than two vertices. Then $G = G_1 \star G_2$ for some directed co-graphs G_1 and G_2 with $|V(G_1)| \geq 2$ or $|V(G_2)| \geq 2$, where $\star \in \{\oplus, \oslash, \otimes\}$. By induction, G_1 or G_2 has twins x, y. Now, by definition of the $\star$-operation, x and y are also twins in G. Thus, every directed co-graph with at least two vertices has a twins as claimed. □

Theorem 3.5.11. *A digraph is a directed co-graph if and only if it can be constructed recursively by taking disjoint union and adding directed twins, starting from a single vertex.*

Proof. Note that we may assume that all graphs considered have at least two vertices. Otherwise, the theorem clearly holds.

First, let G be a directed co-graph. Then, by Proposition 3.5.10, G has twins x and y. Let $G' = G - y$, thus the digraph that emerges when deleting vertex y form G as well as all incident edges of y. Since G' is again a directed co-graph, by induction, G' can be constructed by taking disjoint union and adding twins, starting from single vertices. Since G is obtained from G' by adding twin y to x, G therefore can be constructed by taking disjoint union and adding twins, starting from single vertices, too.

For the other direction, suppose that G can be constructed by taking disjoint union and adding twins, starting from single vertices. We see by induction that G is a directed co-graph. Now, if G is disconnected, then, as every component of G is a directed co-graph, G is a directed co-graph. So, let us assume that G is connected. As every digraph with at most two vertices is a directed co-graph, we may also assume that G has more than two vertices. Now, by the assumption, G has twins x and y so that y is the last vertex adding to $G - y$ in obtaining G. Let $G' = G - y$. Since G' can be constructed by taking disjoint union and adding twins, G' is a directed co-graph by induction. Since G' is connected and has at least two vertices, $G' = G'_1 \star G'_2$ for some directed co-graphs G'_1 and G'_2, where $\star \in \{\oslash, \otimes\}$. Let's assume $x \in G'_1$. We write $G_1 = G[V(G'_1) \cup \{y\}]$ and $G_2 = G'_2$ and notice that, G_1 and G_2 are directed co-graphs.

Then, since x, y are twins in G, $G = G_1 \star G_2$. Hence G is a directed co-graph, and the proof of Theorem 3.5.11 is complete. □

Then, the relation to twin-dh digraphs follows immediately:

Corollary 3.5.12. *Let G be a directed co-graph. Then, G is also twin-distance-hereditary.*

By Lemma 3.5.6 and Theorem 3.5.11, we can further conclude the following result:

Lemma 3.5.13. *Let G be a directed twin-distance-hereditary graph. Then every strong component of G is a directed co-graph.*

Proof. Let H be an induced subdigraph of G that is strongly connected. Then, by Lemma 3.5.6, H is a directed twin-distance-hereditary graph. Thus, there is a directed pruning sequence $S(H)$, that creates H. Assume that there is an element $s_i = (v_i, op_i, v_{a_i})$ in $S(H)$ with operation op_i is a pendant plus (respectively pendant minus) operation. Then, by the allowed operations in twin-dh digraphs, there is no directed path from v_{a_i} to v_i (respectively from v_i to v_{a_i}) in H. This is a contradiction to the fact, that H is strongly connected. Thus, $S(H)$ does not contain any pendant vertex operations. By Theorem 3.5.11 follows, that H is a directed co-graph. □

This lemma admits many algorithmic results. Every digraph problem, which is solvable by considering only the strong components and which is further computable on directed co-graphs, is similarly computable on twin-dh digraphs by Lemma 3.5.13. For example, this holds for several digraph parameters, as we see later on.

With these results it also possible to show that twin-dh digraphs are a subclass of extended directed co-graphs.

Proposition 3.5.14. *Let G be a twin-dh digraph. Then G is also an extended directed co-graph.*

Proof. Let G be a twin-dh digraph. With the following procedure we can get a construction of G with the extended directed co-graph operations. We know from Lemma 3.5.13 that the strong components are directed co-graphs, thus we build the di-co-tree of these components. If a vertex does not belong to any bigger strong component it can be seen as its own strong component. The missing arcs which connect the different strong components in G are built by directed union operations, where we can leave out all arcs except for the arc of the corresponding pendant vertex. □

This result allows us to adopt some results how to solve several graph parameters on this graph class. However, we show that we can even do better on twin-dh digraphs.

Bipartite oriented twin-distance-hereditary graphs

When developing the twin-dh digraphs, we investigated some subclasses, which also can be characterized by forbidden induced subdigraphs. This gives the opportunity for further research e.g. on problems, which are still hard on directed twin-dh digraphs. Bandelt and Mulder [BM86] already considered bipartite distance-hereditary graphs. So first, we introduce the class of bipartite oriented twin-dh graphs. We define the class by limiting the allowed operations in the construction via twins and pendant vertices.

Definition 3.5.15. A graph G is a *bipartite oriented twin-dh graph*, iff it can be constructed by adding false twins and pendant vertices to the single vertex graph. We call the class of bipartite oriented twin-dh graphs BODH for short.

Consequently, if a twin-dh digraph G is bipartite and oriented then, $G \in$ BODH.

Oriented twin-distance-hereditary graphs

We introduce the class of oriented twin-distance-hereditary graphs, where we forbid the bioriented true twin operation.

Definition 3.5.16. A graph G is a *oriented twin-dh graph*, iff it can be constructed by adding false twins, true out-twins, true in-twins and pendant vertices to the single vertex graph. We call the class of oriented twin-dh graphs ODH for short.

Thus, digraph $G \in$ ODH if it is a twin-dh digraph and it contains no bidirectional edges.

Observation 3.5.17. *The following relation between the directed graph classes hold:*

$$BODH \subset ODH \subset DDH$$

and

$$ODH \subset DHD$$

In Figure 3.12 we have examples for digraphs lying in the previous defined classes.

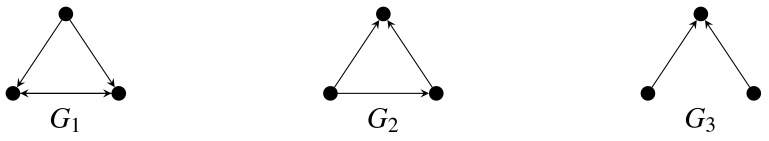


Figure 3.12: These digraphs show the differences of the digraph classes since $G_1 \in$ DDH but $G_1 \notin$ ODH, $G_2 \in$ ODH but $G_2 \notin$ BODH and $G_3 \in$ BODH.

3.5.5 Characterization by Forbidden Induced Subdigraphs

The following part about the characterization of DDH is taken from [KR21]. As already mentioned previously, the new definition is not only based on the property of distance heredity as in the undirected case or in distance-hereditary digraphs. That is, not every digraph which is distance-hereditary, is also a twin-dh digraph. This can be easily shown by e.g. a bioriented path. However, it is possible to give different characterizations of the class DDH by forbidden induced subdigraphs.

Definition 3.5.18. A weakly connected digraph G is a *two-leaves-digraph* if it has at least 4 vertices and if it contains at least two bioriented leaves u, v with $N(u) \neq N(v)$ in $und(G)$, see Figure 3.13.

Theorem 3.5.19. *A digraph G is directed twin-distance-hereditary if and only if it contains none of the following digraphs, see Figure 3.13 as induced subdigraph.*

- $\overrightarrow{C_3}$.
- *any biorientation of the C_n (hole) for $n \geq 5$, domino, house or gem.*
- $H_0, \ldots, H_{27}$.
- *A two-leaves-digraph.*

Proof.
- $\Rightarrow$ None of the digraphs can be constructed with directed twins and directed pendant vertices and the class is hereditary, see Lemma 3.5.6.
- $\Leftarrow$ We proof this by contradiction. Let G be a digraph that does not contain any of the forbidden induced subdigraphs above and let's assume that $G \notin$ DDH. A digraph is not twin-distance-hereditary if it cannot be constructed by the directed twin and pendant vertices operations. We distinguish two cases $G \notin \text{DDH} \wedge un(G) \notin \text{DH}$ and $G \notin \text{DDH} \wedge un(G) \in \text{DH}$. Case one is that G is not twin-dh for structural reasons, thus $G \notin \text{DDH} \wedge un(G) \notin \text{DH}$ which is ensured by the exclusion of any biorientation of the C_n (hole) for $n \geq 5$, domino, house or gem, see Figure 3.13.
  In the other case $G \notin \text{DDH} \wedge un(G) \in \text{DH}$ the digraph is not twin-dh for orientation reasons. This means that there exists a pruning sequence P for $und(G)$ but there is no directed pruning sequence for G because the arcs have a biorientation, which cannot be achieved by the directed twin and pendant vertex operations. By forbidding the two-leaves-digraphs, the pendant vertex operations are not allowed to be bioriented and thus, every undirected pendant vertex operation in P can be replaced by a directed pendant vertex operation. It is left to show that G has as well none of the digraphs of set $\mathcal{H} = \{\overrightarrow{C_3}, H_0, \ldots, H_{27}\}$ as induced subdigraph. Notice that, $\mathcal{H}$ contains every digraph with ≤ 4 vertices that cannot be constructed by the directed twin operations, with no inclusions. If we look at every possibly biorientation of the operations in P, we get any possible directed pruning sequence of G. For every induced subdigraph H of G with ≤ 4 vertices there must exists a biorientation, such that there is a directed pruning sequence, since the set $\mathcal{H}$ is exactly the set of digraphs with ≤ 4 that has no directed pruning sequence. There are no more forbidden induced subdigraphs H' that contains none of the previous excluded digraphs as induced subgraph with more than 4 vertices for the following reason. Assume there is an induced subdigraph H' of G with ≥ 5 vertices which is minimal in the sense that it does not contain a digraph from $\mathcal{H}$ as induced subdigraph and for which there is no directed pruning sequence. Let $V(H') = \{t_1, t_2, u, v, w_1, ..., w_k\}$ with $k \geq 1$ be the vertex set of H', where t_1 and t_2 are twins in the undirected pruning sequence P' of H'. As H' is minimal, every induced subdigraph H^* of H' with 4 vertices is not forbidden. Thus, the different directed neighborhood of t_1 and t_2 must arise by adding the fifth vertex w_1. But if this vertex causes an orientation problem in $H[\{t_1, t_2, u, v, w_1\}]$ then this vertex also causes an orientation problem in $H[\{t_1, t_2, u, w_1\}]$ which would build a forbidden induced subdigraph with 4 vertices. We end up in the same problem if we chose any other two twins. Thus, there cannot be a forbidden induced subdigraph with more than 4 vertices for which there is no directed pruning sequence, if an undirected pruning sequence exists and $G \in$ DDH.

This shows the statement of the theorem. □

To get a better understanding of the construction of the forbidden induced subdigraphs $H_0, \ldots, H_{27}$ we group them as follows. In none of them we can find directed twins, but the undirected versions of them are distance hereditary.

- $H_0, \ldots, H_5$: Digraphs with 4 or less vertices with $und(G) = C_4$ which are strongly connected.

- $H_6, \dots, H_9$: Digraphs with 4 vertices with $und(G) = C_4$ which are not strongly connected.
- $H_{10}, \dots, H_{19}$: Digraphs with 4 vertices with $und(G) = C_4$ with an additional single diagonal edge.
- $H_{20}, \dots, H_{26}$: Digraphs with 4 vertices with $und(G) = C_4$ with an additional bidirectional diagonal edge.
- H_{27}: Forbidden orientation of the K_4.

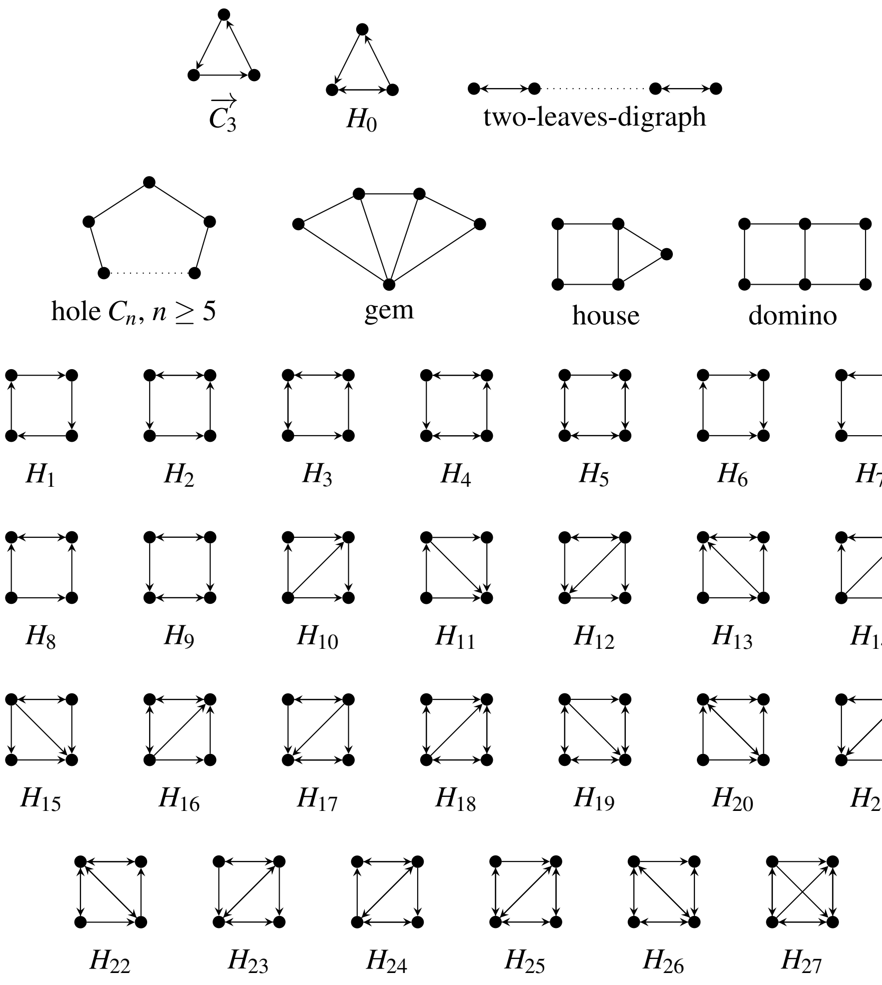


Figure 3.13: Forbidden induced sub(di)graphs.

We also give a characterization for the subclasses BODH and ODH. In the undirected case there is the following characterization for bipartite distance-hereditary graphs.

Lemma 3.5.20 ([BM86]). *An undirected graph G is bipartite distance hereditary if and only if it has no triangle, domino or cycle C_n with $n \geq 5$ as induced subgraph.*

The class BODH is hereditary for the same reasons as its superclass such that we can characterize this class by a set of forbidden induced subdigraphs.

Theorem 3.5.21. *$G \in BODH$ if and only if it contains none of the following graphs, see Figure 3.13, as induced subdigraph.*

- *any orientation of the C_n (hole) for $n \geq 5$, domino or triangle.*
- $\overleftrightarrow{K_2}, H_6, \overrightarrow{C_4} (= H_1)$

Proof.

- $\Rightarrow$ None of the graphs can be constructed with the operations from Definition 3.5.15 and the class is hereditary.
- $\Leftarrow$ We call the set of forbidden induced subdigraphs $\mathcal{F}_{BODH}$. Let G be a digraph with no induced subgraph $H \in \mathcal{F}_{BODH}$, from Theorem 3.5.21. We divide $\mathcal{F}_{BODH}$ into two distinct sets $\mathcal{F}_{BO}$ and $\mathcal{F}_{DH}$ such that $\mathcal{F}_{BODH} = \mathcal{F}_{BO} \cup \mathcal{F}_{DH}$ with $\mathcal{F}_{DH} = \{H \mid H$ is an orientation of a triangle, hole or domino$\}$ and $\mathcal{F}_{BO} = \{\overleftrightarrow{K_2}, \overrightarrow{C_4}, H_6\}$. On this way we divide the forbidden induced subdigraphs in two sets: The forbidden subdigraphs H for which there is not even a pruning sequence for $und(H)$ (structural reasons) and the forbidden subdigraphs F for which there is a pruning sequence for $und(F)$ but there is no twin operation which allows the orientations (orientation reasons). Since orientations of holes with at least 5 vertices are forbidden, dominoes or triangles, $und(G)$ is bipartite distance hereditary, see Lemma 3.5.20. Thus, there exists a pruning sequence P for the underlying undirected graph $und(G)$ using false twins and pendant (plus or minus) vertices. It is left to show that there exists also a directed pruning sequence $\overrightarrow{P}$, by excluding the graphs from $\mathcal{F}_{BO}$. By forbidding the $\overleftrightarrow{K_2}$ there are no bidirectional edges in the digraph. Since there are there are no holes with $n = 3$ or $n \geq 5$ the graph only has tree-like structures with arcs (no bidirectional) and cycles on 4 vertices. The tree-like structures can be constructed by the pendant plus and pendant minus vertices. The remaining graph consists of cycles of length 4. There exists 3 different non-isomorphic digraphs on 4 vertices with 4 oriented edges. Since the C_4 and the F_6 are forbidden induced subdigraphs, the only possible orientation is the one from Figure 3.14, which consist of false twins. Thus, we can always create G with a directed pruning sequence consisting of false twins and pendant plus or minus vertices, such that $G \in BODH$.

□

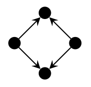


Figure 3.14: Oriented graph G with $und(G)$ is a C_4 and there exist directed twins: Only allowed non-tree-like structure in BODH having false twins.

Theorem 3.5.22. *$G \in ODH$ if and only if it contains none of the following graphs, see Figure 3.13, as induced subdigraph.*

- *$\overrightarrow{C_n}$ for $n \in \{2,3,4\}$.*
- *any orientation of the C_n (hole) for $n \geq 5$, domino, house or gem.*
- *H_6, H_{10}, H_{11}.*

Proof.

- $\Rightarrow$ None of the graphs can be constructed with the operations from Definition 3.5.16 and the class is hereditary.
- This direction of the proof can be conducted exactly like the proof of Theorem 3.5.19. The only difference is that we forbid the $\overrightarrow{C_2} = \overleftrightarrow{K_2}$ which is a bidirectional edge, such that we can omit all graphs that contain bidirectional edges, which leads us to the which leads us to that exactly the above given set of forbidden induced digraphs.

□

To sum up, the class of twin-dh digraphs is a superclass of directed co-graphs and when excluding the bioriented true twin operation, it is a subclass of distance-hereditary digraphs, defined in [LS10]. Further, the class is a subclass of extended directed co-graphs which allows us to adopt interesting results, which we see in Chapter 4. One of these results is that twin-dh digraphs have bounded directed clique-width. Due to the unbounded directed clique-width of extended directed co-graphs, twin-dh digraphs exhibit properties which allow supplemental results such that an investigation of this class is advisable. The property that every strong components is a directed co-graph is helpful in the computation of solutions for several problems.

3.6 Directed Versions of Series-parallel Graphs

The content of the following two subsections is taken from [GKR20b] and [GKL20].

3.6.1 MSP-digraphs

We recall the definition of minimal vertex series-parallel digraphs from [BJG18] which is based on [VTL82]. The motivation of this class is based in the superclass of series-parallel digraphs, which are are exactly the digraphs whose transitive closure equals the transitive closure of a minimal series-parallel digraph, see Subsection 3.6.2. By [BJG18, Section 11.1] msp-digraphs are useful for modeling flow diagrams and dependency charts and they are used in applications for scheduling under constraints.

First, we introduce two operations for two vertex-disjoint digraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Let O_1 be the set of sinks in G_1 and I_2 be the set of sources in G_2.

- The *parallel composition* of G_1 and G_2, denoted by $G_1 \cup G_2$, is the digraph with vertex set $V_1 \cup V_2$ and arc set $E_1 \cup E_2$.

- The *series composition* of G_1 and G_2, denoted by $G_1 \times G_2$ is the digraph with vertex set $V_1 \cup V_2$ and arc set $E_1 \cup E_2 \cup (O_1 \times I_2)$ with $O_1 \times I_2 = \{(u,v) \mid u \in O_1 \text{ and } v \in I_2\}$.

Definition 3.6.1 (Minimal series-parallel digraphs)**.** The class of *minimal series-parallel digraphs*[6], *msp-digraphs* for short, is recursively defined as follows.

1. Every digraph on a single vertex $(\{v\},\emptyset)$, denoted by v, is a minimal series-parallel digraph.
2. If G_1 and G_2 are vertex-disjoint minimal series-parallel digraphs, then
   (a) the parallel composition $G_1 \cup G_2$ and
   (b) then series composition $G_1 \times G_2$ are minimal series-parallel digraphs.

The class of minimal series-parallel digraphs is denoted by MSP.

An expression X using the operations of Definition 3.6.1 is called an *msp-expression* and digraph(X) is the corresponding digraph. For an illustration of such an expression see the following example.

Example 3.6.2. 1. The msp-expression

$$X = ((v_1 \cup v_2) \times (v_3 \cup v_4)) \tag{3.1}$$

defines digraph(X) shown in Figure 3.15.

2. The msp-expression

$$X = (((v_1 \times v_2) \cup (v_3 \times v_4)) \times (v_5 \times v_6)) \tag{3.2}$$

defines digraph(X) shown in Figure 3.16.

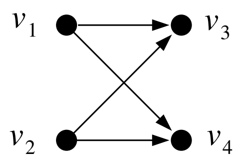


Figure 3.15: Digraph in Example 3.6.2(1.).

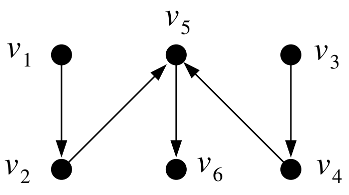


Figure 3.16: Digraph in Example 3.6.2(2.).

For every msp-digraph we can define a tree structure T, which is denoted as *msp-tree*. In [VTL82], the tree-structure for an msp-digraphs is denoted as binary decomposition tree. The leaves of an msp-tree represent the vertices of the digraph and the inner vertices of the msp-tree correspond to the operations applied on the subexpressions defined by the subtrees.

Observation 3.6.3 ([VTL82])**.** *For every minimal series-parallel digraph G one can construct an msp-tree in time $O(n+m)$, where $m = |E(G)|$ and $n = |V(G)|$.*

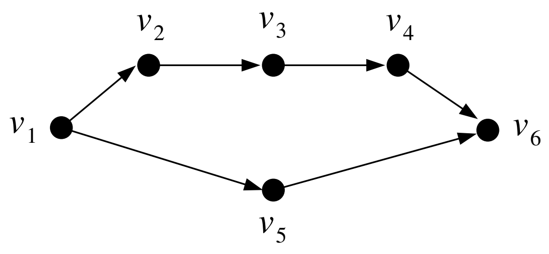


Figure 3.17: An msp-digraph with msp-expression $X = (v_1 \times (((v_2 \times v_3) \times v_4) \cup v_5)) \times v_6$, see [GKL20].

The class of msp-digraphs is not closed under taking induced subdigraphs, as by removing vertex v_3 from digraph(X) in Figure 3.6.1, we get a digraph which is no msp-digraph. The class of oriented complete bipartite graphs, denoted by $\overrightarrow{K_{n,m}}$, is a subclass of msp-digraphs. We have already seen that not every orientation of a tree is a msp-digraph, however a certain orientation of trees form a subclass.

Observation 3.6.4. *Every in- or out-tree is a minimal series-parallel digraph.*

3.6.2 Series-parallel Digraphs

Definition 3.6.5 (Series-parallel digraphs)**.** Series-parallel digraphs are exactly the digraphs whose transitive closure equals the transitive closure of some minimal series-parallel digraph. The class of series-parallel digraphs is denoted by SPD.

Theorem 3.6.6 ([VTL82])**.** *An acyclic digraph is series-parallel, if and only if its transitive closure is N-free, see Figure 3.5.*

From an algorithmic point of view series-parallel digraphs are interesting since several hard graph problems can be solved in polynomial time by dynamic programming along the tree structure of the input graph, see [MS77, Ste85, Ren86].

3.6.3 Series-parallel Partial Order Digraphs

We take a look at the definitions of from [BJG18] that base on [VTL82]. The content about series-parallel partial order digraphs is taken from [GKR21c] and [GKR20b]. A series-parallel partial order is a partially ordered set $(X, \leq)$ that is constructed by the series composition and the parallel composition operation starting with a single element.

- Let $(X_1, \leq)$ and $(X_2, \leq)$ be two disjoint series-parallel partial orders, then distinct elements $x, y \in X_1 \cup X_2$ of a series composition[7] have the same order they have in X_1 or X_2. Respectively, this holds if both of them are from the same set, and $x \leq y$ if $x \in X_1$ and $y \in X_2$.

[6]also known as *minimal vertex series-parallel digraphs*

[7]Notice that, the series composition in this case corresponds to the order composition in the definition of directed co-graphs.

- Two elements $x, y \in X_1 \cup X_2$ of a parallel composition are comparable if and only if both of them are in X_1 or both in X_2, while they keep their corresponding order.

Definition 3.6.7 (Series-parallel partial order)**.** The class of *series-parallel partial orders*, SPO for short, over a set X is recursively defined as follows.

1. Every single element $(\{x\}, \emptyset)$, $x \in X$, is a *series-parallel partial order*.
2. If $(X_1, \leq)$ and $(X_2, \leq)$ are series-parallel partial orders over set X, such that $X_1 \subseteq X$, $X_2 \subseteq X$, and $X_1 \cap X_2 = \emptyset$, then
    (a) the series composition of $(X_1, \leq)$ and $(X_2, \leq)$ and
    (b) the parallel composition of $(X_1, \leq)$ and $(X_2, \leq)$ are *series-parallel partial orders*.

Example 3.6.8. The following partially ordered sets are series-parallel partial orders over set $\{x_1, x_2, x_3, x_4\}$.

- The parallel composition of $(\{x_1\}, \emptyset)$ and $(\{x_3\}, \emptyset)$ leads to the series-parallel partial order $(\{x_1, x_3\}, \emptyset)$.
- The series composition of $(\{x_2\}, \emptyset)$ and $(\{x_4\}, \emptyset)$ leads to the series-parallel partial order $(\{x_2, x_4\}, \{(x_2, x_4)\})$.
- The series composition of $(\{x_1, x_3\}, \emptyset)$ and $(\{x_2, x_4\}, \{(x_2, x_4)\})$ leads to the series-parallel partial order $(\{x_1, x_2, x_3, x_4\}, \{(x_2, x_4), (x_1, x_2), (x_1, x_4), (x_3, x_2), (x_3, x_4)\})$.

Definition 3.6.9 (Series-parallel partial order digraphs)**.** A *series-parallel partial order digraph* $G = (V, E)$ is a digraph, where $(V, \leq)$ is a series-parallel partial order and $(u, v) \in E$ if and only if $u \neq v$ and $u \leq v$. We denote the class of series-parallel partial order digraphs by SPO.

3.6.4 Properties of Series-parallel Partial Order Digraphs

Bechet et al. showed in [BdGR97] the following property of directed co-graphs in relation to series-parallel partial order digraphs.

Lemma 3.6.10 ([BdGR97])**.** *For every directed co-graph G it holds that the asymmetric part of G is a series-parallel partial order digraph and for the symmetric part the underlying undirected graph a co-graph.*

The class of series-parallel partial ordered digraphs is equal to the class of oriented co-graphs since they have exactly the same recursive structure. Thus, this lemma is easy to prove with the following idea.

- **symmetric part:** Exchange each order composition with a directed union composition. Since there are no more oriented arcs left, this tree represents a co-graph.

- **asymmetric part:** Exchange each series composition with a directed union composition. Since there are no more bidirectional edges left, this tree represents an oriented co-graph, e.g. a series-parallel partial order digraph.

Example 3.6.11. The series-parallel partial orders given in Example 3.6.8 show that the digraph shown in Figure 3.18 is a series-parallel partial order digraph.

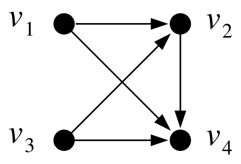


Figure 3.18: Directed co-graph with di-co-expression $X = ((v_1 \oplus v_3) \oslash (v_2 \oslash v_4))$.

Comparing the definitions of the order composition of oriented co-graphs with the series composition of series-parallel partial order digraphs and the disjoint union composition of oriented co-graphs with the parallel composition of series-parallel partial order digraphs, see Figure 3.18 and Example 3.6.8, we obtain the following result.

Observation 3.6.12. *The sets OC and SPO are equal.*

In Subsection 3.7 we give an overview about the relation between the presented digraph classes.

3.6.5 ESP-digraphs

Undirected series-parallel graphs are graphs with two distinguished vertices called terminals, which are formed recursively by parallel and series compositions. In contrast to the classes introduced earlier, we now start with a single edge instead of a single vertex. This subsection is taken from [LGK21]. A multidigraph is a digraph that can have multiple edges between two vertices. More precisely, not only bidirectional edges are allowed but also multiple edges with the same start and end vertex.

Let $G_1 = (V_1, A_1)$ and $G_2 = (V_2, A_2)$ be two vertex-disjoint multidigraphs such that each of them has exactly one source and one sink.

- The *parallel composition* $G_1 \cup G_2$ identifies the source of G_1 with the source of G_2 and the sink of G_1 with the sink of G_2.
- The *series composition* $G_1 \times G_2$ identifies the sink of G_1 with the source of G_2.

Edge series-parallel digraphs have been defined originally as edge series-parallel multidigraphs, from [VTL82].

Definition 3.6.13 (Edge Series-Parallel Multidigraphs)**.** The class of *edge series-parallel multidigraphs*, *esp-digraphs* for short, is recursively defined as follows.

(i) Every digraph of two distinct vertices joined by a single arc $(\{u,v\},\{(u,v)\})$, denoted by (u,v), is an *esp-digraph.*

(ii) If $G_1 = (V_1,A_1)$ and $G_2 = (V_2,A_2)$ are vertex-disjoint minimal edge series-parallel multidigraphs, then

  (a) the parallel composition $G_1 \cup G_2$ is an *esp-digraph* and

  (b) the series composition $G_1 \times G_2$ is an *esp-digraph*.

An expression X using the operations of Definition 3.6.13 is called an *esp-expression* and digraph(X) is the defined digraph. For a better understanding we now give an example of such an expression.

Example 3.6.14. The esp-expression

$$X_e = ((v_1,v_2) \times (((v_2,v_3) \times ((v_3,v_4) \times (v_4,v_5))) \cup (v_2,v_5))) \times (v_5,v_6)$$

defines the esp-digraph shown in Figure 3.19.

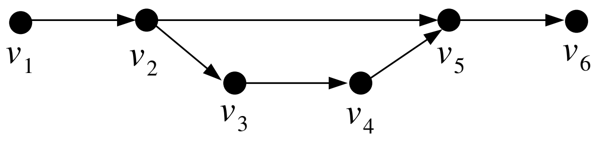


Figure 3.19: Digraph(X_e) in Example 3.6.14.

As for the other digraph classes with a recursive structure, we can define a tree structure for every esp-digraph, denoted as *esp-tree*. The leaves of the esp-tree represent the arcs of the digraph and the inner nodes of the esp-tree correspond to the operations applied on the subexpressions which are defined by the subtrees. For a vertex u of an esp-tree T we denote by $T(u)$ the subtree rooted at u and by $X(u)$ the *sub-expression* defined by $T(u)$. For every esp-digraph the construction of an esp-tree is possible in linear time, see [Val78]. These graphs are also subject in [HY87] where they appear as two-terminal series-parallel (TTSP) graphs and where a parallel algorithm for recognizing directed series-parallel graphs is shown. Additionally, in [Epp92] we can find an improved parallel algorithm for recognizing directed (and undirected) series-parallel graphs.

There is a relation between esp and msp-digraphs. In a *line digraph* $LD(G)$ of a digraph G there exists a vertex for every arc in G. More precisely, there is an arc (u,v) in $LD(G)$ if and only if $u = (x,y)$ and $v = (y,z)$ for the vertices $x,y,z \in V(G)$ [HN60]. In this sense we call digraph G the *root digraph* of $LD(G)$.

Lemma 3.6.15 ([VTL82])**.** *An acyclic multidigraph G with a single source and a single sink is an esp-digraph if and only if its line digraph $LD(G)$ is an msp-digraph.*

Example 3.6.16. The msp-digraph G, which is defined in Figure 3.6.1 defines the line digraph of esp-digraph G_e defined in Example 3.6.14, see Figure 3.19.

Since esp-digraphs can have multi-edges, this class can not be a subclass of the classes introduced earlier.

Example 3.6.17. Let G be a digraph and e_1 is a single arc in $E(G)$. Then $X = (e_1 \times e_1) \cup e_1$ is an esp-digraph but not an msp-digraph. The digraph consisting of a single vertex is an msp-digraph but no esp-digraph.

From the digraphs in the previous example we can conclude the following observation about the relation between msp and esp-digraphs, even when excluding multi-edges.

Observation 3.6.18. *The class of msp-digraphs and esp-digraphs are not comparable.*

As well the bidirectional arcs, which are allowed in esp-digraphs but not in msp-digraphs can be mentioned at this point. Later we also need the following properties of esp-digraphs.

Observation 3.6.19 ([LGK21])**.** *Let G be an esp-digraph. Then, it holds that G has exactly one source and exactly one sink.*

Proposition 3.6.20. *Let G be an esp-digraph. Then, it holds that $und(G)$ is a series-parallel graph.*

Proof. For every start digraph $G = (\{v,u\},(u,v))$, $und(G)$ is series-parallel graph. Further, we can replace every parallel composition by a parallel composition in the undirected case, and every series composition by a series composition in the undirected case. The sources in G are also sources in $und(G)$, same holds for sinks. □

3.7 Hierarchy

In Figure 3.20 we summarize the relation of directed co-graphs, series-parallel digraphs and related graph classes. The directed edges represent the existing relations between the graph classes, which follow by their definitions. For the relations to further graph classes we refer to [BJG18, Figure 11.1]. Since a skew bipath (which is not a distance-hereditary digraph) is is an extended directed co-graph and a $\overrightarrow{C_3}$ is a distance-hereditary digraph but not an extended directed co-graph these classes are incomparable. Additionally, DAGs are not a subclass of DDH, since every acyclic orientation of a C_n with $n \geq 5$ is a DAG but not in DDH. The other direction is trivially not satisfied, since DDH includes bidirectional edges.

$$\begin{tikzcd}
 & & & EDC \arrow[dl] \arrow[dr] & & & \\
 & & DAG \arrow[dll] \arrow[dl] \arrow[d] & & DDH \arrow[dl] \arrow[dr] & & DHD \arrow[dl] \\
ESP^{*} & OT & SPD \arrow[dl] \arrow[dr] & DC \arrow[d] & & ODH \arrow[dll] \arrow[dr] & \\
 & MSP & & OC = SPO \arrow[d] & & & BODH \\
 & & & OTD \arrow[d] & & & \\
 & & & TT & & &
\end{tikzcd}$$

Figure 3.20: Hierarchy between the digraph classes: A directed edge from class A to class B indicates that $B \subseteq A$. Two classes A and B are incomparable if there is neither a directed path from A to B, nor a directed path from B to A. ESP^{*} is the class of esp-digraphs without multi-edges, which we leave out to enable comparability.

4 Directed Graph Parameters on Special Digraph Classes

Directed graph parameters are often hard to compute in general, however they are useful in parameterized complexity. This leads to the idea of looking at directed width-measures of special digraph classes, which we do in this chapter. First, we give some bounds for several directed width-measures on semicomplete digraphs. Then, we move on showing the efficient computation of various directed width-measures on (extended) directed co-graphs and twin-dh digraphs.

4.1 Bounds of Digraph Parameters on Semicomplete Digraphs

The content of this section is taken from [GKRW21]. The rediscovery of path-width and tree-width in the graph minors project by Robertson and Seymour [RS86] has led to a wide range of algorithmic results. In the wake of this success several possible generalizations to directed graphs have since emerged, among which are directed path-width (d-pw), directed tree-width (d-tw) [JRST01b], DAG-width (dagw) [BDH$^+$12] and Kelly-width (kw) [HK08].

While all of these parameters are related, directed path-width and directed tree-width are not parametrically equivalent to either of the other parameters and the equivalence of DAG-width and Kelly-width is an open conjecture.

All these width parameters correspond to different variants of so-called cops and robber games. Width parameters corresponding to variants of the cops and robber game have the inherent advantage of coming with an XP-time (approximation) algorithm for finding a decomposition of (almost) optimal width. They also tend to correlate with structural properties and thus, as exemplified by tree-width, make for great tools for structure theory. However, there exists strong evidence that for digraphs no such parameter can, in addition to these advantages, replicate the algorithmic power of tree-width in undirected graphs [GHK$^+$16].

We show that on semicomplete digraphs, all of the path-width and tree-width inspired parameters are equivalent. Indeed, all of these equivalences are realized by relatively tame functions obtained without complicated proofs.

4.1.1 Comparison of Directed Graph Parameters on General Digraphs

We start with the comparisons between different parameters and thus, the current landscape of bounding functions between these parameters on general digraphs.

Proposition 4.1.1. *Let G be a digraph and $f,g \in \{d\text{-}pw, d\text{-}tw, dagw, kw, d\text{-}lcw, d\text{-}cw\}$. If $f(G) \leq k$, then $g(G) \leq h'_{f,g}(k)$ where $h'_{f,g}\colon \mathbb{N} \to \mathbb{N}$ is given by Table 4.1 if the functions exist.*

Proof. 1. d-pw is unbounded in terms of kw: In [BJG18] the example of complete complete biorientation $\overleftrightarrow{T_h}$ of an undirected binary tree T_h of height h is considered. Since the undirected path-width of T_h is h it follows that $\overleftrightarrow{T_h}$ has directed path-width h. Furthermore, $\overleftrightarrow{T_h}$ has Kelly-width 2 since a directed elimination ordering t of width 1 can be obtained by any level order starting with the leaves of T_h.

2. d-pw is unbounded in terms of d-tw: Holds with the example from 1 which is inspired by the undirected comparisons of path-width and tree-width. Increasing h, the directed tree-width is 1, while the directed path-width increases.

3. d-tw is bounded by d-pw: This follows immediately from the definition.

4. d-pw, d-tw, dagw and kw are unbounded in terms of d-lcw and thus in d-cw: The set of all bioriented cliques is a counterexample.

5. kw is unbounded in terms of d-tw: As an example consider a binary tree, where all edges are oriented from the root to the leaves. Additionally, every vertex has a backward edge to each of its predecessors on the unique path from the root to itself.

6. d-cw and thus also d-lcw is unbounded in terms of d-pw, d-tw, dagw and kw. An acyclic orientation of a grid graph is a counterexample, see Figure 2.6.

7. d-lcw is unbounded in terms of d-cw: In Lemma 11 of [GW05] it has been shown that for $G_1 = \bullet$ and $G_{i+1} = (G_i \cup G_i) \times (G_i \cup G_i)$ for $i \geq 1$, graph G_i has linear NLC-width at least i. The proof idea can be used to show that for $G_1 = \bullet$ and $G_{i+1} = (G_i \oslash G_i) \otimes (G_i \oslash G_i)$ for $i \geq 1$, digraph G_i has directed linear NLC-width at least i. All graphs G_i are directed co-graphs which implies that they have directed clique-width at most 2, see [GWY16]. Since directed linear clique-width is greater or equal to directed linear NLC-width [GR19a], the result follows.

8. d-cw is bounded by d-lcw: This follows immediately from the definition.

□

Notice that, Proposition 4.1.1 contains in particular the known fact that directed path-width poses as an upper bound for all tree-width inspired width parameters. Moreover, on semicomplete digraphs, by Proposition 4.1.7 it also is an upper bound on directed clique-width. It therefore suffices, towards a proof of Theorem 4.1.8, to establish upper bounds on directed path-width in terms of directed tree-width, DAG-width, and Kelly-width, as well as proving that Proposition 4.1.7 can be extended to also include directed linear clique-width.

g \ f	d-pw	d-tw	dagw	kw	d-lcw	d-cw
d-pw	k	∞	∞ [BDH+12]	∞	∞	∞
d-tw	k	k	$3k+1$ [BDH+12]	$6k-2$ [HK08]	∞	∞
dagw	$k+1$ [BDH+12]	∞ [BDH+12]	k	$72k^2$ [AKK+15]	∞	∞
kw	$k+1$ [GHK+14]	∞	??? [HK08]	k	∞	∞
d-lcw	∞	∞	∞	∞	k	∞
d-cw	∞	∞	∞	∞	k	k

Table 4.1: Relations between digraph parameters on **digraphs**. The parameter of the left column is bounded by the respective parameter of the top row by the specified function where k is the corresponding width. We use '∞' if the relation is unbounded, that is if $h'_{f,g}$ does not exist. The cell with '???' represents the remaining relation of the conjecture on DAG-width and Kelly-width.

4.1.2 DAG-width and Directed Path-width on Semicomplete Digraphs

As a first step towards Theorem 4.1.8 we show that DAG-width plus 1 and directed path-width are equal on the class of semicomplete digraphs. Before proving this relation formally, we give brief interjection about the intuitive idea behind it. The definition of the compositions of directed path-width and DAG-width are quiet alike, except for the path and the DAG-structure. In both decompositions we can read informally which vertex is reachable from which other vertex in the corresponding digraph. In the DAG-decomposition we have the possibility to branch, such that two vertices which do not share edges or reachability can occur in separate branches and we do not have an increase of the width due to preserve condition $(dagw\text{-}2)$. But in a semicomplete digraph every vertex has an edge to every other vertex, which means that all vertices are related to each other. This relation must be represented in the DAG-decomposition. By this, it makes no sense to branch, such that we end up in a path as a decomposition structure. In the following we formalize this idea and prove it.

This result also leads to the fact that computing DAG-width of a semicomplete digraph is in NP.

This later fact might be of independent interest since DAG-width is PSPACE-complete in general [AKK+15], but, it is one of only few known parameters from the tree-width inspired family which allows for an efficient solving of parity games [BDHK06].

We look at a nice DAG-decomposition of a digraph D. Since deleting transitive edges from D does neither destroy any of the properties of a DAG-decomposition, nor increase the width of the DAG-decomposition, we get the following property.

Lemma 4.1.2. *If digraph G has a DAG-decomposition of width k, it also has a nice DAG-decomposition (D,X) of width k such that D has no transitive edges.*

Since the DAG-width of a graph is also known as being the non-linear path-width of a digraph the following result seem very natural.

Theorem 4.1.3. *For every semicomplete digraph G it holds that d-pw$(G) \leq$ dagw$(G)-1$.*

Proof. Let G be a semicomplete digraph and let $(D,\mathcal{X})$ be a nice DAG-decomposition for G of width k with digraph D, vertex set V_D and $\mathcal{X}=\{X_u \mid u\in V_D\}$. By Lemma 4.1.2 we can assume that D has exactly one source, every vertex in D has at most two successors and no transitive edges. We show that in case D is not a path, we can convert it into a path without increasing the width. Assume D is not a path. For any vertex r let V_{D_r} is the set of vertices of D which are reachable from r. Let D_t be the maximal subdigraph of D with unique source t. Consider vertex $q\in V_D$ with two successors s and t. We differentiate three cases: All vertices from G which are in bags of D_s are also in the bags of D_t (Case 1.a), the opposite inclusion (Case 1.b) or, at last none of these inclusions (Case 2) occur.

Case 1.a: $(\bigcup_{u\in V_{D_s}} X_u)\cup X_q \subseteq (\bigcup_{u\in V_{D_t}} X_u)\cup X_q$.

In order to define a new DAG-decomposition $(D',\mathcal{X}')$ for G, we simply remove all vertices $V_{D_s}\backslash V_{D_t}$ from D and forget all bags associated with removed vertices. We now show that $(D',\mathcal{X}')$ is a DAG-decomposition for G by checking the conditions of Definition 2.2.14.

- (dagw-1) Is satisfied since

$$\bigcup_{u\in V_{D'}} X_u = \bigcup_{u\in V_D\backslash V_{D_s}} X_u \cup \bigcup_{u\in V_{D_t}} X_u \overset{(*)}{\supseteq} \bigcup_{u\in V_D\backslash V_{D_s}} X_u \cup \bigcup_{u\in V_{D_s}} X_u = \bigcup_{u\in V_D} X_u = V_G$$

The inclusion in $(*)$ holds by assumption of case $1a)$ since $q\in V_D\backslash V_{D_s}$.

- (dagw-2) is still satisfied since for every $a,b,c\in V_{D'}$ it holds that if $a\preccurlyeq_{D'} b\preccurlyeq_{D'} c$ then

$$X'_a\cap X'_b = X_a\cap X_c \subseteq X_b = X'_b$$

- (dagw-3) Let $(a,b)\in E_{D'}$, then it follows that $(a,b)\in E_D$. Therefore, it must hold that $X_a\cap X_b$ guards $X_{\succcurlyeq b}\backslash X_a$. It holds that $X'_a = X_a$ and $X'_b = X_b$. Further, $X'_{\succcurlyeq b}$ is the union of all bags of vertices that we can reach from vertex b in D', such that $X'_{\succcurlyeq b} = \bigcup_{b\preccurlyeq_{D'} u} X_u$.

  (i) If $b\preccurlyeq_{D'} t$, then:

$$\begin{aligned}X'_{\succcurlyeq b} &= \bigcup_{b\preccurlyeq_{D'}u\preccurlyeq_{D'}t} X_u \cup \bigcup_{t\preccurlyeq_{D'}u} X'_u = \bigcup_{b\preccurlyeq_D u\preccurlyeq_D t} X_u \cup \bigcup_{t\preccurlyeq_D u} X_u\\ &(\text{since } X_q\subseteq \bigcup_{b\preccurlyeq_D u\preccurlyeq_D t} X_u)\\ &= \bigcup_{b\preccurlyeq_D u\preccurlyeq_D t} X_u \cup \bigcup_{t\preccurlyeq_D u} X_u \cup \bigcup_{s\preccurlyeq_D u} X_u = \bigcup_{b\preccurlyeq_D u} X_u = X_{\succcurlyeq b}\end{aligned}$$

  (ii) Else $t\prec_{D'} b$, then: Since every successor of b in D is also in D' it holds that

$$X'_{\succcurlyeq b} = \bigcup_{b\preccurlyeq_{D'}u} X_u = \bigcup_{b\preccurlyeq_D u} X_u = X_{\succcurlyeq b}$$

  This leads to $X'_a\cap X'_b = X_a\cap X_b$ guards $X'_{\succcurlyeq b}\backslash X'_a = X_{\succcurlyeq b}\backslash X_a$.

Thus, all requirements of a DAG-decomposition are met by $(D', \mathcal{X}')$.

Case 1.b: $(\bigcup_{u \in V_{D_t}} X_u) \cup X_q \subseteq (\bigcup_{u \in V_{D_s}} X_u) \cup X_q$ can be handled analogously to case 1.a.

Case 2: $(\bigcup_{u \in V_{D_s}} X_u) \cup X_q \nsubseteq (\bigcup_{u \in V_{D_t}} X_u) \cup X_q$ and $(\bigcup_{u \in V_{D_t}} X_u) \cup X_q \nsubseteq (\bigcup_{u \in V_{D_s}} X_u) \cup X_q$. More informally, this means that there exist vertices from G that are only represented in bags of D_s but not in bags of D_t. We show now, that this case cannot occur. There are x, y such that

$$x \in X_q \cup \bigcup_{u \in V_{D \geq s}} X_u, x \notin X_q \cup \bigcup_{u \in V_{D \geq t}} X_u \tag{4.1}$$

$$y \notin X_q \cup \bigcup_{u \in V_{D \geq s}} X_u, y \in X_q \cup \bigcup_{u \in V_{D \geq t}} X_u \tag{4.2}$$

Since G is semicomplete, there is an arc between x and y in G. W.l.o.g. let $(x,y) \in E_G$. By the connectivity property given by (dagw-2) it holds that $x, y \notin \bigcup_{u \preccurlyeq_D q} X_u$, since $x, y \notin X_q$. Let $w \in V_D, x \in X_w, x \notin X_u$ and $u \preccurlyeq_D w$. As equation (4.1) holds, this leads to $s \preccurlyeq_D w$. By (dagw-3) it further holds that $X_{w'} \cap X_w$ guards $X_{\succcurlyeq w} \backslash X_{w'}$ for a predecessor w' of w in D with $w' \neq s$. This means that for all $(z, z') \in E_G$ with $z \in X_{\succcurlyeq w} \backslash X_{w'}$ it holds that $z' \in (X_{\succcurlyeq w} \backslash X_{w'}) \cup (X_{w'} \cap X_w)$.
As assumed before, it holds that $(x,y) \in E_G$ with $x \in X_{\succcurlyeq w} \backslash X_{w'}$. By equation (4.2) it holds that $y \notin X_{w'} \cap X_w \Rightarrow y \in X_{\succcurlyeq w} \backslash X_{w'}$. By equation (4.2) it holds that $y \notin X_{w'} \Rightarrow y \in X_{\succcurlyeq w} = \bigcup_{w \preccurlyeq_D u} X_u$. But since $s \preccurlyeq_D w$ it holds that $\bigcup_{w \preccurlyeq_D u} X_u \subseteq \bigcup_{s \preccurlyeq_D u} X_u$. This contradicts that by equation (4.2) it holds that $y \notin \bigcup_{s \preccurlyeq_D u} X_u$. This leads to the conclusion that case 2 cannot occur.

Consequently, starting at the root, we can transform every DAG D of a DAG-decomposition of the semicomplete digraph G into a directed path. Since directed path-width is exactly the path variant of DAG-width, d-pw$(G) \leq$ dagw$(G) - 1$ holds. □

By Theorem 4.1.3 we can conclude that on semicomplete digraphs, DAG-width plus 1 and path-width are equal.

Corollary 4.1.4. *For every semicomplete digraph G it holds that*

$$\text{d-pw}(G) + 1 = \text{dagw}(G).$$

4.1.3 Escaping Pursuit in the Jungle: Directed Path-width, Directed Tree-width and Kelly-width

Fradkin and Seymour [FS13] gave a description of semicomplete digraphs of bounded directed path-width. Indeed, they proved that every semicomplete digraph of huge directed path-width must contain a subdivision of a large bioriented clique [FS13]. While this result immediately implies that directed path-width acts, parametrically, as a lower bound for all tree-width inspired directed width measures discussed here, the proof uses a Ramsey argument and thus, for G to contain a subdivision of the complete biorientation of K_t, the directed path-width must be exponential in t. However, Fradkin and Seymour introduced another obstruction to small directed path-width on semicomplete digraphs which is similar to the idea of well linked sets. With a bit of more careful analysis we are able to obtain the quadratic bounds of Theorem 4.1.8.

Notice that, [FS13] could also be used for comparisons between directed path-width and DAG-width, but this would only lead to equivalence between those parameters, whereas we could prove equality (plus 1).

Two vertices u,v are *k-connected*, if there are at least k internally-disjoint paths from u to v and from v to u. For digraph $G=(V,E)$ a set $U \subseteq V$ is a *k-jungle* in G if $|U|=k$ and for all $u,v \in U$ it holds that u and v are k-connected.

For both, directed tree-width and Kelly-width, we show that the existence of a $k+1$-jungle is enough to ensure a winning strategy for the robber against k cops in the respective variants of of cops & robber game. We start with directed tree-width.

Proposition 4.1.5. *Let G be a semicomplete digraph. If d-pw$(G) \geq 4(k+1)^2+7(k+1)$ then d-tw$(G) \geq k$.*

Proof. Let us assume d-pw$(G) \geq 4(k+1)^2+7(k+1)$. Then, by the results from [FS13], we know that $G=(V,E)$ contains a $k+1$-jungle $J \subseteq V$. If we can show that the existence of J is enough to ensure that k-cops cannot catch the robber in the visible strong component cops and robber game on G, it follows from Proposition 2.2.13 that the directed tree-width of G must be at least k and thus the assertion follows. Hence what is left to do is describe a winning strategy for the robber against k cops on a $k+1$-jungle J. For the first position (C_0,r_0) we have $C_0=\emptyset$ and the robber may select r_0 to be any vertex of J. Now suppose the game has been going on for i rounds and in each round the robber was able to select a vertex of J as her position. Let (C_{i-1},r_{i-1}) be the current state of the game and let $C_i \subseteq V$ be the next position of the cops. In case $r_{i-1} \notin C_i$ there is nothing to do for the robber and she can stay where she is i.e. $r_i := r_{i-1}$. So we may assume $r_{i-1} \in C_i$. In this case we know $|C_i \setminus \{r_{i-1}\}| \leq k-1$ and thus $|C_{i-1} \cap C_i| \leq k-1$. Hence there must exist a vertex $v \in J \setminus C_i$. As $r_{i-1} \neq v$ we know from J being a $k+1$-jungle that there exist $k+1$ pairwise internally disjoint paths from r_{i-1} to v and vice versa. As $|C_i| \leq k$ in $G-(C_{i-1} \cap C_i)$ at least one path from r_{i-1} to u and one from u to r_{i-1} must be left and thus both vertices belong to the same strong component of $G-(C_{i-1} \cap C_i)$. Thus v is reachable from r_{i-1} and we may set $r_i := v$. As the robber was able to flee to another vertex of J the claim now follows by induction. □

From [AKK+15] and Corollary 4.1.4 we previously mentioned an upper bound on directed path-width in terms of Kelly-width, which is d-pw$(G) \leq 72\text{kw}(G)^2+1$. We can improve this bound following the same general idea as given above. Indeed, since in the strategy as described in the proof of Proposition 4.1.5 the robber only changed her position if she was threatened to be caught if she did not, the strategy above is already a strategy for a visible robber in the strong component game. Since the reachability searching game is a relaxation of the strong component game and the (in)visibility of the robber does not play a role in this strategy it is straightforward to see that using the same technique, an invisible and inert robber can also avoid being caught by k cops in the reachability searching game. From these arguments we obtain the following result.

Proposition 4.1.6. *Let G be a semicomplete digraph. If d-pw$(G) \geq 4(k+1)^2+7(k+1)$ then kw$(G) \geq k$.*

4.1.4 Directed (Linear) Clique-width and Directed Path-width on Semicomplete Digraphs

In [FP19], the authors prove that on semicomplete digraphs, directed path-width can be used to give an upper bound for directed clique-width. The main idea of the proof of [FP19, Lemma 2.14] is to define a directed clique-width expression along a nice path-decomposition. Since this proof only uses linear clique-width operations, we can restrict their result to the following result:

Proposition 4.1.7 ([FP19])**.** *For every semicomplete digraph G it holds that*

$$\text{d-cw}(G) \leq \text{d-lcw}(G) \leq \text{d-pw}(G)+2.$$

Notice that, the other direction, i.e., using directed (linear) clique-width as an upper bound of directed path-width, is not possible for semicomplete digraphs in general. That follows directly from the proof of Proposition 4.1.1, as the counterexample, a bidirectional complete digraph, is a semicomplete digraph.

Using the results from this and previous subsections, it is possible to improve the general results for the comparison of directed width parameters on semicomplete digraphs.

4.1.5 Summary and Conclusion

As by [FP19] for a semicomplete digraph G it holds that d-cw(G) is at most d-pw$(G)+2$, we finally conclude that all above mentioned parameters are upper bounds to directed clique-width. This result is even extendable to directed linear clique-width (d-lcw). More precisely we show by using Theorem 4.1.3 and Propositions 4.1.5, 4.1.6 and 4.1.7 that for any choice of functions $f, g \in \{\text{d-pw}, \text{d-tw}, \text{dagw}, \text{kw}, \text{d-lcw}, \text{d-cw}\}$, there must exist a function $h_{f,g}$ such that, if G is a semicomplete digraph with $f(G) \leq k$ then $g(G) \leq h_{f,g}(k)$ where the functions $h_{f,g}$ are presented in Table 4.2.

Theorem 4.1.8. *Let G be a semicomplete digraph and $f, g \in \{$d-pw, d-tw, dagw, kw, d-lcw, d-cw$\}$. If $f(G) \leq k$, then $g(G) \leq h_{f,g}(k)$ where $h_{f,g} \colon \mathbb{N} \to \mathbb{N}$ is given by Table 4.2.*

Combining these results with the above mentioned theorem of Courcelle et al. on bounded clique-width [CMR00] and the FPT-algorithm for approximating directed tree-width within a linear factor by Campos et al. [CLMS19], we have the following result:

Theorem 4.1.9. *Every problem expressible in monadic second-order logic on quantification over vertices and vertex sets (MSO_1) is fixed parameter tractable on semicomplete digraphs with respect to the parameter directed tree-width.*

g \ f	d-pw	d-tw	dagw	kw	d-lcw	d-cw
d-pw	k	$4k^2+15k+10$	$k-1$	$4k^2+7k$	∞	∞
d-tw	k	k	$k-1$	$6k-2$	∞	∞
dagw	$k+1$	$4k^2+15k+11$	k	k^2	∞	∞
kw	$k+1$	$4k^2+15k+11$	k	k	∞	∞
d-lcw	$k+2$	$4k^2+15k+12$	$k+1$	k^2+2	k	∞
d-cw	$k+2$	$4k^2+15k+12$	$k+1$	k^2+2	k	k

Table 4.2: Relations between digraph parameters on **semicomplete digraphs**. The parameter of the left column is bounded by the respective parameter of the top row by the specified function where k is the corresponding width. We use '∞' if the relation is unbounded, that is if $h_{f,g}$ does not exist.

A more detailed explanation about Courcelles theorem is given later in Chapter 7, more precisely in Subsection 7.6.2.

The landscape of directed width measures is a wild one. Started by the introduction of directed tree-width many different generalizations of undirected tree-width have been invented and received different amounts of attention. Some of these parameters were considered very little; possibly because of the results of [GHK$^+$16], which essentially rule out any algorithmic application of these parameters beyond some specialized routing problems. So while the search for 'good' digraph width parameters inspired by tree-width does not seem very promising, one could turn to the logic based parameters instead. Here directed clique-width reigns supreme, but recently other attempts at finding interesting parameters such as a directed version of *maximum induced matching width* [JKT21] have been made.

Summarized we can say that directed path-width, directed tree-width, Kelly-width and DAG-width are equivalent on semicomplete digraphs. In particular this implies that each of these measures acts as an upper bound on directed clique-width and thus the algorithmic power of directed clique-width can now be accessed by any of the other parameters.

Hence as a consequence of these results on semicomplete digraphs every digraph problem, which is describable in MSO_1 logic is fixed parameter tractable for these width measures if a decomposition of bounded width is given.

Is the directed path-width problem NP-hard on semicomplete digraphs?

Our result, that computing DAG-width is in NP on semicomplete digraphs while it is PSPACE-hard in general [AKR16] recalls the question if computing directed path-width and thus, DAG-width is NP-hard on semicomplete digraphs. This problem has been considered in some works, but while there are given several FPT algorithms to solve this problem [FP13] or ,e.g., using degree orderings [FP19], it is still open if the problem is NP-hard at all. There are some problems as for example the k-vertex disjoint directed path problem which is NP-hard on general digraphs even for $k = 2$. However, when restricting to semicomplete digraphs, for the case $k = 2$ Bang-Jensen and Thomassen [BJT92] and for $k \geq 2$ Chudnovsky et al. [CSS15] showed that the problem is polynomial time solvable. So one can get the impression that on semicomplete the directed path-width problem could be polynomial time solvable as well. The intuitive hardness proof would lead to a reduction from undirected path-width. However, in this case the underlying undirected graph of a semicomplete digraph is a clique, which makes path-width trivially solvable. Thus there is the demand for different ideas. Since we now know that DAG-width (-1) is equal to the directed path-width on semicomplete digraphs, one could use this to solve this complexity issue. Further, there are equivalent definitions for directed path-width as a cops and robber game [Bar06] or the directed vertex separation number. Unfortunately, none of the ideas led to a fruitful approach yet and the problem remains open.

4.2 Computing Directed Graph Parameters on (Extended) Directed Co-graphs

In the following we show how to compute several directed graph parameters on (extended) directed co-graphs. The content in this section is taken from [GKR21b].

4.2.1 Directed Path-width on (Extended) Directed Co-graphs

As already mentioned, determining whether the (undirected) path-width of some given (undirected) graph is at most some given value w is NP-complete [KF79] even for bipartite graphs, complements of bipartite graphs [ACP87], chordal graphs [Gus93], bipartite distance-hereditary graphs [KBMK93], and planar graphs with maximum semidegree $\Delta^0(G) \leq 3$, see [MS88]. Lemma 2.2.7 implies that determining whether the directed path-width of some given digraph is at most some given value w is NP-complete even for digraphs whose underlying graphs lie in the mentioned classes. On the other hand, determining whether the (undirected) path-width of some given (undirected) graph is at most some given value w is polynomial for permutation graphs [BKK95], circular arc graphs [ST07], and co-graphs [BM93]. While undirected path-width can be solved by an FPT-algorithm [Bod96], the existence of such an algorithm for directed path-width is still open. The directed path-width of a digraph $G = (V,E)$ can be computed in time $O(|E|\cdot|V|^{2\text{d-pw}(G)}/(\text{d-pw}(G)-1)!)$ by [KKK$^+$16] and in time $O(\text{d-pw}(G) \cdot |E| \cdot |V|^{2\text{d-pw}(G)})$ by [Nag12]. This leads to XP-algorithms for directed path-width w.r.t. the standard parameter and implies that for each constant w, it is decidable in polynomial time whether a given digraph has directed path-width at most w. Beside the standard parameter there are results for other parameters. A digraph $G = (V,E)$ is ℓ-*semicomplete*, if each vertex $v \in V$ has at most ℓ non-neighbors [KKT15]. In [KKT15] it is shown how to decide whether the directed path-width of an ℓ-semicomplete digraph is at most w in time $(\ell + 2w + 1)^{2w} \cdot n^{O(1)}$. The directed path-width can be computed in time $3^{\tau(und(G))} \cdot |V|^{O(1)}$, where $\tau(und(G))$ denotes the vertex cover number of the underlying undirected graph of G, by [Kob15]. A digraph G is a *sequence digraph* using k sequences, if there are k sequences with entries from $V(G)$, such that $(u,v) \in E(G)$ if and only if in one of the sequences there is an occurrence of u appearing before an occurrence of v [GRR18]. For sequence digraphs with a given decomposition into k sequences the directed path-width can be computed in time $O(k \cdot (1+N)^k)$, where N denotes the maximum sequence length [GRR18].

Right now, special graph classes have been considered very few regarding directed width measures. Not even graph classes of small widths are known. For undirected graphs the Robertson/Seymour Theorem [RS04] leads to characterizations for the set of graphs of tree-width at most k and the set of graphs of path-width at most k by a finite set of forbidden minors. For small values of k these sets are known. Finding forbidden graph minors for some digraph class of bounded directed path-width seems to be much more involved. In [KZ15] it has been shown that digraphs of directed path-width at most one are characterized by a finite number of forbidden directed butterfly minors and in [Wie20] digraphs of directed tree-width at most one are characterized by a minimal, although infinite, family of forbidden butterfly minors. For some classes of digraphs G of directed path-width at most one and directed tree-width at most one such that $und(G)$ is tree-like in [GR19b] characterizations by at most three forbidden directed minors are given. But no graph classes of directed width k are known and also special digraph classes of bounded width are hard to find.

For extended directed co-graphs it is possible to compute the directed path-width in linear time.

Theorem 4.2.1 ([GKR21b]). *Let $G=(V_G,E_G)$ and $H=(V_H,E_H)$ be two vertex-disjoint digraphs, then the following properties hold.*

1. $d\text{-}pw(\bullet)=0$
2. $d\text{-}pw(G\oplus H)=\max\{d\text{-}pw(G),d\text{-}pw(H)\}$
3. $d\text{-}pw(G\oslash H)=\max\{d\text{-}pw(G),d\text{-}pw(H)\}$
4. $d\text{-}pw(G\ominus H)=\max\{d\text{-}pw(G),d\text{-}pw(H)\}$
5. $d\text{-}pw(G\otimes H)=\min\{d\text{-}pw(G)+|V_H|,d\text{-}pw(H)+|V_G|\}$

4.2.2 Directed Tree-width on (Extended) Directed Co-graphs

We can conclude some hardness results from the corresponding undirected tree-width problem. Lemma 2.2.12 implies that determining whether the directed tree-width of some given digraph is at most some given value w is NP-complete even for digraphs whose underlying graphs lie in the mentioned classes (bipartite and co-bipartite graphs).

The results of [JRST01b] lead to an XP-algorithm for directed tree-width w.r.t. the standard parameter which implies that for each constant w, it is decidable in polynomial time whether a given digraph has directed tree-width at most w.

In order to show how to compute directed tree-width efficiently on extended directed co-graphs, we introduce some properties of directed tree-decompositions.

Lemma 4.2.2 ([JRST01b]). *Let G be some digraph and H be a subdigraph of G, then $d\text{-}tw(H)\leq d\text{-}tw(G)$.*

Lemma 4.2.3 (Bidirectional complete subdigraph)**.** *Let $(T, \mathcal{X}, \mathcal{W})$, $T = (V_T, E_T)$, where r_T is the root of T, be a directed tree-decomposition of some digraph $G = (V, E)$ and $G' = (V', E')$ with $V' \subseteq V$ be a bidirectional complete subdigraph. Then, $V' \subseteq W_{r_T}$ or there is some $(r, s) \in E_T$, such that $V' \subseteq W_s \cup X_{(r,s)}$.*

Proof. First, we choose a vertex s in V_T, such that $W_s \cap V' \neq \emptyset$ but for every vertex s' such that $s < s'$ it holds that $W_{s'} \cap V' = \emptyset$.
Next, we show that W_s leads to a set which shows the statement of the lemma. If s is the root of T, then $W_{s'} \cap V' \neq \emptyset$ for none of its successors s' in T i.e., $W_{s'} \cap V' = \emptyset$ for all of its successors s' in T, which implies by (dtw-1) that $V' \subseteq W_s$. Otherwise, let r be the predecessor of s in T. If $V' \subseteq W_s$ the statement is true. Otherwise, let $c \in V' \setminus W_s$ and $c' \in V' \cap W_s$. Then, $(c, c') \in E$ and $(c', c) \in E$ implies that $c \in X_{(r,s)}$ by (dtw-2), since otherwise (c', c, c') is a directed walk in $G - X_{(r,s)}$ with first and last vertex $c' \in W_{\geq s}$ that uses a vertex of $G - (X_{(r,s)} \cup W_{\geq s})$, namely c. □

Lemma 4.2.4 (Bidirectional complete bipartite subdigraph)**.** *Let $G = (V, E)$ be some digraph, $(T, \mathcal{X}, \mathcal{W})$, $T = (V_T, E_T)$, where r_T is the root of T, be a directed tree-decomposition of G. Further, let $A, B \subseteq V$, $A \cap B = \emptyset$, and $\{(u, v), (v, u) \mid u \in A, v \in B\} \subseteq E$. Then, $A \cup B \subseteq W_{r_T}$ or there is some $(r, s) \in E_T$, such that $A \subseteq W_s \cup X_{(r,s)}$ or $B \subseteq W_s \cup X_{(r,s)}$.*

Proof. Similar as in the proof of Lemma 4.2.3 we can find a vertex s in V_T, such that $W_s \cap (A \cup B) \neq \emptyset$ but for every vertex s' with $s < s'$ holds $W_{s'} \cap (A \cup B) = \emptyset$.
If s is the root of T, then $W_{s'} \cap (A \cup B) \neq \emptyset$ for none of its successors s' in T, i.e., $W_{s'} \cap (A \cup B) = \emptyset$ for all of its successors s' in T, which implies by (dtw-1) that $A \cup B \subseteq W_s$.
Otherwise, let r be the predecessor of s in T. If $A \cup B \subseteq W_s$ the statement is true. Otherwise, we know that either there is some $a \in A \cap W_s$ and $b \in B \setminus W_s$ or $a \in A \setminus W_s$ and $b \in B \cap W_s$.
We assume that there is some $a \in A \cap W_s$ and $b \in B \setminus W_s$. Then, $(a, b) \in E$ and $(b, a) \in E$ implies that $b \in X_{(r,s)}$ by (dtw-2). Thus, we have shown $B \subseteq W_s \cup X_{(r,s)}$.
If we assume that there some $b \in B$ such that $b \in W_s$, we conclude $A \subseteq W_s \cup X_{(r,s)}$. □

Lemma 4.2.5. *Let G be a digraph of directed tree-width at most k. Then, there is a directed tree-decomposition $(T, \mathcal{X}, \mathcal{W})$, $T = (V_T, E_T)$, of width at most k for G such that $|W_r| \leq 1$ for every $r \in V_T$.*

Proof. Let $G = (V, E)$ be a digraph and $(T, \mathcal{X}, \mathcal{W})$, $T = (V_T, E_T)$, be a directed tree-decomposition of G. For every $r \in V_T$ such that $|W_r| \leq 1$ the statement of the lemma is fulfilled. Let $r \in V_T$ such that $W_r = \{v_1, \ldots, v_m\}$ for some $m > 1$. Further, let p be the predecessor of r in T and $s_1, \ldots, s_\ell$ be the successors of r in T. Let $(T', \mathcal{X}', \mathcal{W}')$ be defined by the following modifications of $(T, \mathcal{X}, \mathcal{W})$: We replace vertex r in T by the directed path $P(r) = (\{r_1, \ldots, r_m\}, \{(r_1, r_2), \ldots, (r_{m-1}, r_m)\})$ and replace arc (p, r) by (p, r_1) and the ℓ arcs (r, s_j), $1 \leq j \leq \ell$, by the ℓ arcs (r_m, s_j), $1 \leq j \leq \ell$ in T'. We define the sets $W'_{r_j} = \{v_j\}$ for $1 \leq j \leq m$. Further, we define the sets $X'_{(p,r_1)} = X_{(p,r)}$, $X'_{(r_m,s_j)} = X_{(r,s_j)}$, $1 \leq j \leq \ell$, and $X'_{(r_j,r_{j+1})} = X_{(p,r)} \cup \{r_1, \ldots, r_j\}$, $1 \leq j \leq m - 1$.
By the definition, $\mathcal{W}'$ leads to a partition of V. The normality holds for the arcs of T' as follows. First, we consider the arcs (r_{i-1}, r_i), $1 < i \leq m$, which we inserted for sets W_r of size $m > 1$.

The set $W'_{\geq r_i}$ is $\mathcal{X}'_{(r_{i-1},r_i)}$-normal since $W_{\geq r}$ is $X_{(p,r)}$-normal and $X'_{(r_{i-1},r_i)} = X_{(p,r)} \cup \{r_1,\dots,r_{i-1}\}$. Further, the property is fulfilled for arc (p,r_1) and (v_m,s_j), $1 \leq j \leq \ell$ since the considered vertex sets of G did not change. Thus, triple $(T',\mathcal{X}',\mathcal{W}')$ is a directed tree-decomposition of G.
The width of $(T',\mathcal{X}',\mathcal{W}')$ is at most the width of $(T,\mathcal{X},\mathcal{W})$ since for every r_j, $1 \leq j \leq m$, then it holds that $|W'_{r_j} \cup \bigcup_{e\sim r_j} X'_e| \leq |W_r \cup \bigcup_{e\sim r} X_e|$.
If we perform this transformation for every $r \in V_T$ such that $|W_r| > 1$, we obtain a directed tree-decomposition of G which fulfills the properties of the lemma. □

Remark 4.2.6. By considering the directed tree-width forbidding empty sets W_r in [JRST01b] the statement of Lemma 4.2.5 can be strengthened to $|W_r| = 1$ for every $r \in V_T$.

Lemma 4.2.7. *Let $G = (V,E)$ be a digraph of directed tree-width at most k, such that there is a 2-partition (V_1,V_2) of V with $V_1 \neq \emptyset$, $V_2 \neq \emptyset$ and $\{(u,v),(v,u) \mid u \in V_1, v \in V_2\} \subseteq E$. Let $(T,\mathcal{X},\mathcal{W})$, $T = (V_T,E_T)$ be a directed tree-decomposition of width k for G with $|W| \leq 1$ for all $W \in \mathcal{W}$. Then, it holds that either*

(i) $\forall t \in V_T$ *with* $W_t \subseteq V_1$: $|W_t \cup \bigcup_{e\sim t} X_e \cup V_2| \leq k$

(ii) $\forall t \in V_T$ *with* $W_t \subseteq V_2$: $|W_t \cup \bigcup_{e\sim t} X_e \cup V_1| \leq k$

To prove this Lemma we first need some claims. Therefore, let $G = (V,E)$ be a digraph as in the statement of the Lemma.

Claim 4.2.8. If $(T,\mathcal{X},\mathcal{W})$ has width $|V|-1$, for all $t \in V_T$ it holds that $|W_t \cup \bigcup_{e\sim t} X_e \cup V_2 \cup V_1| \leq |V|-1 = k$.

By this claim, the Lemma holds for $k = |V|-1$. So in all further claims we assume that the width k of $(T,\mathcal{X},\mathcal{W})$ is smaller than $|V|-1$.

We further assume w.l.o.g. that for all leafs ℓ of T, $W_\ell \neq \emptyset$.

Claim 4.2.9. For $k < |V|-1$ every vertex $s \in V_T$ with $W_{>s} \cap V_1 \neq \emptyset$ and $W_{>s} \cap V_2 \neq \emptyset$ has exactly one successor t such that $W_{\geq t} \cap V_1 \neq \emptyset$ and $W_{\geq t} \cap V_2 \neq \emptyset$. It further holds that $W_{>s} \setminus W_{\geq t} \subseteq V_1$ or $W_{>s} \setminus W_{\geq t} \subseteq V_2$.

Proof. We show this Claim in two steps.

- We first show that s has at most one successor t, such that $W_{\geq t} \cap V_1 \neq \emptyset$ and $W_{\geq t} \cap V_2 \neq \emptyset$. Assume that s has two successors t_1 and t_2 such that $W_{\geq t_1} \cap V_1 \neq \emptyset$ and $W_{\geq t_1} \cap V_2 \neq \emptyset$ and $W_{\geq t_2} \cap V_1 \neq \emptyset$ and $W_{\geq t_2} \cap V_2 \neq \emptyset$. Then, it holds that $V \setminus W_{\geq t_1} \subseteq X_{(s,t_1)}$ and $V \setminus W_{\geq t_2} \subseteq X_{(s,t_2)}$. As $W_{\geq t_1} \cap W_{\geq t_2} = \emptyset$ it follows that $V \setminus W_{\geq t_1} \cup V \setminus W_{\geq t_2} = V$ and thus, $W_s \cup \bigcup_{e\sim s} X_e = V$. Then the resulting width of $(T = (V_T,E_T),\mathcal{X},\mathcal{W})$ is $|V|-1$ which is a contradiction to the assumption that the width $k < |V|-1$.

- We now show that s has at least one successor t, such that $W_{\geq t} \cap V_1 \neq \emptyset$ and $W_{\geq t} \cap V_2 \neq \emptyset$. Assume that for all successors t of s it holds that either $W_{\geq t} \subseteq V_1$ or $W_{\geq t} \subseteq V_2$. As $W_{>s} \cap V_1 \neq \emptyset$ and $W_{>s} \cap V_2 \neq \emptyset$ there exist successors t_1, t_2 of s such that $W_{\geq t_1} \subseteq V_1$ and $W_{\geq t_2} \subseteq V_2$. Then, it holds that $V_2 \subseteq X_{(s,t_1)}$ and $V_1 \subseteq X_{(s,t_2)}$ and thus, that $V \subseteq W_s \cup \bigcup_{e\sim s} X_e$. Then, the resulting width of $(T = (V_T,E_T),\mathcal{X},\mathcal{W})$ is $|V|-1$ which is a contradiction to the assumption that the width $k < |V|-1$.

As we have now proven that there is exactly one successor t such that $W_{\geq t} \cap V_1 \neq \emptyset$ and $W_{\geq t} \cap V_2 \neq \emptyset$, for all other successors t_i of s it holds that $W_{\geq t_i} \subseteq V_1$ or $W_{\geq t_i} \subseteq V_2$. But by the same argumentation as in the second bullet point, if there are successors t_i and t_j of s such that $W_{\geq t_i} \subseteq V_1$ and $\bigcup_{\tilde{t} \geq t_j} W_{\tilde{t}} \subseteq V_2$, it would follow that $k = |V| - 1$. As this is a contradiction we can conclude that $W_{>s} \setminus W_{\geq t} \subseteq V_1$ or $W_{>s} \setminus W_{\geq t} \subseteq V_2$. □

Claim 4.2.10. Assume that $k < |V| - 1$. Let $\mathcal{L} \subseteq V_T$ be the set of all leaves ℓ of T and $L = \bigcup_{\ell \in \mathcal{L}} W_\ell$, such that for the directed path $(u_1, \ldots, u_q)$ starting at root u_1 and ending with leaf $\ell = u_q$ it holds that $\bigcup_{1 \leq i \leq q} W_{u_i} \cap V_1 \neq \emptyset$ and $\bigcup_{1 \leq i \leq q} W_{u_i} \cap V_2 \neq \emptyset$ and $\forall u_i, 1 \leq i \leq q$:

- $W_{\geq u_i} \backslash W_{\geq u_{i+1}} \subseteq V_1$ or
- $W_{\geq u_i} \backslash W_{\geq u_{i+1}} \subseteq V_2$.

Then, $\mathcal{L} \neq \emptyset$ and it holds that either $L \subseteq V_1$ or $L \subseteq V_2$.

Proof. We search set $\mathcal{L}$ by traversing T starting at the root and choosing all possible paths to leafs that fulfill the conditions above.
Let u_1 be the root of T. Obviously, it holds that $W_{\geq u_1} \cap V_1 \neq \emptyset$ and $W_{\geq u_1} \cap V_2 \neq \emptyset$. For all u_i with only one successor we choose this successor u_{i+1}.
By Claim 4.2.9 for every u_i with more than one successor and $W_{\geq u_i} \cap V_1 \neq \emptyset$ and $W_{\geq u_i} \cap V_2 \neq \emptyset$ there is exactly one successor t such that $W_{\geq t} \cap V_1 \neq \emptyset$ and $W_{\geq t} \cap V_2 \neq \emptyset$. In this case we take this t as u_{i+1}.
For u_i with $W_{\geq u_i} \cap V_1 \neq \emptyset$ or $W_{\geq u_i} \cap V_2 \neq \emptyset$ it holds that $W_{\geq u_i} \subseteq V_1$ or $W_{\geq u_i} \subseteq V_2$. Then, we choose all paths from this u_i to any following leaf and add this leaf to $\mathcal{L}$.
Consequently, it holds that $L \subseteq V_1$ or $L \subseteq V_2$ and by construction for every $\ell \in \mathcal{L}$ it holds that for the path $(u_1, \ldots, u_q)$ from the root to ℓ it holds that $\forall u_i, 1 \leq i \leq q$:

- $W_{\geq u_i} \backslash W_{\geq u_{i+1}} \subseteq V_1$ or
- $W_{\geq u_i} \backslash W_{\geq u_{i+1}} \subseteq V_2$,

as we always choose the path containing vertices of V_1 and V_2, until only vertices from one of the sets are left. By Claim 4.2.9 this path is unique. Further, $\mathcal{L} \neq \emptyset$ holds since by construction at least one path, as described above, must exist. Additionally, as we can assume that there are no leafs with empty bags in T, it follows that $L \neq \emptyset$. □

In Claim 4.2.9 and Claim 4.2.10 we restricted the structure of the decomposition tree. But this does not suffice to prove the Lemma, we need further restrictions. As we cannot simply exclude this structures as we could in Claims 4.2.9 and 4.2.10, we give a way to transform the decomposition, such that the new decomposition tree fulfills more structural characteristics.

Claim 4.2.11. Let $k < |V| - 1$. We can assume that for $(T, \mathcal{X}, \mathcal{W})$ it holds that if $L \subseteq V_1$ ($L \subseteq V_2$ respectively), then all $W_s \subseteq V_1$ ($W_s \subseteq V_2$ respectively) with $W_{>s} \cap V_1 \neq \emptyset$ and $W_{>s} \cap V_2 \neq \emptyset$ have exactly one successor t such that $W_{\geq t} \cap V_1 \neq \emptyset$ and $W_{\geq t} \cap V_2 \neq \emptyset$ and it further holds that $W_{>s} \setminus W_{\geq t} \subseteq V_1$ (V_2 respectively).

Proof. We show that, if $(T,\mathcal{X},\mathcal{W})$ does not fulfill this claim, we can transform it to a decomposition $(T',\mathcal{X}',\mathcal{W}')$ of width k' such that $k' \leq k$ and all former claims remain true.
For every vertex $v \in V_T$ we define $w(v) = W_v \cup \bigcup_{e \sim_T v} X_e$ (the set that determines the width of this tree-decomposition) and for $v \in V_{T'}$ we define $w'(v)$ respectively. Without loss of generality we assume that $L \subseteq V_1$. (As the proof for $L \subseteq V_2$ works analogously. By Claim 4.2.9 we know that s has exactly one successor t such that $W_{\geq t} \cap V_1 \neq \emptyset$ and $W_{\geq t} \cap V_2 \neq \emptyset$. We further know that $W_{>s} \setminus W_{\geq t} \subseteq V_1$ or $W_{>s} \setminus W_{\geq t} \subseteq V_2$.
The only thing which now remains open to show this Claim is that for $(T',\mathcal{X}',\mathcal{W}')$ it holds that if $W'_s \subseteq V_1$ it holds that $\bigcup_{\tilde{s}>s} W'_{\tilde{s}} \setminus \bigcup_{\tilde{t}\geq t} W'_{\tilde{t}} \subseteq V_1$.
In order to prove this, we assume the contrary and then transform the decomposition such that the Claim holds. So therefore we assume that in T it does not hold that $\bigcup_{\tilde{s}>s} W'_{\tilde{s}} \setminus \bigcup_{\tilde{t}\geq t} W'_{\tilde{t}} \subseteq V_1$, which by Claim 4.2.9 means, that $W_{>s} \setminus W_{\geq t} \subseteq V_2$. Let p be the predecessor of s in T and $t, t_1, \ldots, t_r$ the successors of s in T. Then, we construct $T' = (V'_T, E'_T)$ with $V'_T = V_T$ and $E'_T = \{(u,v) \mid (u,v) \in E_T, u,v \neq s\} \cup \{(t_1,s),(s,t),(p,t_1)\} \cup \bigcup_{2\leq i\leq r}\{(t_1,t_i)\}$. In words this means means that t_1 is now a successor of p in T' and the predecessor of s and $t_2,\ldots,t_r$ in T'. Further, it holds that $W'_v = W_v$ for all $v \in V'_T$ and that $X'_e = X_e$ for all $e \in E'_T \cap E_T \setminus \{(t_1,v) \mid (t_1,v) \in E_T\}$. Let $X'_{(p,t_1)} = X_{(p,s)}$, $X'_{(t_1,s)} = X_{(s,t)} \setminus W_s$, $X'_{(t_1,t_i)} = X_{(s,t_i)}$ and for all successors v of t_1 in T let $X'_{(t_1,v)} = X_{(t_1,v)}$.
We briefly show that all conditions of an directed tree decomposition remain fulfilled. As the W-sets does not change, it is obvious that they form a partition of V. Remains to show, that for all edges $(u,v) \in V'_T$ the set $W'_{\geq v}$ remains $X'_{(u,v)}$-normal.

- For all arcs $(u,v) \in V'_T$ with $(u,v) \in V_T$ and $W'_{\geq v} = W_{\geq v}$ and $X'_{(u,v)} = X_{(u,v)}$ it holds that $W'_{\geq v}$ is $X'_{(u,v)}$-normal, as $W_{\geq v}$ is $X_{(u,v)}$-normal

- (p,t_1): $W'_{\geq t_1} = W_{\geq s}$ is $X_{(p,s)} = X_{(p,t_1)}$-normal

- (t_1,s): $W'_{\geq s} = W_{\geq t} \cup W_s$. It holds that $W_{\geq t}$ is $X_{(s,t)}$-normal and thus $W_{\geq t} \cup W_s$ is $X_{(s,t)} \cup W_s$-normal. It follows that $W'_{\geq s} = W_{\geq t} \cup W_s$ is $X'_{(t_1,s)} = X_{(s,t)} \cup W_s$-normal.

We show that for every vertex v in V'_T it holds that $|w'(v')| - 1 \leq k$ by showing that for every vertex $v' \in V'_T$ there exists a vertex $u \in V_T$ such that $|w'(v)| \leq |w(u)|$. As for all other vertices it holds that $w'(v) = w(v)$, we only need to look at the widths induced by p, s and t.

- Consider $w'(p)$. As $W'_{>p} = W_{>p}$, it is possible to set $X'_{(p,t_1)} = X_{(p,s)}$. Then, it holds that $w'(p) = w(p)$ and further that $w'(p) - 1 \leq k$.

- Consider $w'(t_1) = W'_{t_1} \cup X'_{(p,t_1)} \cup X'_{(t_1,s)} \cup \bigcup_{v \in N^+_T(t_1)} X'_{(t_1,v)} \cup \bigcup_{2\leq i\leq r} X'_{(t_1,t_i)}$. As $W_{\geq t} \cap V_1 \neq \emptyset$ and $W_{\geq t} \cap V_2 \neq \emptyset$, it holds that $X_{(s,t)} = V \setminus W_{\geq t}$, so $W_{t_1} \subseteq X_{(s,t)}$. It further holds that $X_{(t_1,\tilde{t_1})} \setminus W_{t_1} \subseteq X_{s,t_1}$, such that $X_{(t_1,\tilde{t_1})} \subseteq X_{s,t_1} \cup W_{t_1}$ for all successors $\tilde{t_1}$ of t_1 in T. Thus, it follows that

$$\begin{aligned}
w'(t_1) &= W'_{t_1} \cup X'_{(p,t_1)} \cup X'_{(t_1,s)} \cup \bigcup_{v \in N_T^+(t_1)} X'_{(t_1,v)} \cup \bigcup_{2 \le i \le r} X'_{(t_1,t_i)} \\
&= W_{t_1} \cup X_{(p,s)} \cup \left(X_{(s,t)} \setminus W_s\right) \cup \bigcup_{v \in N_T^+(t_1)} X_{(t_1,v)} \cup \bigcup_{2 \le i \le r} X_{(s,t_i)} \\
&\subseteq X_{(s,t)} \cup X_{(p,s)} \cup X_{(s,t)} \cup X_{(s,t_1)} \cup W_{t_1} \cup \bigcup_{2 \le i \le r} X_{(s,t_i)} \\
&\subseteq X_{(s,t)} \cup X_{(p,s)} \cup X_{(s,t)} \cup X_{(s,t_1)} \cup X_{(s,t)} \cup \bigcup_{2 \le i \le r} X_{(s,t_i)} \\
&\subseteq W_s \cup \bigcup_{e \sim_T s} X_e \\
&= w(s)
\end{aligned}$$

- Consider $w'(s) = W'_s \cup X'_{(t_1,s)} \cup X'_{(s,t)}$. As $W_{\ge t} \cap V_1 \ne \emptyset$ and $W_{\ge t} \cap V_2 \ne \emptyset$, it holds that $X'_{(t_1,s)} = V \setminus W_{\ge s} \subset V \setminus W_{\ge t} = X'_{(s,t)}$. As further $X'_{(s,t)} = X_{(s,t)}$ it follows that

$$w'(s) = W'_s \cup X'_{(t_1,s)} \cup X'_{(s,t)} \subseteq X'_{(s,t)} = X_{(s,t)} \subseteq w(s).$$

Thus, the widths induced by the vertices v with $w'(v) \ne w(v)$, are not increasing the width of the directed tree-decomposition. □

Proof. of Lemma 4.2.7. By Claim 4.2.8 the Lemma holds for $k = |V| - 1$. Assume that $k < |V| - 1$. Let L be the set of all leaves ℓ in T such that for the path $(u_1, \dots u_q)$ from the root to ℓ it holds that $\bigcup_{1 \le i \le q} W_{u_i} \cap V_1 \ne \emptyset$, $\bigcup_{1 \le i \le q} W_{u_i} \cap V_2 \ne \emptyset$ and $\forall u_i$ with $1 \le i \le q$:

- $W_{\ge u_i} \setminus W_{\ge u_{i+1}} \subseteq V_1$ or
- $W_{\ge u_i} \setminus W_{\ge u_{i+1}} \subseteq V_2$.

By Claim 4.2.10 it holds that $L \ne \emptyset$ and either $L \subseteq V_1$ or $L \subseteq V_2$.
W.l.o.g. assume that $L \subseteq V_1$, for $L \subseteq V_2$ the proof works analogously.
We show that $\forall W_t \in \mathcal{W}$ with $W_t \subseteq V_1$ it holds that $W_t \cup \bigcup_{e \sim t} X_e \cup V_2 \le k$. By the construction in Claim 4.2.10 and the structural information of Claim 4.2.11, there is a vertex u_r such that $L \subseteq W_{>u_r}$, $W_{>u_r} \subseteq V_1$ and $W_{u_r} \subseteq V_2$. Let $(u_1, \dots u_r)$ be the path from the root to this vertex. As $W_{>u_r} \subseteq V_1$, it holds that for all successors u of u_r that $V_2 \subseteq X_{(u_r,u)}$ and as $W_{u_r} \subseteq V_2$, it holds that $V \setminus W_{\ge u_r} \subseteq X_{(u_{r-1},u_r)}$. It then holds that $V_2 \cup (V \setminus W_{\ge u_r}) \subseteq W_{u_r} \cup \bigcup_{e \sim u_r} X_e$ and thus $|V_2 \cup (V \setminus W_{\ge u_{r-1}})| \le k$. Further, for the path $(u_1, \dots, u_{r-1})$ it holds that for every $1 \le i \le r-1$, $X_{(u_i,u_{i+1})} = V \setminus W_{\ge u_{i+1}} \subseteq V \setminus W_{\ge u_{r-1}} = X_{(u_{r-1},u_r)}$. Let now W_t be any element of $\mathcal{W}$ such that $W_t \subseteq V_1$. By Claim 4.2.10 and Claim 4.2.11 it holds that either

(i) There is a successor t' of t such that $W_{\ge t'} \subseteq V_1$ or

(ii) $t = u_i$ with $1 \le i \le r-2$.

In case (i) it holds that $V_2 \subseteq X_{(t,t')}$ and thus, $W_t \cup \bigcup_{e \sim t} X_e \cup V_2 = W_t \cup \bigcup_{e \sim t} X_e \leq k$. In case (ii) it holds that $X_{(u_{i-1},u_i)} = V \setminus W_{\geq u_i}$ and $X_{(u_i,u_{i+1})} = V \setminus W_{\geq u_{i+1}}$. By Claim 4.2.11 it holds that $W_{>u_i} \setminus W_{\geq u_{i+1}} \subseteq V_1$. Thus, we can assume that u_i has no other successors but u_{i+1}, as otherwise we are in case (i). If u_{i+1} is the only successor of u_i, then $W_{u_i} \cup \bigcup_{e \sim u_i} X_e = V \setminus W_{\geq u_{i+1}} \subseteq V \setminus W_{\geq u_{r-1}}$ and then $V_2 \cup W_{u_i} \cup \bigcup_{e \sim u_i} X_e \subseteq V \setminus W_{\geq u_{r-1}} \cup V_2 \subseteq W_{u_{r-1}} \cup \bigcup_{e \sim u_{r-1}} X_e$ and further $|V_2 \cup W_{u_i} \cup \bigcup_{e \sim u_i} X_e| \leq k$ holds. □

On this way we come to the main theorem about the computation of directed tree-width on extended directed co-graphs.

Theorem 4.2.12. *Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be two vertex-disjoint digraphs then the following properties hold.*

1. *d-tw$(\bullet) = 0$*
2. *d-tw$(G \oplus H) = \max\{$d-tw(G), d-tw$(H)\}$*
3. *d-tw$(G \oslash H) = \max\{$d-tw(G), d-tw$(H)\}$*
4. *d-tw$(G \ominus H) = \max\{$d-tw(G), d-tw$(H)\}$*
5. *d-tw$(G \otimes H) = \min\{$d-tw$(G) + |V_H|$, d-tw$(H) + |V_G|\}$*

Proof. Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be two vertex-disjoint digraphs. Further, let $(T_G, \mathcal{X}_G, \mathcal{W}_G)$ be a directed tree-decomposition of G such that r_G is the root of $T_G = (V_{T_G}, E_{T_G})$ and $(T_H, \mathcal{X}_H, \mathcal{W}_H)$ be a directed tree-decomposition of H, such that r_H is the root of $T_H = (V_{T_H}, E_{T_H})$.

1. d-tw$(\bullet) = 0$ holds by a simple directed tree-decomposition.
2. We define a directed tree-decomposition $(T_J, \mathcal{X}_J, \mathcal{W}_J)$ for $J = G \oplus H$. Let ℓ_G be a leaf of T_G. Let T_J be the disjoint union of T_G and T_H with an additional arc (ℓ_G, r_H). Further, let $\mathcal{X}_J = \mathcal{X}_G \cup \mathcal{X}_H \cup \{X_{(\ell_G,r_H)}\}$, where $X_{(\ell_G,r_H)} = \emptyset$ and $\mathcal{W}_J = \mathcal{W}_G \cup \mathcal{W}_H$. The triple $(T_J, \mathcal{X}_J, \mathcal{W}_J)$ satisfies (dtw-1) since the combined decompositions satisfy (dtw-1). Further, $(T_J, \mathcal{X}_J, \mathcal{W}_J)$ satisfies (dtw-2) since additionally in J there is no arc from a vertex of H to a vertex of G. This shows that d-tw$(G \oplus H) \leq \max\{$d-tw(G), d-tw$(H)\}$. Since G and H are induced subdigraphs of $G \oplus H$, by Lemma 4.2.2 the directed tree-width of both leads to a lower bound on the directed tree-width for the combined digraph.
3. The same arguments lead to d-tw$(G \oslash H) = \max\{$d-tw(G), d-tw$(H)\}$.
4. The same arguments lead to d-tw$(G \ominus H) = \max\{$d-tw(G), d-tw$(H)\}$.
5. In order to show d-tw$(G \otimes H) \leq$ d-tw$(G) + |V_H|$ let T_J be the disjoint union of a new root r_J and T_G with an additional arc (r_J, r_G). Further, let $\mathcal{X}_J = \mathcal{X}'_G \cup \{X_{(r_J,r_G)}\}$, where $\mathcal{X}'_G = \{X_e \cup V_H \mid e \in E_{T_G}\}$ and $X_{(r_J,r_G)} = V_H$ and $\mathcal{W}_J = \mathcal{W}_G \cup \{W_{r_H}\}$, where $W_{r_J} = V_H$. Then, $(T_J, \mathcal{X}_J, \mathcal{W}_J)$ is a directed tree-decomposition of width at most d-tw$(G) + |V_H|$

for $G \otimes H$.
In the same way a disjoint union of a new root r_J and T_H with an additional arc (r_J, r_H), $\mathcal{X}'_H = \{X_e \cup V_G \mid e \in E_{T_H}\}$, $X_{(r_J, r_H)} = V_G$, $W_{r_J} = V_G$ lead to a directed tree-decomposition of width at most d-tw$(H) + |V_G|$ for $G \otimes H$. Thus, d-tw$(G \otimes H) \leq \min\{$d-tw$(G) + |V_H|$, d-tw$(H) + |V_G|\}$.
For the reverse direction let $(T_J, \mathcal{X}_J, \mathcal{W}_J)$, $T_J = (V_T, E_T)$, be a directed tree-decomposition of minimum width for $G \otimes H$. By Lemma 4.2.5 we can assume that $|W_t| \leq 1$ for every $t \in V_T$. Then, by Lemma 4.2.7 we can assume that $\forall t \in V_T$ with $W_t \subseteq V_G$ it holds that $|W_t \cup \bigcup_{e \sim t} X_e \cup V_H| \leq$ d-tw$(G \otimes H)$ or $\forall t \in V_T$ with $W_t \subseteq V_H$ it holds that $|W_t \cup \bigcup_{e \sim t} X_e \cup V_G| \leq$ d-tw$(G \otimes H)$.
We assume that $\forall t \in V_T$ with $W_t \subseteq V_H$: $|W_t \cup \bigcup_{e \sim t} X_e \cup V_G| \leq$ d-tw$(G \otimes H)$.
We define $(T'_J, \mathcal{X}'_J, \mathcal{W}'_J)$ as follows. We initialize $T'_J = (V'_T, E'_T)$ with $V'_T = V_T$, $E'_T = E_T$, $X'_e = X_e \cap V_H$, and $W'_s = W_s \cap V_H$. Whenever this leads to an empty set W'_s such that $\bigcup_{\tilde{s} \geq s} W'_{\tilde{s}} = \emptyset$, delete vertex s. Then, $(T'_J, \mathcal{X}'_J, \mathcal{W}'_J)$ is a directed tree-decomposition of H. The width of $(T'_J, \mathcal{X}'_J, \mathcal{W}'_J)$ is at most d-tw$(G \otimes H) - |V_G|$ as shown in the following.

- Suppose s is a vertex in T'_J such that $W'_s = W_s$. Then, it holds that $W_s \subseteq V_H$ and thus $|W_s \cup \bigcup_{e \sim s} X_e \cup V_G| \leq$ d-tw$(G \otimes H)$. So it holds that $|W'_s \cup \bigcup_{e \sim s} X'_e| \leq$ d-tw$(G \otimes H) - |V_G|$.
- Suppose s is a vertex in T'_J such that $W'_s \neq W_s$. Then $W_s \subseteq V_G$ and $W_s \neq \emptyset$. By construction of T'_J, we know that $\bigcup_{\tilde{s} \geq s} W_{\tilde{s}} \cap V_H \neq \emptyset$. Now, we distinguish two cases:
  - Suppose there is a successor t of s such that $\bigcup_{\tilde{t} \geq t} W_{\tilde{t}} \subseteq V_H$. Then $V_G \subseteq X_{(s,t)}$ and thus $|W'_s \cup \bigcup_{e \sim s} X'_e| \leq$ d-tw$(G \otimes H) - |V_G|$.
  - Suppose that for all successors t of s it holds that $\bigcup_{\tilde{t} \geq t} W_{\tilde{t}} \not\subseteq V_H$. Then, as $\bigcup_{\tilde{t} \geq t} W_{\tilde{t}} \cap V_H \neq \emptyset$ and $\bigcup_{\tilde{t} \geq t} W_{\tilde{t}} \cap V_G \neq \emptyset$ it follows by the same argumentation as in the proof of Lemma 4.2.7 that $|W_s \cup \bigcup_{e \sim s} X_e \cup V_G| \leq$ d-tw$(G \otimes H)$. So it holds that $|W'_s \cup \bigcup_{e \sim s} X'_e| \leq$ d-tw$(G \otimes H) - |V_G|$.

Thus, the width of $(T'_J, \mathcal{X}'_J, \mathcal{W}'_J)$ is at most d-tw$(G \otimes H) - |V_G|$ and since $(T'_J, \mathcal{X}'_J, \mathcal{W}'_J)$ is a directed tree-decomposition of H, it follows that d-tw$(H) \leq$ d-tw$(G \otimes H) - |V_G|$.
If we assume that $\forall t \in V_T$ with $W_t \subseteq V_G$: $|W_t \cup \bigcup_{e \sim t} X_e \cup V_H| \leq$ d-tw$(G \otimes H)$ or $\forall t \in V_T$, it follows that d-tw$(G) \leq$ d-tw$(G \otimes H) - |V_H|$.

This shows the statements of the theorem. □

The proof of Theorem 4.2.12 is constructive as we give a tree-decomposition $(T, \mathcal{X}, \mathcal{W})$ of minimum width for every directed co-graph. Since for any operation we define a decomposition where T is a path, we conclude that for any directed co-graph there is a tree-decomposition $(T, \mathcal{X}, \mathcal{W})$ of minimum width such that T is a path. For general digraphs the directed tree-width is at most the directed path-width.

Lemma 4.2.13 ([GKR21b]). *Let G be some digraph, then d-tw$(G) \leq$ d-pw(G).*

Next, we give some examples where the equality does not hold in Lemma 4.2.13.

Example 4.2.14. Every complete biorientation of a rooted tree has directed tree-width 1 and a directed path-width depending on its height. The path-width of perfect 2-ary trees of height h is $\lceil h/2 \rceil$ (cf. [Sch89]) and for $k \geq 3$ the path-width of perfect k-ary trees of height h is exactly h by [EST94, Corollary 3.1].

Remark 4.2.15. The results of Theorem 4.2.1 and Theorem 4.2.12 imply that for every directed co-graph its directed path-width equals its directed tree-width using the definition allowing empty sets W_r of [JRST01a]. Since Lemma 4.2.13 holds for both variants of directed tree-width allowing and forbidding empty sets W_r and directed tree-width allowing empty sets W_r is smaller or equal to directed tree-width forbidding empty sets W_r, the statements of Theorem 4.2.12 also hold when considering the directed tree-width forbidding empty sets W_r in [JRST01b].

4.2.3 Further Directed Width Measures on Extended Directed Co-graphs

In the following we give an overview of how to compute some directed width-measures for the operations of (extended) directed co-graphs. The proofs can be read in [GKR19b].

$X =$	$\bullet$	$G \oplus H$	$G \oslash H$	$G \ominus H$	$G \otimes H$
$\text{fvs}(X)$	0	$\text{fvs}(G) + \text{fvs}(H)$	$\text{fvs}(G) + \text{fvs}(H)$	$\text{fvs}(G) + \text{fvs}(H)$	$\min\{\text{fvs}(G) + \lvert V_H \rvert, \text{fvs}(H) + \lvert V_G \rvert\}$
$\text{fas}(X)$	0	$\text{fas}(G) + \text{fas}(H)$	$\text{fas}(G) + \text{fas}(H)$	$\text{fas}(G) + \text{fas}(H)$	$\text{fas}(G) + \text{fas}(H) + \lvert V_G \rvert \cdot \lvert V_H \rvert$
$\text{cr}(X)$	0	$\max\{\text{cr}(G), \text{cr}(H)\}$	$\max\{\text{cr}(G), \text{cr}(H)\}$	$\max\{\text{cr}(G), \text{cr}(H)\}$	$\min\{\text{cr}(G) + \lvert V_H \rvert, \text{cr}(H) + \lvert V_G \rvert\}$
$\text{dagw}(X)$	1	$\max\{\text{dagw}(G), \text{dagw}(H)\}$	$\max\{\text{dagw}(G), \text{dagw}(H)\}$	$\max\{\text{dagw}(G), \text{dagw}(H)\}$	$\min\{\text{dagw}(G) + \lvert V_H \rvert, \text{dagw}(H) + \lvert V_G \rvert\}$
$\text{kw}(X)$	1	$\max\{\text{kw}(G), \text{kw}(H)\}$	$\max\{\text{kw}(G), \text{kw}(H)\}$	$\max\{\text{kw}(G), \text{kw}(H)\}$	$\leq \min\{\text{kw}(G) + \lvert V_H \rvert, \text{kw}(H) + \lvert V_G \rvert\}$ $\geq \max\{\text{kw}(G), \text{kw}(H)\}$
$\text{ddp}(X)$	1	$\max\{\text{ddp}(G), \text{ddp}(H)\}$	$\text{ddp}(G) + \text{ddp}(H)$	$\leq \text{ddp}(G) + \text{ddp}(H)$ $\geq \max\{\text{ddp}(G), \text{ddp}(H)\}$	$\min\{\text{ddp}(G) + \lvert V_H \rvert, \text{ddp}(H) + \lvert V_G \rvert\}$

Table 4.3: Overview: results about the computation of directed width-measures on directed co-graph operations from [GKR19b] for two vertex-disjoint digraphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$. At two points an exact computation is not possible, so only a range is given.

4.2.4 Overview of Directed Width Measures on Extended Directed Co-graphs

In Table 4.4 we summarize the known relations between the previous mentioned width measures.

$$\begin{tikzcd}
 & & & & \text{ddp} \arrow[d] & \text{fas} \arrow[d] \\
\text{d-tw} & \text{dagw} \arrow[l, "{[BDH^+12]}"'] & \text{kw} \arrow[l, "{[AKK^+15]}"'] & \text{d-pw} \arrow[l, "{[GHK^+14]}"'] & \text{cr} \arrow[l, "{[Gru08]}"'] & \text{fvs} \arrow[l, "{[GHK^+14]}"']
\end{tikzcd}$$

Table 4.4: Known relations between digraph width measures. A directed edge from measure β to measure α indicates that within the cited work there has been shown some function f such that for every digraph it holds $\alpha(G) \leq f(\beta(G))$.

The previous results lead us to close relations between the considered parameters on extended directed co-graphs.

Theorem 4.2.16. *For every extended directed co-graph G, it holds that*

$$kw(G)-1 \leq d\text{-}pw(G) = d\text{-}tw(G) = cr(G) = dagw(G)-1 \leq fvs(G) \leq fas(G) \qquad (4.3)$$

and

$$dagw(G) \leq ddp(G). \qquad (4.4)$$

For the inequations given in (4.3) equality is not possible by the following examples.

- Let K'_n be the $2n$ vertex digraph which is obtained by a complete digraph K_n on n vertices and adding a pendant vertex for every of the n vertices of K_n, then for the complete biorientation $\overleftrightarrow{K'_n}$ it holds that $\mathrm{kw}(\overleftrightarrow{K'_n} \otimes \overleftrightarrow{K'_n}) = 2n-1 < 3n-1 = \mathrm{d\text{-}pw}(\overleftrightarrow{K'_n} \otimes \overleftrightarrow{K'_n})$.
- For transitive tournaments $\overrightarrow{T_n}$, $n \geq 2$, it holds that $\mathrm{dagw}(\overrightarrow{T_n}) = 1 < n = \mathrm{ddp}(\overrightarrow{T_n})$.
- For the disjoint union of two $\overleftrightarrow{K_n}$, $n \geq 3$, it holds that $\mathrm{dagw}(2\overleftrightarrow{K_n}) = n < 2n-2 = \mathrm{fvs}(2\overleftrightarrow{K_n})$.
- For a $\overleftrightarrow{K_n}$, $n \geq 3$, it holds that $\mathrm{fvs}(\overleftrightarrow{K_n}) = n-1 < \frac{n(n-1)}{2} = \mathrm{fas}(\overleftrightarrow{K_n})$.

Furthermore, the two inequations (4.3) and (4.4) cannot be combined by the following examples.

- For transitive tournaments $\overrightarrow{T_n}$, $n \geq 1$, it holds that $\mathrm{fas}(\overrightarrow{T_n}) = 0 < n = \mathrm{ddp}(\overrightarrow{T_n})$.
- For the disjoint union of $\ell \geq 3$ many $\overleftrightarrow{K_n}$, $n \geq 3$, it holds that $\mathrm{ddp}(\ell\overleftrightarrow{K_n}) = n < \ell \cdot n - \ell = \mathrm{fvs}(\ell\overleftrightarrow{K_n})$.

Theorem 4.2.17 ([GKR21b])**.** *For every extended directed co-graph G which is given by a binary ex-di-co-tree, the directed path-width, directed tree-width, directed feedback vertex set number, directed feedback arc set number, cycle rank, and DAG-width can be computed in time $O(|V(G)|)$.*

Since di-co-trees can be computed in linear time and following the results shown in Table 4.3 for the computation of the DAG-depth for directed co-graphs operations, we obtain the following result.

Theorem 4.2.18. *For every directed co-graph G the directed path-width, directed tree-width, directed feedback vertex set number, directed feedback arc set number, cycle rank, DAG-width, and DAG-depth can be computed in time $O(|V(G)|+|E(G)|)$.*

By Lemma 2.2.7 and Lemma 2.2.12 the equality of directed path-width and directed tree-width for directed co-graphs generalizes the known results from [BM93] but cannot be obtained by the known results.

For general digraphs, d-pw(G) leads to a lower bound for pw$(und(G))$ by Lemma 2.2.6 and d-tw(G) leads to a lower bound for tw$(und(G))$ by Lemma 2.2.11. For directed co-graphs we obtain a closer relation as follows.

Corollary 4.2.19. *Let G be a directed co-graph and $\overleftrightarrow{\omega}(G)$ be the order of a largest bidirectional complete subdigraph of G. Then, it holds that*

$$\overleftrightarrow{\omega}(G) = d\text{-}pw(G) - 1 = d\text{-}tw(G) - 1 \leq pw(und(G)) - 1 = tw(und(G)) - 1 = \omega(und(G)).$$

All values are equal if and only if G is a complete bioriented graph.

Proof. The equality $\mathrm{pw}(und(G)) - 1 = \mathrm{tw}(und(G)) - 1 = \omega(und(G))$ has been shown in [BM93]. The equality $\overleftrightarrow{\omega}(G) = \text{d-pw}(G) - 1 = \text{d-tw}(G) - 1$ follows by Lemma 4.2.3 (or Lemma 4.2.3) and Theorem 4.2.16. The upper bound follows by Lemma 2.2.6 or Lemma 2.2.11. □

The relations between digraph width measures given in Table 4.4 can be improved when restricting to directed co-graphs as shown in Table 4.5.

$$\begin{tikzcd}
 & & & & \text{ddp} \arrow[d] & & \\
\text{d-tw} \arrow[r, leftrightarrow] & \text{dagw} \arrow[r, leftrightarrow] & \text{kw} \arrow[r, leftrightarrow] & \text{d-pw} \arrow[r, leftrightarrow] & \text{cr} & \text{fvs} \arrow[l] & \text{fas} \arrow[l]
\end{tikzcd}$$

Table 4.5: Known relations between digraph width measures restricted to directed co-graphs. A directed edge from measure β to measure α indicates that by the results summarized in Table 4.4 and Theorem 4.2.16 there has been shown some function f such that for every directed co-graph it holds $\alpha(G) \leq f(\beta(G))$.

To sum up, there are linear time algorithms for the directed path-width, directed tree-width, directed feedback vertex set number, directed feedback arc set number, cycle rank and DAG-width of extended directed co-graphs and a linear-time algorithm for the DAG-depth of directed co-graphs. We can compare all considered parameters for extended directed co-graphs and obtain equality for directed path-width, directed tree-width, cycle rank and DAG-width. This shows bounds for the class of directed co-graphs for the directed vertex set number, DAG-depth and Kelly-width. The results on directed path-width and directed tree-width generalize the equivalence of path-width and tree-width of co-graphs which is known from [BM93] to digraphs. The shown equality also holds for more general directed tree-width definitions such as allowing empty sets W_r in [JRST01a]. That is not possible for the directed tree-width approach suggested by Reed in [Ree99], which uses sets W_r of size one only for the leaves of T of a directed tree-decomposition $(T, \mathcal{X}, \mathcal{W})$. To obtain a counter-example let $S_{1,n} = (V, E)$ be a star graph on $1+n$ vertices, i.e., $V = \{v_0, v_1, \ldots, v_n\}$ and $E = \{\{v_0, v_i\} \mid 1 \leq i \leq n\}$. Further, let G_n be the complete biorientation of $S_{1,n}$, which is a directed co-graph. Then, $\mathrm{tw}(S_{1,n}) = 1$ and by Theorem 4.2.16 and Theorem 2.2.11 we know $\text{d-pw}(G_n) = \text{d-tw}(G_n) \leq 1$. Using the approach of [Ree99] in any possible tree-decomposition $(T, \mathcal{X}, \mathcal{W})$ for G_n there is a leaf u of T such that $W_u = \{v_0\}$. Further, there is some $u' \in V_T$, such that $(u', u) \in E_T$. By the normality for edge (u', u) it holds that $X_{(u',u)} = \{v_1, \ldots, v_n\}$ which implies that using the approach of [Ree99] the directed tree-width of G is at least n.

The results on the width measures for the directed union of digraphs can be used to show that most of the considered width measures can be obtained by the width of of its strong components. In order to process the strong components of a digraph G, we use its *acyclic condensation*, which is the digraph whose vertices are the strongly connected components $V_1,\dots,V_c$ of G and there is an edge from V_i to V_j if there is an edge (v_i,v_j) in G such that $v_i \in V_i$ and $v_j \in V_j$. Using the acyclic condensation, every digraph G can be represented by the directed union of its strong components [GR19b]. Furthermore, we obtain that for directed co-graphs Kelly-width can be bounded by DAG-width (Theorem 4.2.16). Due to [HK08, Conjecture 30], [AKK$^+$15], and [BJG18, Section 9.2.5] this remains open for general digraphs and is related to one of the biggest open problems in graph searching, namely whether the monotonicity costs for Kelly- and DAG-width games are bounded. It remains open whether there is a linear or polynomial time algorithm to compute Kelly-width on directed co-graphs.

4.3 Directed Graph Parameters on Twin-dh Digraphs

The content of this section is taken from [KR21].

4.3.1 Directed Graph Parameters with a Tree-like Decomposition

In the previous section we presented algorithms to compute different directed width measures on (extended) directed co-graphs in linear time. Among these are directed path-width, directed tree-width, DAG-width and cycle rank. Those algorithms are not extendable directly to twin-dh digraphs, but by Lemma 3.5.13, the results can be expanded to the latter. We have stated the following Lemma in [GR19b] for directed path-width and directed tree-width. The proof is extendable straight-forward to DAG-width and cycle rank.

Lemma 4.3.1. *The directed path-width (directed tree-width, DAG-width and cycle rank respectively) of a digraph G is the maximum of the directed path-widths (directed tree-widths, DAG-widths and cycle ranks respectively) of all strong components of G.*

By this lemma we can show that it is possible to bound the computation of the mentioned parameters on a twin-distance-hereditary digraph.

Theorem 4.3.2. *Let G be a twin-distance-hereditary digraph. Then, it holds that directed path-width, directed tree-width, DAG-width, and cycle rank are computable in time $O(|V(G)|+|E(G)|)$ and further, that*

$$\textit{d-pw}(G) = \textit{d-tw}(G) = \textit{dagw}(G) - 1 = \textit{cr}(G). \tag{4.5}$$

Proof. It is possible to get all strong components $C_1,\dots,C_r$ of G in linear time. By Lemma 3.5.13, all C_i, $1 \le i \le r$ are directed co-graphs. By [GKR21b], it is possible to get the directed path-width, directed tree-width, DAG-width and cycle rank of directed co-graphs in linear time and it holds that $\text{d-pw}(C_i) = \text{d-tw}(C_i) = \text{dagw}(C_i) - 1 = \text{cr}(C_i)$ for all $1 \le i \le r$. By Lemma 4.3.1, the directed path-width (directed tree-width, DAG-width and cycle rank respectively) of G is the maximum of the directed path-widths (directed tree-widths, DAG-widths and cycle ranks respectively) over all C_i, $1 \le i \le r$. It then follows that those parameters can be computed in linear time and that $\text{d-pw}(G) = \text{d-tw}(G) = \text{dagw}(G) - 1 = \text{cr}(G)$. $\square$

Note that, as twin-dh digraphs are a subclass of extended directed co-graphs, the equality of these graph parameters follows directly from the results in [GKR21b]. However, on extended directed co-graphs, the computation is only possible in linear time if an expression of the digraph is given. Since it is open if an expression of an extended directed co-graph can be computed in linear time, in this case the computation of the parameters without a given expression is still open.

4.3.2 Directed Clique-width

The parameter directed clique-width differs from the previously mentioned parameters as instead of representing the size of strong components in some way, it describes the number of different neighborhoods. Especially for bioriented cliques, the above mentioned parameters are infinitely large whereas directed clique-width is linear. In the undirected case, co-graphs are exactly the graphs of clique-width at most 2 and distance-hereditary graphs have clique-width at most 3. This leads to the idea of regarding directed clique-width on twin-dh digraphs.

Theorem 4.3.3. *Every twin-dh digraph has directed clique-width at most* 3.

Proof. We show a construction for a directed clique-width 3-expression for every $G = (V,E) \in \text{DDH}$ and then argue, why this is best possible. The method we use is closely related to the undirected case: Distance-hereditary graphs have clique-width at most 3, see [GR00]. Let $G \in \text{DDH}$ and $S(G) = (s_1,\dots,s_n)$ be a directed pruning sequence creating G. We give an algorithm to construct a 3-expression traversing $S(G)$ starting with the last element of the sequence. The idea for computing this expression is to use three labels $1,2$ and 3 as follows: After every step of the algorithm, expressions which are already constructed consist of vertices labeled with 1 and 3, where 1 means that the vertex has not been finally treated and possibly has edges to other vertices not inserted yet, whereas for vertices labeled by 3 all incident edges have already been considered. The label 2 is only used as a working label. For initialization, let $X_v = \bullet_1$ for every vertex $v \in V$. As this is a 1-expression, it is also a 3-expression. In the following let now $s_i = (v_i, op_i, a_i)$, for simplification denoted by (v, op, a), be the currently treated element of S and X_v and X_a be the 3-expressions which exists by induction for v and a. Then, we get a 3-expression by the following rules depending on the operation op.

(1) $op = +: \ X_a := \rho_{2\to3}(\alpha_{2,1}(\rho_{1\to2}(X_v) \oplus X_a))$

(2) $op = -: \ X_a := \rho_{2\to3}(\alpha_{1,2}(\rho_{1\to2}(X_v) \oplus X_a))$

(3) $op = \circ: \ X_a := X_v \oplus X_a$

(4) $op = \to: \ X_a := \rho_{2\to1}(\alpha_{2,1}(\rho_{1\to2}(X_v) \oplus X_a))$

(5) $op = \leftarrow: \ X_a := \rho_{2\to1}(\alpha_{1,2}(\rho_{1\to2}(X_v) \oplus X_a))$

(6) $op = \leftrightarrow$: $X_a := \rho_{2\to 1}(\alpha_{1,2}(\alpha_{2,1}(\rho_{1\to 2}(X_v) \oplus X_a)))$

To prove correctness we first need some definitions. For w a vertex of $V(G)$, let $G(w)_i$ be the graph consisting of w and every vertex that is generated by operations on w after step i, which means that $G(w)_i$ is created by the directed pruning sequence $S(w)_i$ which contains elements $s_k = (v_k, op_k, v_{a_k})$ with $k \geq i$ and v_{a_k} has been generated by a series of operations by w. For $i = n$, this means that $G(w)_i = (\{w\}, \emptyset)$. Notice that, for element v_0, which is the first anchor in the directed pruning sequence, i.e., $s_1 = (v_1, op_1, v_0)$, it holds that $S(v_0)_1 = S$ and $G(v_0)_1 = G$. We then show by induction that at any step i with $n \geq i \geq 0$ of the algorithm, it holds that X_w is a 3-expression of $G(w)_i$ for all vertices $w \in V$. We further assume that every X_w contains only vertices labeled by 1 and 3, where the vertices labeled by 1 are exactly those, which are created by a series of twin operations on w (including w itself). To simplify, we call such a vertex a *far twin* in the following.
After the initialization, it is easy to see for $i = n$ that for all $w \in V$, $X_w = \bullet_1$ is a 3-expression of $G(w)_i = (\{w\}, \emptyset)$. Obviously, the only vertex in $G(w)_i$ is w which is labeled by 1.
Consider now step i. By induction we know that X_a is a 3-expression of $G(a)_{i+1}$ where far twins of a are labeled by 1 and all other vertices are labeled by 3. Further, X_v is a 3-expression of $G(v)_{i+1}$ where all far twins of v are labeled by 1 and all other vertices are labeled by 3. We now show that after step i it holds that X_a is a 3-expression of $G(a)_i$. Therefore, we consider element $s_i =: (v, op, a)$ in step i.

(1) As v is a pendant plus vertex of a, there exist edges from every far twin of v to every far twin of a. By $\rho_{1\to 2}(X_v)$ we relabel every vertex in X_v which is labeled by 1, i.e., every far twin of v with 2. We join this expression with X_a and add edges from all labels 2 to 1, which inserts all edges created by the pendant plus relation of v to a. As v is a pendant plus vertex of a, all far twins of a can not be far twins of v and thus, we relabel these vertices from 2 to 3. Now, the newly created X_a is a 3-expression of $G(a)_i$ consisting only of labels 1 and 3, where the vertices labeled by 1 are exactly the far twins of a.

(2) Analogously to (1).

(3) As v is a false twin of a, no new edges are inserted by this operation and further all far twins of v are also far twins of a. Therefore, we only need to join expressions X_v and X_a to create an expression for $G(a)_i$ where every far twin of a is labeled by 1 and every other vertex is labeled by 3.

(4) As v is a true in-twin of a, like in (1), there exist edges from every far twin of v to every far twin of a. We therefore use the same method to join the expressions X_v and X_a and create edges between them. But unlike in (1), every far twin of v is a far twin of a. Therefore, we relabel the far twins of v from 2 to 1, to obtain a 3-expression for $G(a)_i$ in which exactly all far twins of a are labeled by 1.

(5) Analogously to (4).

(6) Is very similar to (4) and (5), with the only difference that we need edges from every far twin of v to every far twin of a and the other way round. For this purpose, we

insert edges from labels 2 to 1 and from labels 1 to 2, before relabeling, to obtain a 3-expression for $G(a)_i$ in which exactly all far twins of a are labeled by 1.

Further, the directed clique-width of a twin-dh digraph has to be at least 3, as can be seen by the following counterexample: The $\overrightarrow{P_3}$, which means a directed path of 3 vertices, is twin-distance-hereditary, but it is not expressible by a 2-expression, see Example 2.2.24. □

By Courcelles Theorem on clique-width, the bounded directed clique-width of these digraphs leads to the computability of every problem, which is describable in monadic second order logic.

Corollary 4.3.4. *Let G be a twin-dh digraph. Then every graph problem, which is describable in MSO_1 logic, is computable in polynomial time on G.*

From the results which are presented later in Subsection 7.6.2, we can follow, that the r-Dichromatic number problem can be solved in polynomial time on twin-dh digraphs. By [GWY16] we can solve the problems Directed Hamiltonian Path, Directed Hamiltonian Cycle, Directed Cut, and Regular Subdigraph using an XP-algorithm w.r.t. the parameter directed NLC-width in polynomial time. Directed NLC-width is a digraph parameter which is closely related to directed clique-width, since we can transform every directed clique-width k-expression into an equivalent NLC-width k-expression, see [GWY16]. Thus, twin-dh digraphs have bounded NLC-width and we can solve the above mentioned problems in polynomial time on this class.

4.3.3 Conclusion

We show that several directed width parameters, namely directed path-width, directed tree-width, DAG-width and cycle rank can be computed in linear time on twin-dh digraphs. From the associated proof (as well as from the fact that this twin-dh digraphs are a subclass of extended directed co-graphs) further the equality of all these parameters follows.

Further, we can conclude by our results in this chapter and Chapter 4 that for twin-dh digraphs, as for directed co-graphs, Kelly-width can be bounded by DAG-width. Due to [HK08, Conjecture 30], [AKK$^+$15], and [BJG18, Section 9.2.5] this remains open for general digraphs and is related to one of the biggest open problems in graph searching, namely whether the monotonicity costs for Kelly- and DAG-width games are bounded.

Like in the undirected case, every twin-dh digraph has directed clique-width at most 3, though not every digraph of directed clique-width 3 is a twin-dh digraph. From that we can conclude several interesting results, since there are many NP-hard problems which are solvable on digraphs of bounded directed clique-width.

It would be interesting for future work to consider other superclasses of twin-dh digraphs and whether it is still possible to find efficient algorithms to compute several graph parameters on these classes and at which point they become NP-hard.

5 NP-hard Problems on Various Recursive Digraph Classes: Subset Sum Problem with Digraph Constraint

5.1 Introduction

The *subset sum problem (SSP)* is one of the simplest and most fundamental NP-hard problems in combinatorial optimization. More formally this means, in SSP there is given a set $A = \{a_1, \ldots, a_n\}$ of n items. Every item a_j has a size s_j and there is a capacity c. All values are assumed to be positive integers and $s_j \leq c$ for every $j \in \{1, \ldots, n\}$. The task is to choose a subset A' of A, such that the sum of the sizes of the items in A' is maximized and is at most c. We consider two extensions of this problem: The *subset sum problem with digraph constraint* (SSG) and *subset sum problem with weak digraph constraint* (SSGW). Both problems have been introduced recently by Gourvès et al. [GMT18]. In both problems there is given a digraph with sizes assigned to the vertices. In the SSG we want to find a subset of vertices whose total size does not exceed a given capacity and which contains a vertex if at least one of its predecessors is part of the solution. Within SSGW we want to find a subset of vertices whose total size does not exceed a given capacity and which contains a vertex if all its predecessors are part of the solution. Since SSG and SSGW generalize SSP, they are NP-hard. Both problems are integer-valued problems, which motivates to observe whether they are weakly NP-hard, i.e., the existence of pseudo-polynomial algorithms.

For related works we refer to [GMT18, Section 3]. In [GMT18] it has been shown that on directed acyclic graphs (DAGs) SSG is strongly NP-hard and SSGW is even APX-hard. Further, they showed that the restriction to oriented trees (OT) allows to give a pseudo-polynomial algorithm using dynamic programming along the tree.

In the following we show that both problems are NP-hard even on oriented co-graphs and minimal series-parallel digraphs. We provide pseudo-polynomial solutions for SSG and SSGW on directed co-graphs and sp-digraphs and deduce a pseudo-polynomial time solution for SSG on series-parallel digraphs. The considered digraph classes are incomparable w.r.t. inclusion to oriented trees considered in [GMT18], see Figure 3.20. Moreover, the digraphs of our interest allow to define dense graphs, i.e., graphs where the number of directed edges is quadratic in the number of vertices.

In Table 5.1 we summarize the known results from [GMT18] and the results of this work about subset sum problems with special digraph constraints.

	SSG		SSGW	
transitive tournaments	$O(n^2)$	Remark 5.4.6	$O(n \cdot c^4 + m)$	Theorem 5.4.12
bioriented cliques	$O(n)$	Remark 5.4.7	$O(n \cdot c^4 + m)$	Theorem 5.4.12
DAGs	strongly NP-hard	[GMT18]	APX-hard	[GMT18]
oriented trees	$O(n \cdot c^3)$	[GMT18]	$O(n \cdot c^2)$	[GMT18]
directed co-graphs	$O(n \cdot c^2 + m)$	Theorem 5.4.5	$O(n \cdot c^4 + m)$	Theorem 5.4.12
minimal series-parallel	$O(n \cdot c^2 + m)$	Theorem 5.5.6	$O(n \cdot c^4 + m)$	Theorem 5.5.10
series-parallel	$O(n \cdot c^2 + n^{2.37})$	Theorem 5.5.7	open	

Table 5.1: Known running times for SSG and SSGW with digraph constraints restricted to special graph classes. Let n be the number of vertices and m the number of directed edges of the input digraph and c be the capacity.

The content of this chapter is from [GKR20b] (except for the outlook).

5.2 Problem Definition

Let $A = \{a_1, \dots, a_n\}$ be a set of n items, such that every item a_j has a size s_j. For a subset A' of A we define

$$s(A') := \sum_{a_j \in A'} s_j$$

and the *capacity constraint* by

$$s(A') \leq c. \tag{5.1}$$

Name: Subset sum problem (SSP)
Instance: A set $A = \{a_1, \dots, a_n\}$ of n items. Every item a_j has a size s_j and there is given a capacity c.
Task: Find a subset A' of A that maximizes $s(A')$ subject to (5.1).

The parameters n, s_j, and c are assumed to be positive integers. For a survey on the subset sum problem we refer to [KPP10, Chapter 4]. In order to consider generalizations of the subset sum problem we will consider constraints for a digraph $G = (A, E)$ with objects assigned to the vertices.

The *digraph constraint* ensures that $A' \subseteq A$ contains a vertex y, if it contains at least one predecessor of y, i.e.

$$\forall y \in A \left(N^-(y) \cap A' \neq \emptyset\right) \Rightarrow y \in A'. \tag{5.2}$$

The *weak digraph constraint* ensures that A' contains a vertex y, if it contains every predecessor of y, i.e.,

$$\forall y \in A \left(N^-(y) \subseteq A' \wedge N^-(y) \neq \emptyset\right) \Rightarrow y \in A'. \tag{5.3}$$

This allows us to state the following optimization problems given in [GMT18].

Name: Subset sum problem with digraph constraint (SSG)
Instance: A set $A = \{a_1, \dots, a_n\}$ of n items and a digraph $G = (A, E)$. Every item a_j has a size s_j and there is a capacity c.
Task: Find a subset A' of A that maximizes $s(A')$ subject to (5.1) and (5.2).

Name: Subset sum problem with weak digraph constraint (SSGW)
Instance: A set $A = \{a_1, \dots, a_n\}$ of n items and a digraph $G = (A, E)$. Every item a_j has a size s_j and there is a capacity c.
Task: Find a subset A' of A that maximizes $s(A')$ subject to (5.1) and (5.3).

In these problems the parameters n, s_j, and c are assumed to be positive integers. The results in [GMT18] also consider null sizes, which are excluded in here. All our solutions can be extended to pseudopolynomial solutions which solve SSG and SSGW using null sizes, see Section 5.6. Further, in the defined problems a subset A' of A is called *feasible*, if it satisfies the prescribed constraints of the problem. By $OPT(I)$ we denote the value of an optimal solution for input I.

Observation 5.2.1. *Every feasible solution for SSG is also a feasible solution for SSGW, but not vice versa.*

Observation 5.2.2. *$A' = \emptyset$ and $A' = A$ for $s(A) \leq c$ are feasible solutions for every instance of SSG and for every instance of SSGW.*

In order to give equivalent characterizations for SSG and SSGW we use binary integer programs.

Remark 5.2.3. To formulate SSG and SSGW as a binary integer program, we introduce a binary variable $x_j \in \{0,1\}$ for each item $a_j \in A$, $1 \leq j \leq n$. The idea is to have $x_j = 1$ if and only if item $a_j \in A'$.

1. SSG corresponds to maximizing $\sum_{j=1}^{n} s_j x_j$ subject to $\sum_{j=1}^{n} s_j x_j \leq c$, $x_i \leq x_j$ for every $j \in \{1, \dots n\}$ and for every $a_i \in N^-(a_j)$, and $x_j \in \{0,1\}$ for every $j \in \{1, \dots n\}$.

2. SSGW corresponds to maximizing $\sum_{j=1}^{n} s_j x_j$ subject to $\sum_{j=1}^{n} s_j x_j \leq c$, $\sum_{\{i \mid a_i \in N^-(a_j)\}} x_i \leq x_j + \text{indegree}(a_j) - 1$ for every $j \in \{1, \dots n\}$, and $x_j \in \{0,1\}$ for every $j \in \{1, \dots n\}$.

The complexity for SSG and SSGW restricted to DAGs and oriented trees was considered in [GMT18].

Theorem 5.2.4 ([GMT18])**.** *On DAGs SSG is strongly NP-hard and SSGW is APX-hard.*

5.3 Basic Results

Let $G = (A, E)$ be a digraph and $x \in A$. By R_x we denote the vertices of A which are reachable from x and by S_x we denote the vertices of A which are in the same strongly connected component as x. Thus, it holds that $\{x\} \subseteq S_x \subseteq R_x \subseteq A$.

Lemma 5.3.1. *Let A' be a feasible solution for SSG on a digraph $G = (A, E)$ and $x \in A$. Then, it holds that $x \in A'$ if and only if $R_x \subseteq A'$.*

Lemma 5.3.2. *Let A' be a feasible solution for SSG on a digraph $G = (A, E)$ and $x \in A$. Then, it holds that $x \in A'$ if and only if $S_x \subseteq A'$.*

Lemma 5.3.3. *SSG is solvable in $O(2^t \cdot (n+m))$ time on digraphs with n vertices, m arcs, and t strongly connected components.*

Proof. By Lemma 5.3.2 for every feasible solution A' and every strongly connected component S, it either holds that $S \subseteq A'$ or $S \cap A' = \emptyset$. Since all strongly connected components are vertex disjoint, we can solve SSG by verifying 2^t possible feasible solutions. Verifying the capacity constraint can be done in $O(n)$ time and verifying the digraph constraint can be done in $O(n+m)$ time. □

In the condensation $con(G)$ of a digraph $G = (V, E)$ every strongly connected component C of G is represented by a vertex v_C. Moreover, there is an arc between two vertices v_C and $v_{C'}$ if there exist $u \in C$ and $v \in C'$, such that $(u, v) \in E$. For every digraph G it holds that $con(G)$ is a directed acyclic graph.

In order to solve SSG it is useful to consider the condensation of the input digraph $G = (A, E)$. By defining the size of a vertex v_C of $con(G)$ by the sum of the sizes of the vertices in C, the following result has been shown in [GMT18, Lemma 2].

Lemma 5.3.4 ([GMT18])**.** *For a given instance of SSG on digraph G, there is a bijection between the feasible solutions (and thus the set of optimal solutions) of SSG for G and the feasible solutions (and thus the set of optimal solutions) for $con(G)$.*

Thus, in order to solve SSG we can restrict ourselves to DAGs by computing the condensation of the input graph in a first step. The next example shows that Lemma 5.3.4 does not hold for SSGW.

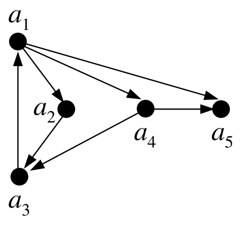


Figure 5.1: Digraph in Example 5.3.5.

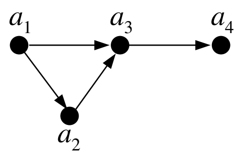


Figure 5.2: Digraph in Example 5.3.7.

Example 5.3.5. We consider the digraph G in Figure 5.1. For SSGW with $c = 2$ and all sizes $s_j = 1$ we have among others $\{a_4\}$ as a feasible solution. Since $con(G)$ is a path of length one, formally

$$con(G) = (\{v_{\{a_1,a_2,a_3,a_4\}}, v_{\{a_5\}}\}, \{(v_{\{a_1,a_2,a_3,a_4\}}, v_{\{a_5\}})\}),$$

the only feasible solution is $\{a_5\}$, which implies that $\{a_4\}$ is not a feasible solution for SSGW using $con(G)$.

The transitive closure $td(G)$ of a digraph G has the same vertex set as G and for two distinct vertices u, v there is an arc (u, v) in $td(G)$ if and only if there is a directed path from u to v in G. The transitive reduction $tr(G)$ of a digraph G has the same vertex set as G and as few arcs of G as possible, such that G and $tr(G)$ have the same transitive closure. The transitive closure is unique for every digraph. The transitive reduction is unique for directed acyclic graphs. However, for arbitrary digraphs the transitive reduction is not unique. The time complexity of the best known algorithm for finding the transitive reduction of a graph is the same as the time to compute the transitive closure of a graph or to perform Boolean matrix multiplication [AGU72]. The best known algorithm to perform Boolean matrix multiplication has running time $O(n^{2.3729})$ by [Gal14].

Lemma 5.3.6. *For a given instance of SSG on a directed acyclic graph G, the set of feasible solutions and thus the set of optimal solutions of SSG for G and for $tr(G)$ are equal.*

Proof. Since a transitive reduction is a subdigraph of the given graph, every feasible solution A' for G is also a feasible solution for the transitive reduction $tr(G)$. To show the reverse direction, let A' be a feasible solution for $tr(G)$. By the definition of $tr(G)$ we know that for every vertex v, every predecessor u of v in G is also a predecessor of v in $tr(G)$ or there is a path from u to v in $tr(G)$. By Lemma 5.3.1 we know that A' is also a feasible solution for G. □

Thus, in order to solve SSG we can restrict ourselves to transitive reductions. The next example shows that Lemma 5.3.6 does not hold for SSGW.

Example 5.3.7. We consider the digraph G in Figure 5.2. For SSGW with $c = 2$ and all sizes $s_j = 1$ we have among others $\{a_2\}$ as a feasible solution. Since $tr(G)$ is a path, formally

$$tr(G) = (\{a_1, a_2, a_3, a_4\}, \{(a_1, a_2), (a_2, a_3), (a_3, a_4)\}),$$

a_2 implies by (5.3) that a_3 and a_4 must be part of the solution, which implies that $\{a_2\}$ is not a feasible solution for SSGW using $tr(G)$.

In the correctness proofs of our algorithms in Sections 5.4 and 5.5 we use the following lemmata.

Lemma 5.3.8. *Let $G = (V_G, E_G)$ be a digraph and let $H = (V_H, E_H)$ be an induced subdigraph of G. If A' is a feasible solution for SSG on G, then $A' \cap V_H$ is a feasible solution for SSG on H.*

Proof. If A' is a feasible solution for SSG on G, then it holds that

$$\forall y \in V_G \left(N_G^-(y) \cap A' \neq \emptyset\right) \Rightarrow y \in A'.$$

By restricting to y having no predecessors from $V_G \setminus V_H$, we obtain

$$\forall y \in V_G \left(N_G^-(y) \cap A' \cap V_H \neq \emptyset\right) \Rightarrow y \in A'.$$

By restricting y to $V_H \subseteq V_G$ we obtain

$$\forall y \in V_H \left(N_H^-(y) \cap A' \cap V_H \neq \emptyset\right) \Rightarrow y \in A' \cap V_H,$$

i.e., $A' \cap V_H$ is a feasible solution for SSG on H. □

The reverse direction of Lemma 5.3.8 does not hold, since vertices with predecessors in $A' \cap (V_G \setminus V_H)$ are not considered by the feasible solutions for SSG on H. By considering the induced subdigraph $H = (\{a_2, a_3, a_4\}, \{(a_2, a_3), (a_3, a_4)\})$ of digraph G in Example 5.3.7 we observe that Lemma 5.3.8 does not hold for SSGW.

Next, we give two weaker forms of Lemma 5.3.8 which also hold for SSGW.

Lemma 5.3.9. *Let $G = (V_G, E_G)$ be a digraph and let $H = (V_H, E_H)$ be a weakly connected component of G. If A' is a feasible solution for SSGW on G, then $A' \cap V_H$ is a feasible solution for SSGW on H.*

Proof. If A' is a feasible solution for SSGW on G, then it holds that

$$\forall y \in V_G \left(N_G^-(y) \subseteq A' \wedge N_G^-(y) \neq \emptyset\right) \Rightarrow y \in A'.$$

By restricting y to $V_H \subseteq V_G$ we obtain

$$\forall y \in V_H \left(N_G^-(y) \subseteq A' \wedge N_G^-(y) \neq \emptyset\right) \Rightarrow y \in A' \cap V_H.$$

Since H is a weakly connected component of G for all $y \in V_H$ it holds that $N_H^-(y) = N_G^-(y)$ such that

$$\forall y \in V_H \left(N_H^-(y) \subseteq A' \cap V_H \wedge N_H^-(y) \neq \emptyset\right) \Rightarrow y \in A' \cap V_H,$$

i.e., $A' \cap V_H$ is a feasible solution for SSGW on H. □

Lemma 5.3.10. *Let $G = (V_G, E_G)$ be a digraph and let $H = (V_H, E_H)$ be an induced subdigraph of G, such that no non-source of H has a predecessor in $V_G \setminus V_H$. If A' is a feasible solution for SSGW on G, then $A' \cap V_H$ is a feasible solution for SSGW on H.*

Proof. If A' is a feasible solution for SSGW on G, then it holds that

$$\forall y \in V_G \left(N_G^-(y) \subseteq A' \wedge N_G^-(y) \neq \emptyset\right) \Rightarrow y \in A'.$$

By restricting y to $V_H \subseteq V_G$ we obtain that

$$\forall y \in V_H \left(N_G^-(y) \subseteq A' \wedge N_G^-(y) \neq \emptyset\right) \Rightarrow y \in A' \cap V_H.$$

By restricting y to be a non-source of H, we obtain

$$\forall y \in V_H \left(N_G^-(y) \subseteq A' \wedge N_G^-(y) \neq \emptyset \wedge N_H^-(y) \neq \emptyset\right) \Rightarrow y \in A' \cap V_H.$$

Since no non-source of H has a predecessor in $V_G \setminus V_H$, we obtain

$$\forall y \in V_H \left(N_G^-(y) \subseteq A' \wedge N_H^-(y) \neq \emptyset\right) \Rightarrow y \in A' \cap V_H.$$

Then it holds in H that

$$\forall y \in V_H \left(N_H^-(y) \subseteq A' \cap V_H \wedge N_H^-(y) \neq \emptyset\right) \Rightarrow y \in A' \cap V_H,$$

i.e., $A' \cap V_H$ is a feasible solution for SSGW on H. □

Lemma 5.3.11. *Let $G = (V_G, E_G)$ be a digraph such that there is a 2-partition (V_1, V_2) of V_G with $\{(u,v) \mid u \in V_1, v \in V_2\} \subseteq E_G$. If A' is a feasible solution for SSGW on G such that $V_1 \subseteq A'$, then $A' \cap V_2$ is a feasible solution for SSGW on $G[V_2]$.*

Proof. If A' is a feasible solution for SSGW on G, then it holds that

$$\forall y \in V_G \left(N_G^-(y) \subseteq A' \wedge N_G^-(y) \neq \emptyset\right) \Rightarrow y \in A'.$$

By restricting y to $V_2 \subseteq V_G$ we obtain

$$\forall y \in V_2 \left(N_G^-(y) \subseteq A' \wedge N_G^-(y) \neq \emptyset\right) \Rightarrow y \in A' \cap V_2.$$

Since $V_1 \subseteq A'$ it holds that

$$\forall y \in V_2 \left(N_G^-(y) \subseteq V_1 \cup A' \cap V_2 \wedge N_G^-(y) \neq \emptyset\right) \Rightarrow y \in A' \cap V_2.$$

Thus, it holds that

$$\forall y \in V_2 \left(N_{G[V_2]}^-(y) \subseteq A' \cap V_2 \wedge (N_{G[V_1]}^-(y) \cup N_{G[V_1]}^-(y)) \neq \emptyset\right) \Rightarrow y \in A' \cap V_2.$$

Since $V_1 = N_{G[V_1]}^-(y) \neq \emptyset$ it holds that

$$\forall y \in V_2 \left(N_{G[V_2]}^-(y) \subseteq A' \cap V_2\right) \Rightarrow y \in A' \cap V_2.$$

By the properties of the logical implication it also holds that

$$\forall y \in V_2 \left(N_{G[V_2]}^-(y) \subseteq A' \cap V_2 \wedge N_{G[V_2]}^-(y) \neq \emptyset\right) \Rightarrow y \in A' \cap V_2,$$

i.e., $A' \cap V_2$ is a feasible solution for SSGW on $G[V_2]$. □

Further, we will use the following result for solutions of SSP on digraphs with sizes assigned to the vertices.

Observation 5.3.12. *Let $G = (V_G, E_G)$ be a digraph with sizes assigned to the vertices and let $H = (V_H, E_H)$ be an induced subdigraph of G. If $A' \subseteq V_G$ satisfies (5.1), then $A' \cap V_H$ satisfies (5.1).*

5.4 SSG and SSGW on Directed Co-graphs

Observation 5.4.1. *Let G be a directed co-graph and T be a di-co-tree for G. For every vertex u of T which corresponds to a series operation, the subtree rooted at u defines a strongly connected subdigraph of G. Further, for every vertex u of T representing a series operation, such that no predecessor of u corresponds to a series operation, the leaves of the subtree rooted at u correspond to a strongly connected component of G.*

Since SSP corresponds to SSG and also to SSGW on a digraph without arcs, which is an oriented co-graph, we obtain the following result.

Proposition 5.4.2. *SSG and SSGW are NP-hard on oriented co-graphs.*

Next, we will show pseudo-polynomial solutions for SSG and SSGW restricted to directed co-graphs. The main idea is a dynamic programming along the recursive structure of a given directed co-graph.

5.4.1 Subset Sum with Digraph Constraint (SSG)

By Lemma 5.3.4 in order to solve SSG we can restrict ourselves to directed acyclic graphs. This can be done by replacing every strongly connected component S by a new vertex x_S whose size is the sum of the sizes of the vertices in S. In order to identify the strongly connected components of directed co-graphs using a di-co-tree we apply Observation 5.4.1. We perform a breadth first search on a di-co-tree T starting at the root and for every vertex u of T which corresponds to a series operation we substitute the subtree rooted at u by a single vertex whose size is the sum of the sizes of the vertices corresponding to the leaves of the subtree rooted at u. This does not reduce the size of the digraph or its di-co-tree in general, e.g. for oriented co-graphs we have no non-trivial strongly connected component.

We consider an instance of SSG such that $G = (A, E)$ is a directed co-graph which is given by some di-co-expression X. For some subexpression X' of X let $F(X', s) = 1$ if there is a solution A' in the graph defined by X' satisfying (5.1) and (5.2) such that $s(A') = s$, otherwise let $F(X', s) = 0$. We use the notation $s(X') = \sum_{a_j \in X'} s_j$.

Lemma 5.4.3. *Let $0 \leq s \leq c$.*

1. *$F(a_j, s) = 1$ if and only if $s = 0$ or $s_j = s$. In all other cases $F(a_j, s) = 0$.*
2. *$F(X_1 \oplus X_2, s) = 1$, if and only if there are some $0 \leq s' \leq s$ and $0 \leq s'' \leq s$ such that $s' + s'' = s$ and $F(X_1, s') = 1$ and $F(X_2, s'') = 1$. In all other cases $F(X_1 \oplus X_2, s) = 0$.*
3. *$F(X_1 \oslash X_2, s) = 1$, if and only if*
   - *$F(X_2, s) = 1$ for $0 \leq s \leq s(X_2)$[1] or*
   - *there is an $s' > 0$, such that $s = s' + s(X_2)$ and $F(X_1, s') = 1$.*

[1]The value $s = 0$ is for choosing an empty solution in digraph$(X_1 \oslash X_2)$.

In all other cases $F(X_1 \oslash X_2, s) = 0$.

4. $F(X_1 \otimes X_2, s) = 1$, *if and only if* $s = 0$ *or* $s = s(X_1) + s(X_2)$. *In all other cases* $F(X_1 \otimes X_2, s) = 0$.

Proof. We show the correctness of the stated equivalences. Let $0 \leq s \leq c$.

1. The only possible solutions in digraph(a_j) are $\emptyset$ and $\{a_j\}$ which have size 0 and s_j, respectively.

2. If $F(X_1 \oplus X_2, s) = 1$, then by Lemma 5.3.8 there are s' and s'' such that $s' + s'' = s$ and solutions in digraph(X_1) and in digraph(X_2) which guarantee $F(X_1, s') = 1$ and $F(X_2, s'') = 1$. Further, for every s' and s'', such that $s' + s'' = s$, $F(X_1, s') = 1$, and $F(X_2, s'') = 1$, it holds that $F(X_1 \oplus X_2, s) = 1$ since the operation (disjoint union) does not create new edges.

3. If $F(X_1 \oslash X_2, s) = 1$, then we distinguish two cases. If the solution of size s in digraph$(X_1 \oslash X_2)$ contains no vertices of digraph(X_1), then by Lemma 5.3.8 there is a solution in digraph(X_2) which guarantees $F(X_2, s) = 1$. If the solution A' of size s in digraph$(X_1 \oslash X_2)$ contains at least one vertex of digraph(X_1), then by (5.2) solution A' has to contain all vertices of digraph(X_2) and by Lemma 5.3.8 there is a solution in digraph(X_1) which guarantees $F(X_1, s - s(X_2)) = 1$. Further, for every $0 \leq s \leq s(X_2)$ where $F(X_2, s) = 1$ we have $F(X_1 \oslash X_2, s) = 1$ since the solutions from digraph(X_2) do not contain any predecessors of vertices from digraph(X_1) in digraph$(X_1 \oslash X_2)$. Also for every $1 \leq s' \leq s(X_1)$ where $F(X_1, s') = 1$ for $s = s' + s(X_2)$ we have $F(X_1 \oslash X_2, s) = 1$ since every solution in digraph(X_1) has to be extended by X_2 since at least one predecessor of digraph(X_2) is part of the solution and thus, all vertices of digraph(X_2) have to belong to the solution.

4. If $F(X_1 \otimes X_2, s) = 1$, then we distinguish two cases. If the solution of size s is empty, then $s = 0$. Otherwise, $s = s(X_1) + s(X_2)$ since digraph$(X_1 \otimes X_2)$ is strongly connected and thus, all vertices of digraph(X_1) and all vertices of digraph(X_2) have to be part of the solution.
   Further, if $s = 0$ or $s = s(X_1) + s(X_2)$, it holds that $F(X_1 \otimes X_2, s) = 1$ since the empty and the complete vertex set both satisfy (5.2).

□

Corollary 5.4.4. *There is a solution with sum s for an instance of SSG such that G is a directed co-graph which is given by some di-co-expression X if and only if* $F(X, s) = 1$. *Therefore,* $OPT(I) = \max\{s \mid F(X, s) = 1\}$.

Theorem 5.4.5. *SSG can be solved in directed co-graphs with n vertices and m arcs in* $O(n \cdot c^2 + m)$ *time and* $O(n \cdot c)$ *space.*

Proof. Let $G = (A, E)$ be a directed co-graph and T be a di-co-tree for G with root r. For some vertex u of T we denote by T_u the subtree rooted at u and X_u the co-expression defined

by T_u. In order to solve the SSG problem for an instance I on graph G, we traverse di-co-tree T into bottom-up order. For every vertex u of T and $0 \leq s \leq c$ we compute $F(X_u,s)$ following the rules given in Lemma 5.4.3. By Corollary 5.4.4 we can solve our problem by $F(X_r,s) = F(X,s)$.
A di-co-tree T can be computed in $O(n+m)$ time from a directed co-graph with n vertices and m arcs, see Theorem 3.3.4. All $s(X_i)$ can be precomputed in $O(n)$ time. Our rules given in Lemma 5.4.3 show the following running times.

- For every $a_j \in A$ and every $0 \leq s \leq c$ value $F(a_j,s)$ is computable in $O(1)$ time.
- For every $0 \leq s \leq c$, every $F(X_1 \oplus X_2,s)$ and every $F(X_1 \oslash X_2,s)$ can be computed in $O(c)$ time from $F(X_1,s')$ and $F(X_2,s'')$.
- For every $0 \leq s \leq c$, every $F(X_1 \otimes X_2,s)$ can be computed in $O(1)$ time from $s(X_1)$ and $s(X_2)$.

Since we have n leaves and $n-1$ inner vertices in T, the running time is in $O(nc^2+m)$. $\square$

In [GMT18, Lemma 4] it is shown that SSG is polynomial on acyclic tournaments without stating a running time. Since acyclic tournaments, and equivalently transitive tournaments, are a subclass of oriented co-graphs, we reconsider the following result.

Remark 5.4.6. Every transitive tournament G can be defined from a single vertex graph v_1 by repeatedly adding a vertex of maximum indegree and outdegree 0, i.e., an in-dominated vertex $v_2,\ldots,v_n$ (cf. Lemma 2.3.1). This order can be defined in $O(n^2)$ time from G. The feasible solutions w.r.t. the digraph constraint (5.2) are $\emptyset$ and for $1 \leq k \leq n$ the set $\{v_i \mid k \leq i \leq n\}$. This leads to at most $n+1$ possible solutions for SSG for which we have to check the capacity constraint (5.1) and among those satisfying (5.1) we select one set with largest sum of sizes. Thus, SSG is solvable in $O(n^2)$ time on transitive tournaments with n vertices.

Remark 5.4.7. Within a bioriented clique $G = (A,E)$ the whole vertex set is a strongly connected component. By Lemma 5.3.2 the only possible solutions are A and $\emptyset$. Thus, SSG is solvable in $O(n)$ time on bioriented cliques with n vertices.

5.4.2 Subset Sum With Weak Digraph Constraint (SSGW)

Next, we consider SSGW on directed co-graphs. In order to get useful information about the sources within a solution, we use an extended data structure. We consider an instance of SSGW such that $G = (A,E)$ is a directed co-graph which is given by some di-co-expression X. For some subexpression X' of X let $H(X',s,s') = 1$ if there is a solution A' in the digraph defined by X' satisfying (5.1) and (5.3) such that $s(A') = s$ and the sum of sizes of the sources in A' is s', otherwise let $H(X',s,s') = 0$. We denote by $o(X)$ the sum of the sizes of all sources in digraph(X).

Remark 5.4.8. A remarkable difference between SSGW and SSG w.r.t. co-graph operations is the following. When considering $X_1 \oslash X_2$ we can combine solutions A_1 of X_1 satisfying (5.1) and (5.3) which do not contain all items of X_1 with solutions A_2 of X_2 satisfying only (5.1) to obtain solution $A_1 \cup A_2$ of $X_1 \oslash X_2$ satisfying (5.1) and (5.3), if $s(A_1)+s(A_2) \leq c$.

Furthermore, within $X_1 \otimes X_2$ we can combine solutions A_1 of X_1 satisfying (5.1) which do not contain all items and solutions A_2 of X_2 satisfying (5.1) which do not contain all items to obtain solution $A_1 \cup A_2$ of $X_1 \otimes X_2$ satisfying (5.1) and (5.3), if $s(A_1) + s(A_2) \leq c$.

Thus, in order to solve SSGW on a directed co-graph G, we use solutions for SSP on subexpressions for G. We consider an instance of SSP such that $G = (A, E)$ is a directed co-graph which is given by some di-co-expression X. For a subexpression X' of X let $H'(X', s) = 1$ if there is a solution A' in the digraph defined by X' satisfying (5.1) such that $s(A') = s$, otherwise let $H'(X', s) = 0$.

Lemma 5.4.9. *Let* $0 \leq s \leq c$.

1. $H'(a_j, s) = 1$ *if and only if* $s = 0$ *or* $s = s_j$. *In all other cases* $H'(a_j, s) = 0$.
2. $H'(X_1 \oplus X_2, s) = 1$, *if and only if there are some* $0 \leq s' \leq s$ *and* $0 \leq s'' \leq s$ *such that* $s' + s'' = s$ *and* $H'(X_1, s') = 1$ *and* $H'(X_2, s'') = 1$.
   In all other cases $H'(X_1 \oplus X_2, s) = 0$.
3. $H'(X_1 \oslash X_2, s) = H'(X_1 \oplus X_2, s)$
4. $H'(X_1 \otimes X_2, s) = H'(X_1 \oplus X_2, s)$

Proof. We show the correctness of the stated equivalences. Let $0 \leq s \leq c$.

1. The only possible solutions in digraph(a_j) are $\emptyset$ and $\{a_j\}$ which have size 0 and s_j, respectively.
2. If $H'(X_1 \oplus X_2, s) = 1$, then by Observation 5.3.12 there are s' and s'' such that $s' + s'' = s$ and solutions in digraph(X_1) and in digraph(X_2) which guarantee $H'(X_1, s') = 1$ and $H'(X_2, s'') = 1$. Further, for every s' and s'', such that $s' + s'' = s$, $H'(X_1, s') = 1$, and $H'(X_2, s'') = 1$, we can combine these two solutions into one solution of size s in digraph$(X_1 \oplus X_2)$. Thus, it holds that $H'(X_1 \oplus X_2, s) = 1$.
3. Since the arcs are irrelevant for the capacity constraint (5.1), it holds that $H'(X_1 \oslash X_2, s) = H'(X_1 \oplus X_2, s)$.
4. Since the arcs are irrelevant for the capacity constraint (5.1), it holds that $H'(X_1 \otimes X_2, s) = H'(X_1 \oplus X_2, s)$.

□

This allows us to compute the values $H(X', s, s')$ as follows.

Lemma 5.4.10. *Let* $0 \leq s, s' \leq c$.

1. $H(a_j, s, s') = 1$ *if and only if* $s = s' = 0$ *or* $s_j = s = s'$.
   In all other cases $H(a_j, s, s') = 0$.
2. $H(X_1 \oplus X_2, s, s') = 1$, *if and only if there are* $0 \leq s_1 \leq s$, $0 \leq s_2 \leq s$, $0 \leq s'_1 \leq s'$, $0 \leq s'_2 \leq s'$, *such that* $s_1 + s_2 = s$, $s'_1 + s'_2 = s'$, $H(X_1, s_1, s'_1) = 1$, *and* $H(X_2, s_2, s'_2) = 1$.
   In all other cases $H(X_1 \oplus X_2, s, s') = 0$.

3. $H(X_1 \oslash X_2, s, s') = 1$, *if and only if*

   - $H(X_1, s, s') = 1$ *for* $1 \leq s < s(X_1)$ *or*
   - $H'(X_2, s) = 1$ *for* $0 \leq s \leq s(X_2)$[2] *and* $s' = 0$ *or*
   - *there are* $1 \leq s_2 \leq s(X_2)$, *such that* $s(X_1) + s_2 = s$, $o(X_1) = s'$, *and* $H(X_2, s_2, o(X_2)) = 1$, *or*
   - $s = s(X_1) + s(X_2)$ *and* $s' = o(X_1)$, *or*
   - *there are* $0 \leq s_1 < s(X_1)$, $0 \leq s_2 \leq s(X_2)$, *such that* $s_1 + s_2 = s$, $H(X_1, s_1, s') = 1$, *and* $H'(X_2, s_2) = 1$.

   In all other cases $H(X_1 \oslash X_2, s, s') = 0$.

4. $H(X_1 \otimes X_2, s, 0) = 1$, *if and only if*

   - $H'(X_1, s) = 1$ *for* $1 \leq s < s(X_1)$ *or*
   - $H'(X_2, s) = 1$ *for* $0 \leq s < s(X_2)$[3] *or*
   - *there are* $1 \leq s_2 \leq s(X_2)$, *such that* $s(X_1) + s_2 = s$, *and* $H(X_2, s_2, o(X_2)) = 1$, *or*
   - *there are* $1 \leq s_1 \leq s(X_1)$, *such that* $s_1 + s(X_2) = s$, *and* $H(X_1, s_1, o(X_1)) = 1$, *or*
   - $s = s(X_1) + s(X_2)$, *or*
   - *there exist* $1 \leq s_1 < s(X_1)$ *and* $1 \leq s_2 < s(X_2)$ *such that* $s_1 + s_2 = s$, $H'(X_1, s_1) = 1$, *and* $H'(X_2, s_2) = 1$.

   In all other cases $H(X_1 \otimes X_2, s, s') = 0$.

Proof. We show the correctness of the stated equivalences. Let $0 \leq s, s' \leq c$.

1. The only possible solutions in digraph(a_j) are $\emptyset$ and $\{a_j\}$ which have size 0 and s_j, respectively. Further, a single vertex is a source.

2. If $H(X_1 \oplus X_2, s, s') = 1$, then by Lemma 5.3.9 there are s_1, s_2 and s'_1, s'_2 such that $s_1 + s_2 = s$, $s'_1 + s'_2 = s'$ and solutions in digraph(X_1) and in digraph(X_2) which guarantee $H(X_1, s_1, s'_1) = 1$ and $H(X_2, s, s'_2) = 1$.
   Further, for every $0 \leq s_1 \leq s$, $0 \leq s_2 \leq s$, $0 \leq s'_1 \leq s'$, $0 \leq s'_2 \leq s'$, such that $s_1 + s_2 = s$, $s'_1 + s'_2 = s'$, $H(X_1, s_1, s'_1) = 1$, and $H(X_2, s_2, s'_2) = 1$, it holds that $H(X_1 \oplus X_2, s, s') = 1$ since the operation (disjoint union) does not create new edges.

3. If $H(X_1 \oslash X_2, s, s') = 1$, then we distinguish the following cases. If the solution of size s in digraph$(X_1 \oslash X_2)$ is a non-empty proper subset of the vertices of digraph(X_1), then by Lemma 5.3.10 there is a solution in digraph(X_1) which guarantees $H(X_1, s, s') = 1$. Next, assume that the solution A' of size s in digraph$(X_1 \oslash X_2)$ contains no vertices of digraph(X_1). Since every solution satisfying constraints (5.1) and (5.3) is also a solution which satisfies only (5.1), we have $H'(X_1 \oslash X_2, s) = 1$. And since A' contains no vertices

[2]The value $s = 0$ is for choosing an empty solution in digraph$(X_1 \oslash X_2)$.
[3]The value $s = 0$ is for choosing an empty solution in digraph$(X_1 \otimes X_2)$.

of digraph(X_1), Observation 5.3.12 implies that there is a solution in digraph(X_2) which guarantees $H'(X_2,s) = 1$.
If the solution A' of size s in digraph$(X_1 \oslash X_2)$ contains all vertices of digraph(X_1), then the order composition and the weak digraph constraint (5.3) imply that the set A' can be extended by every solution of digraph(X_2) which includes all sources of digraph(X_2). Thus, by Lemma 5.3.11, there is a solution in digraph(X_2), which guarantees $H(X_2,s-s(X_1),o(X_2)) = 1$.
Further, if the solution A' of size s in digraph$(X_1 \oslash X_2)$ contains all vertices of digraph(X_1), it is also possible to extend A' by all vertices of digraph(X_2) and thus $s = s(X_1)+s(X_2)$.
Finally, if the solution A' of size s in digraph$(X_1 \oslash X_2)$ contains some but not all vertices of digraph(X_1) and possibly vertices of digraph(X_2), then by Lemma 5.3.10 and Observation 5.3.12 there are s_1 and s_2 such that $s_1+s_2 = s$ and solutions in digraph(X_1) and in digraph(X_2) which guarantee $H(X_1,s_1,s') = 1$ and $H'(X_2,s_2) = 1$.
The solutions of size $1 \leq s < s(X_1)$ from digraph(X_1) remain feasible for digraph$(X_1 \oslash X_2)$.
Every subset A' of size $0 \leq s \leq s(X_2)$ from digraph(X_2) which satisfies (5.1) leads to a solution A' of size s satisfying (5.1) and (5.3) in digraph$(X_1 \oslash X_2)$ since every vertex of digraph(X_2) gets a predecessor in digraph(X_1), which is not in A'.
Further, the set of all vertices of digraph(X_1) extended by every solution of digraph(X_2) of size s_2 which includes all sources of digraph(X_2) leads to a feasible solution for digraph$(X_1 \oslash X_2)$ of size $s(X_1)+s_2$. The size of the sources has to be updated to $o(X_1)$, since the sources of digraph(X_1) are the sources of digraph$(X_1 \oslash X_2)$.
Moreover, the complete vertex set of digraph$(X_1 \oslash X_2)$ is obviously a feasible SSGW solution if it fulfills the capacity constraint.
Furthermore, by Remark 5.4.8 we can combine SSGW solutions of size $s_1 < s(X_1)$ of digraph(X_1) and SSP solutions of size s_2 of digraph(X_2) to a SSGW solution of size s_1+s_2 of digraph$(X_1 \oslash X_2)$.

4. First, we want to mention that $H(X_1 \otimes X_2,s,s') = 1$ is only possible for $s' = 0$, since digraph$(X_1 \otimes X_2)$ has no sources. If $H(X_1 \otimes X_2,s,0) = 1$, we distinguish the following cases. Assume the solution of size s in digraph$(X_1 \otimes X_2)$ is a proper and non-empty subset of the vertices of digraph(X_1). Since $H(X_1 \otimes X_2,s,0) = 1$, it holds that $H'(X_1 \otimes X_2,s) = 1$. And since A' contains only vertices of digraph(X_1), Observation 5.3.12 implies that there is a solution in digraph(X_1) which guarantees $H'(X_1,s) = 1$.
If the solution of size s in digraph$(X_1 \otimes X_2)$ is a proper subset of the vertices of digraph(X_2), then by the same arguments as for digraph(X_1) there is a solution in digraph(X_2) which guarantees $H'(X_2,s) = 1$.
If the solution A' of size s in digraph$(X_1 \otimes X_2)$ contains all vertices of digraph(X_1), then the series composition and the weak digraph constraint (5.3) imply that the set A' can be extended by all solutions of digraph(X_2) which include all sources of digraph(X_2). Thus, by Lemma 5.3.11, there is a solution in digraph(X_2), which guarantees $H(X_2,s-s(X_1),o(X_2))$.
If the solution A' of size s in digraph$(X_1 \otimes X_2)$ contains all vertices of digraph(X_2), then by the same arguments as for digraph(X_1) there is a solution in digraph(X_1), which

guarantees $H(X_1, s - s(X_2), o(X_1))$.
If the solution A' of size s in digraph$(X_1 \otimes X_2)$ contains all vertices of digraph(X_1) or all vertices of digraph(X_2), then by (5.3) solution A' can be extended by all vertices of digraph(X_2) or all vertices of digraph(X_1), respectively, and thus $s = s(X_1) + s(X_2)$.
Finally, if the solution A' of size s in digraph$(X_1 \otimes X_2)$ contains some but not all vertices of digraph(X_1) and some but not all vertices of digraph(X_2), then by Observation 5.3.12 there are s_1 and s_2 such that $s_1 + s_2 = s$ and solutions in digraph(X_1) and in digraph(X_2) which guarantee $H'(X_1, s_1) = 1$ and $H'(X_2, s_2) = 1$.
Every subset A' of size $1 \leq s < s(X_1)$ from digraph(X_1) which satisfies (5.1) leads to a solution A' of size s satisfying constraints (5.1) and (5.3) in digraph$(X_1 \otimes X_2)$ since every vertex of digraph(X_1) gets a predecessor in digraph(X_2), which is not in A'.
In the same way every subset A' of size $0 \leq s < s(X_2)$ from digraph(X_2) which satisfies (5.1) leads to a solution A' of size s satisfying (5.1) and (5.3) in digraph$(X_1 \otimes X_2)$ since every vertex of digraph(X_2) gets a predecessor in digraph(X_1), which is not in A'.
Further, all the set of all vertices of digraph(X_1) extended by every solution of digraph(X_2) of size s_2 which includes all sources of digraph(X_2) leads to a feasible solution for digraph$(X_1 \otimes X_2)$ of size $s(X_1) + s_2$ and the set of all vertices of digraph(X_2) extended by every solution of digraph(X_1) of size s_1 which includes all sources of digraph(X_1) leads to a feasible solution for digraph$(X_1 \otimes X_2)$ of size $s_1 + s(X_2)$.
Moreover, the complete vertex set of digraph$(X_1 \otimes X_2)$ is obviously a feasible SSGW solution.
Furthermore, by Remark 5.4.8, we can combine SSP solutions of size $s_1 < s(X_1)$ and SSP solutions of size $s_2 < s(X_2)$ to a SSGW solution of size $s_1 + s_2$.

□

In order to solve the SSGW problem we traverse di-co-tree T into bottom-up order and perform the following computations depending on the type of operation.

Corollary 5.4.11. *There is a solution with sum s for an instance of SSGW such that G is a directed co-graph which is given by some di-co-expression X if and only if $H(X, s, s') = 1$. Therefore, $OPT(I) = \max\{s \mid H(X, s, s') = 1\}$.*

The next result can be obtained by similar arguments as given within the proof of Theorem 5.4.5.

Theorem 5.4.12. *SSGW can be solved in directed co-graphs with n vertices and m arcs in $O(n \cdot c^4 + m)$ time and $O(n \cdot c^2)$ space.*

5.5 SSG and SSGW on Series-parallel Digraphs

We now look at SSG and SSGW on (minimal) series-parallel digraphs.

5.5.1 Subset Sum with Digraph Constraint (SSG)

Lemma 5.5.1. *Let G be a minimal series-parallel digraph. Then, for every vertex $x \in V(G)$ there is a sink x_s of G, such that there is a directed path from x to x_s in G and there is a source x_o of G, such that there is a path from x_o to x in G.*

Proof. Can be shown by induction on the recursive definition of minimal series-parallel digraphs. □

Lemma 5.5.2. *Let $G = (A, E)$ be a minimal series-parallel digraph. Then, every non-empty feasible solution of SSG contains a sink of G.*

Proof. If a feasible solution A' contains some $x \in A$, then Lemma 5.5.1 implies that there is a sink x_s of G, such that there is a path from x to x_s in G which implies by (5.2) that $x_s \in A'$. □

Since SSP corresponds to SSG and also to SSGW on a digraph without arcs, which is a minimal series-parallel digraph, we obtain the following result.

Proposition 5.5.3. *SSG and SSGW are NP-hard on minimal series-parallel digraph.*

Next, we show pseudo-polynomial solutions for SSG and SSGW restricted to (minimal) series-parallel digraphs. The main idea is a dynamic programming along the recursive structure of a given (minimal) series-parallel digraph. We consider an instance of SSG such that $G = (A, E)$ is a minimal series-parallel digraph which is given by some msp-expression X. For some subexpression X' of X let $F(X', s) = 1$ if there is a solution A' in the graph defined by X' satisfying (5.1) and (5.2) such that $s(A') = s$, otherwise let $F(X', s) = 0$. We use the notation $s(X') = \sum_{a_j \in X'} s_j$.

Lemma 5.5.4. *Let $0 \le s \le c$.*

1. *$F(a_j, s) = 1$ if and only if $s = 0$ or $s_j = s$. In all other cases $F(a_j, s) = 0$.*
2. *$F(X_1 \cup X_2, s) = 1$, if and only if there are some $0 \le s' \le s$ and $0 \le s'' \le s$ such that $s' + s'' = s$ and $F(X_1, s') = 1$ and $F(X_2, s'') = 1$.*
   In all other cases $F(X_1 \cup X_2, s) = 0$.
3. *$F(X_1 \times X_2, s) = 1$, if and only if*
   - *$F(X_2, s) = 1$ for $0 \le s \le s(X_2)$[4] or*
   - *there is some $1 \le s' \le s(X_1)$ such that $s = s' + s(X_2)$ and $F(X_1, s') = 1$.*

   In all other cases $F(X_1 \times X_2, s) = 0$.

Proof. We show the correctness of the stated equivalences. Let $0 \le s \le c$.

1. The only possible solutions in digraph(a_j) are $\emptyset$ and $\{a_j\}$ which have size 0 and s_j, respectively.

[4]The value $s = 0$ is for choosing an empty solution in digraph($X_1 \times X_2$).

2. If $F(X_1 \cup X_2, s) = 1$, then by Lemma 5.3.8 there are s' and s'' such that $s' + s'' = s$ and solutions in digraph(X_1) and in digraph(X_2) which guarantee $F(X_1, s') = 1$ and $F(X_2, s'') = 1$.
   Further, for every s' and s'', such that $s' + s'' = s$, $F(X_1, s') = 1$, and $F(X_2, s'') = 1$, it holds that $F(X_1 \cup X_2, s) = 1$ since the parallel composition creates no additional arcs.

3. If $F(X_1 \times X_2, s) = 1$, then we distinguish two cases. If the solution of size s in digraph$(X_1 \times X_2)$ contains no vertex of digraph(X_1), then by Lemma 5.3.8 there is a solution in digraph(X_2) which guarantees $F(X_2, s) = 1$.
   Otherwise, the solution A' of size s in digraph$(X_1 \times X_2)$ contains at least one vertex of digraph(X_1). By the definition of the series composition and the digraph constraint (5.2) every solution from digraph(X_1) which contains a sink has to be extended by every vertex of X_2 which is reachable by a source from digraph(X_2). Since by Lemma 5.5.2 every non-empty feasible solution of SSG contains a sink, every solution from digraph(X_1) has to be extended by every vertex of X_2 which reachable by a source from digraph(X_2). By Lemma 5.5.1 every solution from digraph(X_1) has to be extended by all vertices of digraph(X_2). Thus, by Lemma 5.3.8 there is a solution in digraph(X_1) which guarantees $F(X_1, s - s(X_2)) = 1$.
   Further, for every $0 \leq s \leq s(X_2)$ where $F(X_2, s) = 1$ we have $F(X_1 \times X_2, s) = 1$ since the solutions from digraph(X_2) do not contain any predecessors of vertices from digraph(X_1) in digraph$(X_1 \times X_2)$.
   For every $1 \leq s' \leq s(X_1)$ where $F(X_1, s') = 1$ the definition of the series composition and the digraph constraint (5.2) imply that for $s = s' + s(X_2)$ it holds that $F(X_1 \times X_2, s) = 1$ for reasons given above.

□

Corollary 5.5.5. *There is a solution with sum s for some instance of SSG such that G is a minimal series-parallel digraph which is given by some msp-expression X if and only if $F(X,s) = 1$. Therefore, $OPT(I) = \max\{s \mid F(X,s) = 1\}$.*

Theorem 5.5.6. *SSG can be solved in minimal series-parallel digraphs with n vertices and m arcs in $O(n \cdot c^2 + m)$ time and $O(n \cdot c)$ space.*

Proof. Let $G = (V, E)$ be a minimal series-parallel digraph and T be an msp-tree for G with root r. For some vertex u of T we denote by T_u the subtree rooted at u and X_u the msp-expression defined by T_u. In order to solve the SSG problem for an instance I graph G, we traverse msp-tree T into bottom-up order. For every vertex u of T and $0 \leq s \leq c$ we compute $F(X_u, s)$ following the rules given in Lemma 5.5.4. By Corollary 5.5.5 we can solve our problem by $F(X_r, s) = F(X, s)$.
According to Observation 3.6.3 an msp-tree T can be computed in linear time w.r.t. vertices and edges from a minimal series-parallel digraph. All $s(X_i)$ can be precomputed in $O(n)$ time. Our rules given in Lemma 5.5.4 show the following running times.

- For every $a_j \in V$ and every $0 \leq s \leq c$ value $F(a_j, s)$ is computable in $O(1)$ time.

- For every $0 \leq s \leq c$, every $F(X_1 \cup X_2, s)$ can be computed in $O(c)$ time from $F(X_1, s')$ and $F(X_2, s'')$.
- For every $0 \leq s \leq c$, every $F(X_1 \times X_2, s)$ can be computed in $O(1)$ time from $F(X_1, s')$, $F(X_2, s'')$, and $s(X_2)$.

Since we have n leaves and $n-1$ inner vertices in T, the running time is in $O(nc^2 + m)$. $\square$

Theorem 5.5.7. *SSG can be solved in series-parallel digraphs with n vertices and m arcs in $O(n \cdot c^2 + n^{2.3729})$ time and $O(n \cdot c)$ space.*

Proof. Let G be some series-parallel digraph. By Lemma 5.3.6 we can use the transitive reduction of G, which can be computed in $O(n^{2.3729})$ time by [Gal14]. $\square$

5.5.2 Subset Sum with Weak Digraph Constraint (SSGW)

Next, we consider SSGW on minimal series-parallel digraph. In order to get useful information's about the sinks within a solution, we use an extended data structure. We consider an instance of SSGW such that $G = (A, E)$ is a minimal series-parallel digraph which is given by some msp-expression X. For some subexpression X' of X let $H(X', s, s') = 1$ if there is a solution A' in the graph defined by X' satisfying (5.1) and (5.3) such that $s(A') = s$ and the sum of sizes of the sinks in A' is s', otherwise let $H(X', s, s') = 0$. We denote by $i(X)$ the sum of the sizes of all sinks in digraph(X).

Lemma 5.5.8. *Let $0 \leq s, s' \leq c$.*

1. *$H(a_j, s, s') = 1$ if and only if $s = s' = 0$ or $s_j = s = s'$.
   In all other cases $H(a_j, s, s') = 0$.*
2. *$H(X_1 \cup X_2, s, s') = 1$, if and only if there are $0 \leq s_1 \leq s$, $0 \leq s_2 \leq s$, $0 \leq s'_1 \leq s'$, $0 \leq s'_2 \leq s'$, such that $s_1 + s_2 = s$, $s'_1 + s'_2 = s'$, $H(X_1, s_1, s'_1) = 1$, and $H(X_2, s_2, s'_2) = 1$.
   In all other cases $H(X_1 \cup X_2, s, s') = 0$.*
3. *$H(X_1 \times X_2, s, s') = 1$, if and only if*
   - *$0 \leq s \leq s(X_2)$[5] and $0 \leq s' \leq s(X_2)$, such that $H(X_2, s, s') = 1$ or*
   - *there are $1 \leq s_1 \leq s(X_1)$ and $1 \leq s'_1 < i(X_1)$, such that $s_1 = s$, $0 = s'$, and $H(X_1, s_1, s'_1) = 1$, or*
   - *there are $1 \leq s_1 \leq s(X_1)$, such that $s_1 + s(X_2) = s$, $i(X_2) = s'$, and $H(X_1, s_1, i(X_1)) = 1$, or*
   - *there are $1 \leq s_1 \leq s(X_1)$, $1 \leq s'_1 < i(X_1)$, $1 \leq s_2 \leq s(X_2)$, and $1 \leq s'_2 \leq s(X_2)$, such that $s_1 + s_2 = s$, $s'_2 = s'$, $H(X_1, s_1, s'_1) = 1$, and $H(X_2, s_2, s'_2) = 1$.*

   In all other cases $H(X_1 \times X_2, s, s') = 0$.

[5]The value $s = s' = 0$ is for choosing an empty solution in digraph$(X_1 \times X_2)$. The values $s > s' = 0$ are for choosing a solution without sinks in digraph$(X_1 \times X_2)$

Proof. We show the correctness of the stated equivalences. Let $0 \leq s, s' \leq c$.

1. The only possible solutions in digraph(a_j) are $\emptyset$ and $\{a_j\}$ which have size 0 and s_j, respectively. Further, a single vertex corresponds to a sink.

2. If $H(X_1 \cup X_2, s, s') = 1$, then by Lemma 5.3.9 there are s_1, s_2 and s'_1, s'_2 such that $s_1 + s_2 = s$, $s'_1 + s'_2 = s'$ and solutions in digraph(X_1) and in digraph(X_2) which guarantee $H(X_1, s_1, s'_1) = 1$ and $H(X_2, s, s'_2) = 1$.

   Further, for every $0 \leq s_1 \leq s$, $0 \leq s_2 \leq s$, $0 \leq s'_1 \leq s'$, $0 \leq s'_2 \leq s'$, such that $s_1 + s_2 = s$, $s'_1 + s'_2 = s'$, $H(X_1, s_1, s'_1) = 1$, and $H(X_2, s_2, s'_2) = 1$, it holds that $H(X_1 \cup X_2, s, s') = 1$ since we do not create any new edges by the parallel composition.

3. If $H(X_1 \times X_2, s, s') = 1$, then we distinguish four cases. If the solution of size s and sink size s' in digraph$(X_1 \times X_2)$ contains no vertices of digraph(X_1), then by Lemma 5.3.10 there is a solution in digraph(X_2) which guarantees $H(X_2, s, s') = 1$.
   If the solution of size s and sink size s' in digraph$(X_1 \times X_2)$ contains only vertices of digraph(X_1) but not all sinks of digraph(X_1), then by Lemma 5.3.10 there is a solution in digraph(X_1) which guarantees $H(X_1, s, s') = 1$.

   If the solution A' of size s and sink size s' in digraph$(X_1 \times X_2)$ contains all sinks of digraph(X_1), the series composition and the weak digraph constraint (5.3) imply that the set A' has to be extended by all sources of digraph(X_2). After ignoring the sources of digraph(X_2) (because the graph is acyclic), there must exist new sources, which have to be contained in A', since all their predecessors were sources in the original graph and so on. Thus, set A' contains all vertices of X_2 and by Lemma 5.3.10 there is a solution in digraph(X_1) which guarantees $H(X_1, s - s(X_2), i(X_1)) = 1$.

   If the solution A' of size s and sink size s' in digraph$(X_1 \times X_2)$ contains vertices of digraph(X_1) but not all sinks of digraph(X_1) and vertices of digraph(X_2), then by Lemma 5.3.10 there are s_1, s'_1 and s_2, s'_2 such that $s_1 + s_2 = s$, $s'_2 = s'$ and solutions in digraph(X_1) and in digraph(X_2) which guarantee $H(X_1, s_1, s'_1) = 1$ and $H(X_2, s_2, s'_2) = 1$.

   Further, the solutions of size $0 \leq s \leq s(X_2)$ from digraph(X_2) remain feasible in digraph$(X_1 \times X_2)$ since the solutions from digraph(X_2) do not contain any predecessors of vertices from digraph(X_1) in digraph$(X_1 \times X_2)$.

   The solutions from digraph(X_1) which do not contain all sinks of X_1, i.e., $1 \leq s'_1 < i(X_1)$ remain feasible in digraph$(X_1 \times X_2)$, but the sizes of sinks have to be changed to 0 since these sinks are no longer sinks in the digraph$(X_1 \times X_2)$.

   Next we consider solutions A' from digraph(X_1) which contain all sinks of digraph(X_1), i.e., $s' = i(X_1)$. As mentioned above, the series composition and the weak digraph constraint (5.3) imply that the set A' has to be extended by all vertices of X_2. The sizes of sinks have to be changed to $i(X_2)$, since all sinks of X_2 are also sinks in the digraph$(X_1 \times X_2)$.

Further, we can combine solutions of size $1 \leq s_1 \leq s(X_1)$ from digraph(X_1), which do not contain all sinks of X_1, i.e., $1 \leq s'_1 < i(X_1)$, and solutions of size $1 \leq s_2 \leq s(X_2)$ from digraph(X_2), to a solution of size $s_1 + s_2$ and sizes of sinks s'_2 in digraph$(X_1 \times X_2)$.

□

Corollary 5.5.9. *There is a solution with sum s for some instance of SSGW such that G is a minimal series-parallel digraph which is given by some msp-expression X if and only if* $H(X,s,s') = 1$. *Therefore,* $OPT(I) = \max\{s \mid H(X,s,s') = 1\}$.

Theorem 5.5.10. *SSGW can be solved in minimal series-parallel digraphs with n vertices and m arcs in* $O(n \cdot c^4 + m)$ *time and* $O(n \cdot c^2)$ *space.*

Proof. Let $G = (V,E)$ be a minimal series-parallel digraph and T be an msp-tree for G with root r. For some vertex u of T we denote by T_u the subtree rooted at u and X_u the msp-expression defined by T_u. In order to solve the SSGW problem for an instance I graph G, we traverse msp-tree T into bottom-up order. For every vertex u of T and $0 \leq s,s' \leq c$ we compute $H(X_u,s,s')$ following the rules given in Lemma 5.5.8. By Corollary 5.5.9 we can solve our problem by $H(X_r,s,s') = H(X,s,s')$.

An msp-tree T can be computed in $O(n+m)$ time from an msp-digraph with n vertices and m arcs, see Observation 3.6.3. All $s(X_i)$ and all $i(X_i)$ can be precomputed in $O(n)$ time. Our rules given in Lemma 5.5.8 show the following running times.

- For every $a_j \in A$ and every $0 \leq s,s' \leq c$ value $H(a_j,s,s')$ is computable in $O(1)$ time.
- For every $0 \leq s,s' \leq c$, every $H(X_1 \cup X_2,s,s')$ can be computed in $O(c^2)$ time from $H(X_1,s_1,s'_1)$ and $H(X_2,s_2,s'_2)$.
- For every $0 \leq s,s' \leq c$, every $H(X_1 \times X_2,s,s')$ can be computed in $O(c^2)$ time from $H(X_1,s_1,s'_1)$, $H(X_2,s_2,s'_2)$, and $i(X_1)$.

Since we have n leaves and $n-1$ inner vertices in T, the running time is in $O(nc^4 + m)$. □

5.6 Conclusions

The presented methods allow us to solve SSG and SSGW with digraph constraints given by directed co-graphs and (minimal) series-parallel digraphs in pseudo-polynomial time.

In contrast to [GMT18], we do not consider null sizes in the method used. This allows to verify whether a solution consists of all vertices or contains all sinks of a subgraph by using the sum of the sizes of the corresponding items. SSG and SSGW using null sizes can also be solved in pseudo-polynomial time on directed co-graphs and (minimal) series-parallel digraphs by additional counting the number of vertices or sinks within a SSGW solution.

For future work it could be interesting to find a solution for SSGW for series-parallel digraphs in general. Example 5.3.7 shows that Lemma 5.3.6 and the recursive structure of minimal series-parallel digraphs cannot be used in this case.

It remains to analyze whether the shown results also hold for other graph classes. Therefore one could consider edge series-parallel digraphs, see Subsection 3.6.5. Further, it remains to look at more general graph classes, such as graphs of bounded directed clique-width, as e.g. twin-dh digraph, which have directed clique-width at most three and which are a superclass of directed co-graphs. Since in the directed case bounded directed tree-width does not imply bounded directed clique-width, solutions for subset sum problems with digraph constraints of bounded directed tree-width are interesting as well.

Moreover, one could take a look at related problems. These include the two minimization problems which are introduced in [GMT18] by adding a maximality constraint to SSG and SSGW. Additionally, a generalization of the results for SSG to the partially ordered knapsack problem [JN83, KP04] or the one-neighbor and the all-neighbors knapsack problem[6] seems promising. In the following, we provide a brief look at what research has recently developed from these findings.

5.6.1 An Outlook to the Knapsack Problem with Special Neighbor Constraints

The *knapsack problem (KP)* is about a given set $A = \{a_1, \dots, a_n\}$ of $n \geq 1$ items, where every item a_j has a size s_j and a profit p_j, and a given capacity c. We further assume that all values are non-negative integers and $s_j \leq c$ for every $j \in \{1, \dots, n\}$. The aim is to choose a subset A' of A, such that $p(A') := \sum_{a_j \in A'} p_j$ is maximized while the *capacity constraint* holds, which means that the sum of the sizes of the items in A' does not exceed the given capacity c. The previously discussed subset sum problem (SSP) is a special case of the knapsack problem for which we have $p_j = s_j$.

As in SSG and SSGW we can also define knapsack problems with special neighbor constraints, see [BHW11, BHW12]. Within the *one-neighbor knapsack problem*, we can put an item into subset A' only if at least one of its neighbors is in A'. Meanwhile within the *all-neighbors knapsack problem*, we can put an item to A' only if all its neighbors are in A'. For both problems, there are variants with *uniform* and *general* profits and weights. In [GGK22] there are given upper bounds for the time complexity of computing the different variants of the problems where the constraints are given by graphs of special digraph classes, namely on directed co-graphs, minimal series-parallel digraphs, and directed trees.

In [GGK22] there is given the following overview about the results of the various problems on the mentioned digraph classes.

	graph	one-neighbor		all-neighbors	
uniform	undirected	linear	[BHW12]	$O(n \cdot c) \subseteq O(n^2)$	[BHW12]
	directed	strongly NP-hard	[BHW12]	strongly NP-hard	[BHW12]
general	undirected	APX-hard	[BHW12]	PFTAS	[BHW12]
				$O(n \cdot P + n^2)$	[GGK22]
	directed	strongly NP-hard	[BHW12]	strongly NP-hard	[BHW12]

Table 5.2: Time complexity of knapsack problems with neighbor constraints

[6]The general, directed, all-neighbors knapsack problem is closely related to the partially ordered knapsack problem

	graph	one-neighbor		all-neighbors	
uniform	undirected	linear	[BHW12]	$O(1)$	one component
	directed	$O(n^3)$	[GGK22]	$O\left(n^3\right)$	[GGK22]
general	undirected	NP-hard	[GGK22]	$O(n)$	[GGK22]
	directed	NP-hard	[GGK22]	NP-hard	[GGK22]
		$O(n \cdot P^2 + n)$	[GGK22]	$O(n \cdot (P+1) \cdot (P+n))$	[GGK22]

Table 5.3: Time complexity of knapsack problems with neighbor constraints given by trees

	graph	one-neighbor		all-neighbors	
uniform	undirected	linear	[BHW12]	$O(n \cdot c) \subseteq O(n^2)$	[BHW12]
	directed	$O(n^3)$	[GGK22]	$O(n^3)$	[GGK22]
general	undirected	$O(n \cdot P^2 + n^2)$	[GGK22]	$O(n \cdot P + n^2)$	[GGK22]
	directed	$O(n \cdot P^2 + n^2)$	[GGK22]	$O(n \cdot (P+1) \cdot \max\{n, P+1\})$	[GGK22]

Table 5.4: Time complexity of knapsack problems with neighbor constraints given by co-graphs

	one-neighbor		all-neighbors	
uniform	$O(n^3)$	[GGK22]	$O(n^3)$	[GGK22]
general	$O(n \cdot P^2 + n^2)$	[GGK22]	$O(n \cdot (P+1) \cdot \max\{n, P+1\})$	[GGK22]

Table 5.5: Time complexity of knapsack problems with neighbor constraints given by msp-digraphs

6 NP-hard Problems on Various Recursive Digraph Classes: Computing Directed Steiner Path Covers

In this chapter we show the directed Steiner path cover problem on a special digraph class. The content is taken from [GKR^{+}22].

6.1 Introduction

In the Steiner tree problem on undirected graphs we have given a graph G with non-negative edge weights and a subset of so-called terminal vertices in G. We search for a subtree of minimum (edge) weight that contains all so-called terminal vertices and possibly additional non-terminal vertices. The Steiner path problem is a special restriction of the Steiner tree problem in which the required terminal vertices lie on one path with minimal cost. It can also be seen as a generalization of the Hamiltonian path problem, since if we choose each vertex as a terminal vertex and if we have uniform edge weights, the Steiner path problem is equal to the Hamiltonian path problem. In [AACKS14] they observed the Euclidean bottleneck Steiner path problem. Moreover, in [MJV13] there was given a linear time solution for the Steiner path problem on trees.

For the well known Steiner tree problem there are efficient algorithms on special graph classes like series-parallel graphs [WC83], outerplanar graphs [WC82] and graphs of bounded tree-width [BCKN15, CMZ12]. We consider the class Steiner tree problem (CSP), which is a generalization of the Steiner tree problem where the vertices are partitioned into classes of terminals [RW90]. The unit-weight version of CSP is linear time solvable on co-graphs [WY95].

A Steiner tree always exists within connected graphs. Nevertheless, it is not always possible to find a Steiner path, which motivates us to look at Steiner path cover problems. This Steiner path cover problem was already considered in [CL18] on interval graphs.

In the following we investigate the directed Steiner path cover problem for which we give the following definition. Let G be a digraph with vertex set $V(G)$ and edge set $E(G)$ and let $T \subseteq V(G)$ be a set of terminal vertices. Further, let $c : E(G) \to \mathbb{R}^{\geq 0}$ be a function that assigns a weight to each edge. A *directed Steiner path cover* for G is a set of vertex-disjoint simple directed paths P in G that consists of all terminal vertices T and possibly some additional non-terminal (Steiner) vertices of $V(G) - T$. We define the *size* of a directed Steiner path cover as the number of its paths, i.e., the size is $|P|$, while the *cost* is defined as the sum of weights of the edges used in the paths in a directed Steiner path cover of minimum size.

Name: Directed Steiner path cover problem
Instance: A digraph G, a set of terminal vertices $T \subseteq V(G)$, and edge weights $c : E(G) \to \mathbb{R}^{\geq 0}$.
Task: Find a directed Steiner path cover P of minimum size for G that minimizes $\sum_{p \in P} \sum_{e \in p} c(e)$.

Minimizing only the sum of the weights of the edges is not reasonable since this sum is minimized if we take each terminal vertex on its own as a path of length 0. So we primary require that the number of paths is minimal.

The directed Steiner path cover problem is NP-hard since it is a generalization of the directed Hamiltonian path problem.

Name: Directed Hamiltonian path problem
Instance: A digraph G.
Task: Find a directed Hamiltonian path in G.

This gives a motivation for a restriction of the problem to special graph classes. We observe digraphs of a very natural graph class, namely directed co-graphs.

For graphs where all edges have the same weight, the above definition of the directed Steiner path cover problem results in a minimum number of Steiner vertices. Graphs without edge weights can be considered as a special case of graphs with unit-edge weights. Since edge weights do not occur in co-graphs, we use the following problem definition.

Name: Unit-edge-weight directed Steiner path cover problem
Instance: A digraph G and a set of terminal vertices $T \subseteq V(G)$.
Task: Find a directed Steiner path cover of minimum size for G such that the number of Steiner vertices is minimal.

In the following we present how to compute the value of a directed Steiner path cover of minimum size and cost for the disjoint union, order composition and series composition of two directed co-graphs in linear time from the corresponding values of the involved directed co-graphs. For this purpose, we give a definition of a useful normal form for directed Steiner path covers in digraphs which are defined by the order composition or series composition of two directed co-graphs. Moreover, we conclude that a directed Steiner path cover of minimum size and cost for a directed co-graph can be computed in linear time.

Name: Directed Steiner path problem
Instance: A digraph G, a set $T \subseteq V(G)$ of terminal vertices, and a function $c : E(G) \to \mathbb{R}^{\geq 0}$ that assigns to each edge some weight.
Task: Find a directed Steiner path p in graph G that minimizes $\sum_{e \in p} c(e)$.

6.2 Normal form for Directed Steiner Path Covers

In the following we give a definition of a normal form for directed Steiner path covers in directed co-graphs which are defined by the order composition or series composition of two directed co-graphs.

Let G be a directed co-graph and let $T \subseteq V(G)$ be a set of terminal vertices. Further, let C be a directed Steiner path cover for G with respect to T, while $s(C)$ denotes the number of Steiner vertices in the paths of C.

Lemma 6.2.1. *Let C be a directed Steiner path cover for some directed co-graph $G = A \oslash B$ or $G = A \otimes B$ with respect to a set $T \subseteq V(G)$ of terminal vertices. Then, there is a directed Steiner path cover C' with respect to T that does not contain paths p and p' satisfying one of the structures (1)-(4), such that $|C| \geq |C'|$ and $s(C) \geq s(C')$ applies. Let $q_1, \ldots, q_4$ denote sub-paths which may be empty.*

1. *$p = (x, q_1)$ or $p = (q_1, x)$ where $x \notin T$. Comment: No path starts or ends with a Steiner vertex.*

2. *$p = (q_1, u, x, v, q_2)$ where $u \in V(A)$, $v \in V(B)$, and $x \notin T$. Comment: On a path, the neighbors u, v of a Steiner vertex x are both contained in the same digraph.*

3. *$p = (q_1, x)$, $p' = (u, q_2)$, where $x \in V(A)$, $u \in V(B)$, $p \neq p'$. Comment: No path p ends in A, if there is a path $p' \neq p$ that starts in B.*

4. *$p = (\ldots, x, u, v, y, \ldots)$ where $u, v \notin T$. Comment: The paths contain no edge between two Steiner vertices.*

If $G = A \otimes B$, then the cover C' also does not contain paths satisfying structures (5)-(8).

5. *$p = (x, q_1)$, $p' = (u, q_2)$, where $x \in V(A)$, $u \in V(B)$, $p \neq p'$. Comment: All paths start in the same digraph.*

6. *$p = (q_1, x, y, q_2)$, $p' = (q_3, u, v, q_4)$ where $x, y \in V(A)$, $u, v \in V(B)$. Comment: The cover C' contains edges of only one of the digraphs.*

7. *$p = (x, q_1)$, $p' = (q_2, u, y, v, q_3)$, where $x, y \in V(A)$, $u, v \in V(B)$, and $y \notin T$. Comment: If a path starts in A then there is no Steiner vertex in A with two neighbors on the path in B.*

8. *$p = (x, q_1)$, $p' = (q_2, u, v, q_3)$, where $x \in V(A)$ and $u, v \in V(B)$. Comment: If a path starts in A, then no edge of B is contained in the cover.*

Proof. 1. If x is removed from p we get a cover with one Steiner vertex less than C.

2. If x is removed from p, we get a cover with one Steiner vertex less than C.

3. We combine the paths to only one path (q_1, x, u, q_2) and we get a cover with one path less than C.

4. Since G is a directed co-graph, the underlying undirected graph is a co-graph such that the path cannot include a P_4, i.e., a simple path of 4 vertices, as induced subgraph. Thus, there must be at least one additional arc. If such an additional arc would shorten the Steiner path by skipping u, v or both then we remove u or v or both and take the shortcut for getting a cover C'. Additional arcs that do not shorten the path would create a forbidden induced subgraph from Figure 3.4 which is not possible. For details see Table 6.1.

5. The new paths are q_1 and $(x, u.q_2)$. The cover C' is as good as C.

6. If $p \neq p'$, then (q_1, x, v, q_4) and (q_3, u, y, q_2) are the paths in cover C', see Figure 6.1. If $p = p'$, then we have to distinguish whether $(u,v) \in q_1$, $(u,v) \in q_2$, $(x,y) \in q_3$, or

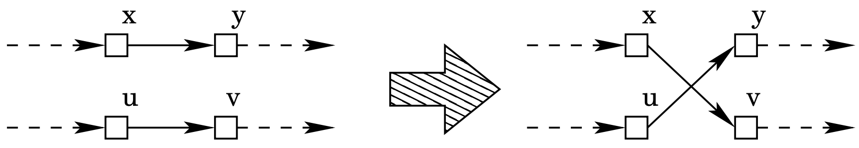


Figure 6.1: Substitution in the proof of Lemma 6.2.1(6.) for $p \neq p'$.

$(x,y) \in q_4$. We show how to handle the first case, the other three cases are similar. Let $p = (q_3, u, v, q_5, b, a, q_6, x, y, q_2)$, where $b \in V(B)$ and $a \in V(A)$. Then the new path in cover C' is $(q_3, u, a, q_6, x, v, q_5, b, y, q_2)$. Such vertices a and b must exist because $v \in V(B)$ and $x \in V(A)$, possibly it holds $a = x$ or $b = v$. In any case cover C' is as good as C, see Figure 6.2.

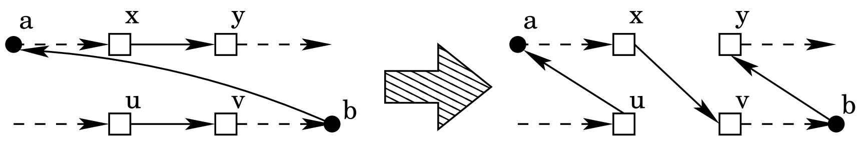


Figure 6.2: Substitution in the proof of Lemma 6.2.1(6.) for $p = p'$.

7. If $p \neq p'$, then q_1 and (q_2, u, x, v, q_3) are the new paths in cover C'. If $p = p'$, i.e., $q_1 = (q'_2, u, y, v, q_3)$, where q'_2 is obtained from q_2 by removing x, then (q'_2, u, x, v, q_3) is the new path in cover C'. The cover C' is as good as C, see Figure 6.3. If $p \neq p'$, then the edge (a,b) is missing in Figure 6.3.

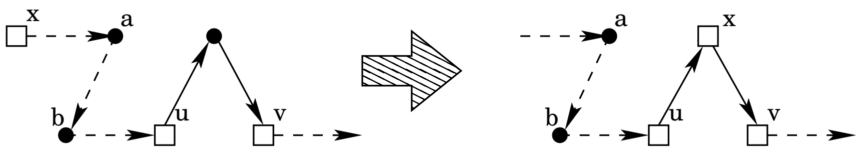


Figure 6.3: Substitution in the proof of Lemma 6.2.1(7.).

8. If $p \neq p'$, then q_1 and (q_2, u, x, v, q_3) are the new paths in cover C'. If $p = p'$, i.e., $q_1 = (q'_2, u, v, q_3)$, where q'_2 is obtained from q_2 by removing x, then (q'_2, u, x, v, q_3) is the new path in cover C'. The cover C' is as good as C, see Figure 6.4. If $p \neq p'$, then the edge (a, b) is missing in Figure 6.4.

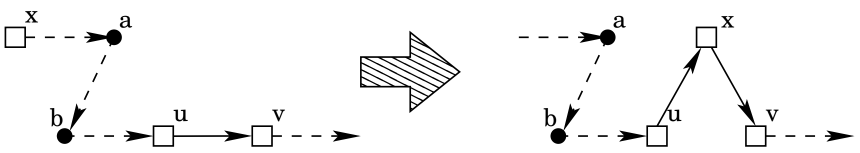


Figure 6.4: Substitution in the proof of Lemma 6.2.1(8.).

Operations 1, 2, 4 and 7 reduce the number of Steiner vertices by one, the remaining operations 3, 5 and 6 do not change the number of Steiner vertices. Therefore, operations 1, 2, 4 and 7 can only be executed at most $|V - (T_A \cup T_B)|$ times.

Operation 6 reduces the number of paths by one, the remaining operations do not increase the number of paths. Therefore operation 6 can be executed at most $\max\{|T_A|, |T_B|\}$ times.

Let us now consider those edges on a path that connect vertices of A and vertices of B. The maximum number of those edges is $|V(A)| + |V(B)| - 1$. Operation 7 can remove two such edges, operations 3 and 5 can add two such edges. Since the other operations 1, 2, 4 and 6 do not reduce the number of edges, operations 3 and 5 can be used at most $(|V(A)| + |V(B)| - 1)/2 + |V - (T_A \cup T_B)|$ times. □

The hypothesis of Lemma 6.2.1 is symmetric in A and B and thus, the statement of Lemma 6.2.1 is also valid for co-graphs $G = A \otimes B$ if we switch A and B.

Definition 6.2.2. A directed Steiner path cover C for some directed co-graph $G = A \oslash B$ or $G = A \otimes B$ is said to be in *normal form* if none of the operations described in the proof of Lemma 6.2.1 is applicable.

Now we assume that a directed Steiner path cover for some directed co-graph $G = A \oslash B$ or $G = A \otimes B$ is always in normal form, since the operations of the proof of Lemma 6.2.1 do not increase the number of paths or Steiner vertices of a cover. Lemma 6.2.1 implies the following theorem.

Theorem 6.2.3. *For each directed co-graph $G = A \otimes B$ and set of terminal vertices $T \subseteq V(G)$ any directed Steiner path cover C in normal form with respect to T does not contain an edge of digraph A, and no path in C starts or ends in digraph A if $|T_A| < |T_B|$.*

Proof. [*by contradiction*] Assume, the Steiner path cover C contains an edge of digraph A. Then by Lemma 6.2.1(5), all paths starts in digraph A. By Lemma 6.2.1(4), it holds that no Steiner vertex v of $V(A)$ is contained in C, where the neighbors of v are both of digraph B. By Lemma 6.2.1 (1), (2), and (5), it holds that all vertices of $V(B)$ from C are connected with a terminal vertex of $V(A)$, thus $|T_A| > |T_B|$.↯

Second, we have to show that no path in C starts or ends in digraph A. Assume on the contrary, that there is one path that starts in A. By Lemma 6.2.1(6), it holds that all paths start in A. Continuing as in the first case this leads to a contradiction. □

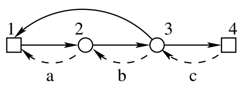
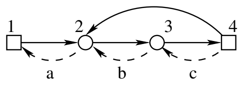
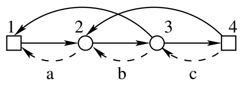
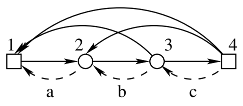
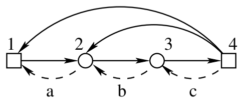
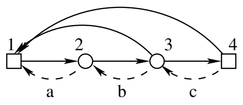
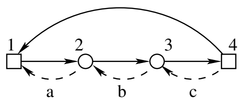
graph with underlying P_4 and additional edges that do not shorten the path	none	a	b	c
	a, b	a, c	b, c	a, b, c
	D_5 $\{1,2,3\}$	D_4 $\{1,2,3\}$	D_4 $\{1,2,3\}$	D_3 $\{1,3,4\}$
	D_3 $\{2,3,4\}$	D_4 $\{1,2,3\}$	D_4 $\{1,2,3\}$	D_3 $\{1,3,4\}$
	D_5 $\{2,3,4\}$	D_2 $\{1,2,4\}$	D_2 $\{1,2,3\}$	D_4 $\{2,3,4\}$
	D_2 $\{1,2,4\}$	D_4 $\{2,3,4\}$	D_2 $\{1,2,3\}$	D_2 $\{1,2,4\}$
	D_5 $\{2,3,4\}$	D_4 $\{1,2,3\}$	D_4 $\{1,2,3\}$	D_4 $\{2,3,4\}$
	D_4 $\{2,3,4\}$	D_4 $\{2,3,4\}$	D_4 $\{1,2,3\}$	D_2 $\{1,2,4\}$
	D_5 $\{1,2,3\}$	D_4 $\{1,2,3\}$	D_4 $\{1,2,3\}$	D_4 $\{2,3,4\}$
	D_4 $\{2,3,4\}$	D_4 $\{2,3,4\}$	D_4 $\{1,2,3\}$	D_6 $\{1,2,3,4\}$
	D_5 $\{2,3,4\}$	D_3 $\{1,2,3\}$	D_2 $\{1,2,3\}$	D_1 $\{1,2,3\}$
	D_1 $\{1,3,4\}$	D_3 $\{1,3,4\}$	D_3 $\{1,3,4\}$	D_3 $\{1,3,4\}$
	D_5 $\{1,2,3\}$	D_1 $\{2,3,4\}$	D_1 $\{1,2,4\}$	D_1 $\{1,2,4\}$
	D_2 $\{1,2,4\}$	D_2 $\{1,2,4\}$	D_1 $\{1,2,4\}$	D_2 $\{1,2,4\}$
	D_1 $\{1,2,3\}$	D_1 $\{2,3,4\}$	D_1 $\{1,2,4\}$	D_1 $\{1,2,3\}$
	D_3 $\{2,3,4\}$	D_2 $\{1,2,4\}$	D_2 $\{1,2,3\}$	D_3 $\{1,3,4\}$

Table 6.1: The leftmost column shows a graph with underlying undirected P_4 and at least one additional arc that do not shorten the path. The other columns shows the forbidden subgraphs that are contained in the leftmost graph depending on the edges of the P_4.

Remark 6.2.4. For each directed co-graph $G = A \oslash B$ and set of terminal vertices $T \subseteq V(G)$ any directed Steiner path cover C in normal form with respect to T it holds that each path that starts in A either remains in A or it crosses over to B and remains in B. Each path that reaches a vertex of B has to stay in B since no edge from a vertex in B to a vertex in A exists.

6.3 Computing the Optimal Number of Paths

Let G be a directed co-graph and $T \subseteq V(G)$ be a set of terminal vertices. We define $p(G,T)$ as the minimum number of paths within a Steiner path cover for G with respect to T. Further, let $s(G,T)$ be the minimum number of Steiner vertices in a directed Steiner path cover of size $p(G,T)$ with respect to T. If it is clear from the context, we do not specify set T. To name a single vertex by v, we use $\bullet_v$ here instead of just $\bullet$, as in the actual definition of co-graphs.

Lemma 6.3.1. *Let A and B be two vertex-disjoint directed co-graphs and let $T_A \subseteq V(A)$ and $T_B \subseteq V(B)$ be two sets of terminal vertices. The following equations apply.*

1. $p(\bullet_v, \emptyset) = 0$ *and* $p(\bullet_v, \{v\}) = 1$
2. $p(A \oplus B, T_A \cup T_B) = p(A,T_A) + p(B,T_B)$
3. $p(A \otimes B, \emptyset) = 0$
4. $p(A \otimes B, T_A \cup T_B) = \max\{1, p(B,T_B) - |V(A)|\}$ *if* $1 \leq |T_B|$ *and* $|T_A| \leq |T_B|$
5. $p(A \otimes B, T_A \cup T_B) = \max\{1, p(A,T_A) - |V(B)|\}$ *if* $1 \leq |T_A|$ *and* $|T_A| > |T_B|$
6. $p(A \oslash B, T_A \cup T_B) = p(A,T_A)$ *if* $p(A) \geq p(B)$
7. $p(A \oslash B, T_A \cup T_B) = p(B,T_B)$ *if* $p(A) < p(B)$

Proof. 1. - 3. Obviously holds.

4. We show $p(A \otimes B) \geq \max\{1, p(B) - |V(A)|\}$ by an indirect proof. We assume that in a directed Steiner path cover C for $A \otimes B$ there are less than $\max\{1, p(B) - |V(A)|\}$ paths. Removing all vertices of A from all paths in C leads to a directed Steiner path cover of size $|C| + |V(A)| < p(B)$ for B. ↯We now show that $p(A \otimes B) \leq \max\{1, p(B) - |V(A)|\}$ applies. We can use any vertex of A to combine two paths of the cover of B to one path, as the series composition of A and B creates every possible directed edge between A and B. If there exists more terminal vertices in T_A than there are paths in the cover of B, i.e., $p(B) < |T_A|$, we split paths of B and reconnect them with terminal vertices of T_A, which is always possible since $|T_A| \leq |T_B|$.

5. Similar to 4.

6. We show that $p(A \oslash B) \leq p(A)$ holds. Consider that it is possible to append any path of A by any path of B, see Lemma 6.2.1(3). As this creates no edge between B and A, we cannot extend a path of B with a path of A.

We show by an indirect proof that $p(A \oslash B) \geq p(A)$ applies. We assume that a directed Steiner path cover C for $A \oslash B$ contains less than $p(A)$ paths. By removing all vertices of B from all paths in C we get a Steiner path cover of size $|C| < p(A)$. ↯

7. Similar to 6.

□

6.4 Computing the Optimal Number of Steiner Vertices

Remark 6.4.1. For two vertex-disjoint directed co-graphs A, B and two sets of terminal vertices $T_A \subseteq V(A)$, $T_B \subseteq V(B)$ it holds that $s(A \oplus B, T_A \cup T_B) = s(A, T_A) + s(B, T_B)$, as the disjoint union does not create new edges.

Remark 6.4.2. Let $G = A \oslash B$ be a directed co-graph, and let C be a directed Steiner path cover of G such that $p = (q_1, u_1, x, q_2, v_1)$ is a path in A, $p_1 = (u_2, q_3)$ and $p_2 = (v_2, q_4)$ are paths in B and all paths are vertex-disjoint paths in C, where $x \notin T$, $u_1, u_2, v_1, v_2 \in T$, and $q_1, \ldots, q_4$ are sub-paths. Now, we split p at vertex x into two paths and combine them with p_1 and p_2 to obtain (q_1, u_1, u_2, q_3) and (q_2, v_1, v_2, q_4) as new paths. On this way we get a Steiner path cover without increasing the cost. If we switch A and B, we obtain (u_2, q_3, q_1, u_1) and (v_2, q_4, q_2, v_1) as new paths such that the statement holds as well.

We conclude the central lemma of our work. The proof is done by induction on the structure of the co-graph.

Lemma 6.4.3. *Let G be a directed co-graph and C a directed Steiner path cover for G with respect to a set $T \subseteq V(G)$ of terminal vertices. Then, it holds that $p(G) + s(G) \leq |C| + s(C)$.*

Proof. We proof this by induction. Obviously, the statement is valid for directed co-graphs with only one vertex. We now assume that the statement is valid for directed co-graphs of n vertices. Let A and B are vertex-disjoint directed co-graphs with at most n vertices each.
Disjoint union: Let $G = A \oplus B$ be a directed co-graph with more than n vertices. By Lemma 6.3.1, and Remark 6.4.1, we know that $p(A \oplus B) + s(A \oplus B) = p(A) + p(B) + s(A) + s(B)$. By the induction hypothesis, $p(A) + s(A) \leq |C_{|A}| + s(C_{|A})$ and $p(B) + s(B) \leq |C_{|B}| + s(C_{|B})$ apply, where $C_{|A}$ denotes the cover C restricted to digraph A, i.e., the cover which results if we remove all vertices of B from C. This yields to the statement of the lemma.

$$p(A \oplus B) + s(A \oplus B) \leq |C_{|A}| + s(C_{|A}) + |C_{|B}| + s(C_{|B}) = |C| + s(C)$$

Series composition: Let $G = A \otimes B$ be a directed co-graph on more than n vertices and without loss of generality it holds that $|T_A| \leq |T_B|$.

1. Let $X(A)$ denote the vertices of A used in cover C, and let D be the cover for B that results by removing the vertices of $X(A)$ from the cover C. By induction hypothesis, we get $p(B) + s(B) \leq |D| + s(D)$.

2. Let $nt(X(A))$ be the number of non-terminal vertices of $X(A)$. By Theorem 6.2.3 we get that $s(C) = s(D) + nt(X(A))$ and $|C| = |D| - |T_A| - nt(X(A))$. Consequently, we come to $|C| + s(C) = |D| + s(D) - |T_A|$.

Combining these results together we get:

$$p(B) + s(B) - |T_A| \leq |D| + s(D) - |T_A| = |C| + s(C)$$

To prove the statement of the lemma, we start with considering the case $p(B) - 1 \leq |V(A)|$. Then, we get $p(A \otimes B) = 1$. If $|T_A| \geq p(B) - 1$, then $d := |T_A| - (p(B) - 1)$ many Steiner vertices from B, are replaced by terminal vertices from A, if they are available. Otherwise, if $|T_A| < p(B) - 1$, we get $-d = (p(B) - 1) - |T_A|$ many Steiner vertices from A with which we combine the paths. Consequently, it holds that $s(A \otimes B) \leq \max\{0, s(B) - d\}$ as the number of Steiner vertices in an optimal cover is at most the number of Steiner vertices in a certain cover. Thus, since $p(A \otimes B) = 1$ we obtain for $s(B) \geq d$:

$$\begin{aligned} p(A \otimes B) + s(A \otimes B) &\leq 1 + s(B) - d = 1 + s(B) - (|T_A| - (p(B) - 1)) \\ &= \cancel{1} + s(B) - |T_A| + p(B) - \cancel{1} \leq |C| + s(C) \end{aligned}$$

If $s(B) < d$, then all Steiner vertices of B are replaced by terminal vertices of A and as we have $|T_A| \leq |T_B|$, some of the paths of B are reconnected by the remaining terminal vertices of A. Consequently, we get $p(A \otimes B) + s(A \otimes B) = 1 \leq |C| + s(C)$.
Now, we come to the case where $p(B) - 1 > |V(A)|$, i.e., it is not possible to combine all paths in an optimal cover for B with vertices of A. By Lemma 6.3.1, we know that $p(A \otimes B) = \max\{1, p(B) - |V(A)|\}$. Consequently, for $p(A \otimes B) > 1$ it follows:

$$\begin{aligned} p(A \otimes B) + s(A \otimes B) &\leq p(B) - |V(A)| + s(B) + nt(A) \\ &= p(B) + s(B) - |T_A| \leq |C| + s(C) \end{aligned}$$

Since the non-terminal vertices of A are now used to combine paths of the cover, the non-terminal vertices of A become Steiner vertices.
Order composition: Let $G = A \oslash B$ be a directed co-graph with more than n vertices. By the induction hypothesis, we get $p(A) + s(A) \leq |C_{|A}| + s(C_{|A})$ and $p(B) + s(B) \leq |C_{|B}| + s(C_{|B})$. Let us first consider the case $p(A) > p(B)$. By Lemma 6.3.1 it holds $p(A \oslash B) = p(A)$. We can append any path of A by any path of B, and by Remark 6.4.2 it holds that for every path that there is more in A than in B, a Steiner vertex of B can be removed. Additionally, as an optimal cover has at most as many Steiner vertices as a concrete cover, we get $s(A \oslash B) \leq s(C_{|A}) + s(C_{|B}) - \min\{s(C_{|B}), |C_{|A}| - |C_{|B}|\}$. If we sum up both equations we come to

$$p(A \oslash B) + s(A \oslash B) \leq p(A) + s(C_{|A}) + s(C_{|B}) - \min\{s(C_{|B}), |C_{|A}| - |C_{|B}|\}$$

If $s(C_{|B}) \geq |C_{|A}| - |C_{|B}|$ applies, and as $s(C) = s(C_{|A}) + s(C_{|B})$ applies, we get

$$p(A \oslash B) + s(A \oslash B) \leq p(A) + s(C) - |C_{|A}| + |C_{|B}|.$$

The statement would be shown if $p(A) - |C_{|A}| + |C_{|B}| \leq |C|$ applied. It holds $p(A) \leq |C_{|A}|$ since an optimal cover has at most as many paths as a concrete cover, and it holds $|C_{|B}| \leq |C|$, since $|C| = \max\{|C_{|A}|, |C_{|B}|\}$ by Remark 6.2.4. We sum up these equations which leads to $p(A) + |C_{|B}| \leq |C_{|A}| + |C|$. That is equivalent to $p(A) - |C_{|A}| + |C_{|B}| \leq |C|$, such that $p(A \oslash B) + s(A \oslash B) \leq |C| + s(C)$ has been shown.

If $s(C_{|B}) < |C_{|A}| - |C_{|B}|$, then we get $p(A \oslash B) + s(A \oslash B) \leq p(A) + s(C_{|A})$. It is left to show that $p(A) + s(C_{|A}) \leq |C| + s(C)$ applies. Since an optimal cover has at most as many paths as a concrete cover, it holds $p(A) \leq |C_{|A}|$. Further, we know that $|C_{|A}| \leq |C|$ since $|C| = \max\{|C_{|A}|, |C_{|B}|\}$ by Remark 6.2.4. Moreover, it holds $s(C_{|A}) \leq s(C)$, as a part can at most be as big as the whole.
We can show the other case $p(A) \leq p(B)$ on a similar way. □

To see why Lemma 6.4.3 is crucial for the rest of this work, consider the digraph B of Figure 6.5 that is not a directed co-graph. Terminal vertices $T_A = \{f, g\}$ and $T_B = \{a, c, e, u, w, x\}$ are shown as squares. In the left part of the figure a Steiner path cover $C_\ell = \{(a, b, c, d, e), (u, v, w, x, y)\}$ for graph B is shown with $|C_\ell| = 2$ and $s(C_\ell) = 4$ which is optimal. In the right part of the figure a Steiner path cover $C_r = \{(a, b, c, w, x, y), (e), (u)\}$ for B is shown with $|C_r| = 3$ and $s(C_r) = 2$.

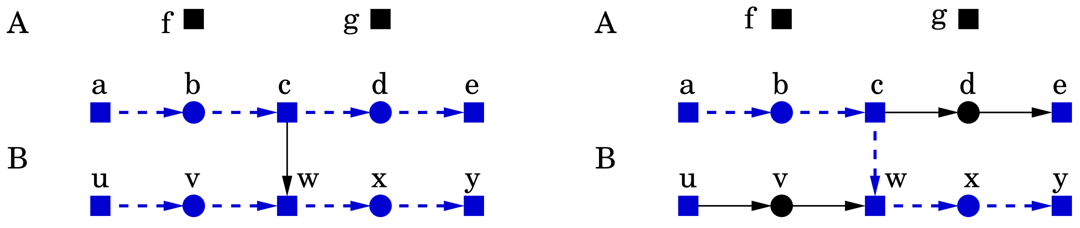


Figure 6.5: Small example that shows the contra positive of the statement of Lemma 6.4.3 in a graph B that is not directed co-graph.

The right cover can be extended to an optimal cover for $A \otimes B$ if the vertices of A are used to combine the path: $\{(u, f, a, b, c, w, x, y, g, e)\}$ is an optimal cover for $A \otimes B$ with only one path and 2 Steiner vertices. The left Steiner path cover can not be extended to an optimal cover for $A \otimes B$. For example, we can split path (a, b, c, d, e) at vertex b into two paths (a) and (c, d, e) and reconnect them by a vertex of A and get (a, f, c, d, e). The other vertex of A must be used to combine the remaining two paths to $(a, f, c, d, e, g, u, v, w, x, y)$ which results in a cover for $A \otimes B$ that consists of one path but 3 Steiner vertices. For graph B the statement of Lemma 6.4.3 is not satisfied: $p(B) + s(B) = 2 + 4 = 6 > |C_r| + s(C_r) = 3 + 2 = 5$
In the proof of Lemma 6.4.5 we use the statement of Lemma 6.4.3 to show that optimal solutions for directed co-graphs A and B can be combined to an optimal solution for $A \oslash B$ and $A \otimes B$.

Remark 6.4.4. Let G be a directed co-graph and let C be a directed Steiner path cover for G with respect to some set of terminal vertices $T \subseteq V(G)$. Then $s(C) \geq s(G)$ holds only if $|C| = p(G)$. If $|C| > p(G)$ then $s(C)$ might be smaller than $s(G)$.

Lemma 6.4.5. *Let A and B be two vertex-disjoint digraphs, and let $T_A \subseteq V(A)$, $T_B \subseteq V(A)$ be sets of terminal vertices. Then the following equations applies:*

1. $s(\bullet_v, \emptyset) = 0$ *and* $s(\bullet_v, \{v\}) = 0$
2. $s(A \oplus B, T_A \cup T_B) = s(A, T_A) + s(B, T_B)$

3. *$s(A \otimes B) = \max\{0, s(B) + p(B) - p(A \otimes B) - |T_A|\}$ if $|T_A| \leq |T_B|$*

4. *$s(A \otimes B) = \max\{0, s(A) + p(A) - p(A \otimes B) - |T_B|\}$ if $|T_A| > |T_B|$*

5. *$s(A \oslash B) = s(A) + s(B)$ if $p(A) = p(B)$*

6. *$s(A \oslash B) = s(A) + s(B) - \min\{s(A), p(B) - p(A)\}$ if $p(A) < p(B)$*

7. *$s(A \oslash B) = s(A) + s(B) - \min\{s(B), p(A) - p(B)\}$ if $p(A) > p(B)$*

Proof. 1. Obvious.

2. See Remark 6.4.1

3. At first we show $s(A \otimes B) \leq \max\{0, s(B) + p(B) - p(A \otimes B) - |T_A|\}$.
Lemma 6.4.3 implies that $s(A \otimes B) + p(A \otimes B) \leq s(C) + |C|$ holds for any cover C for directed co-graph $A \otimes B$ and any set of terminal vertices T. We consider the cover C for $A \otimes B$ which is obtained by an optimal cover D for B in the following way: Take the terminal vertices of A to either combine paths of D or to remove a Steiner vertex of D by replacing $v \notin T$ by a terminal vertex of A in a path as $(\dots, u, v, w, \dots) \in D$, where $u, w \in T$. If $|T_A| \geq s(B) + p(B)$, then we can combine all paths of D while all Steiner vertices can be removed by terminal vertices of A. Since $|T_A| \leq |T_B|$ holds, we can split some of the paths and reconnected them by the remaining terminal vertices of A. Consequently, we get $s(C) + |C| = 1$ and $s(A \otimes B) = 0$.
Otherwise, if $|T_A| < s(B) + p(B)$, then we have $s(C) + |C| = s(B) + p(B) - |T_A|$, and by Lemma 6.4.3, the statement follows.

$$\begin{array}{rrcl} & s(A \otimes B) + p(A \otimes B) & \leq & s(B) + p(B) - |T_A| \ = \ s(C) + |C| \\ \iff & s(A \otimes B) & \leq & s(B) + p(B) - p(A \otimes B) - |T_A| \end{array}$$

We prove now that $s(A \otimes B) \geq \max\{0, s(B) + p(B) - p(A \otimes B) - |T_A|\}$.
By $X(A)$ we denote the vertices of $V(A)$ that are part of the paths in an optimal cover C for $A \otimes B$. Let D be the cover for B which we get by removing the vertices of $X(A)$ from C. Then, by Theorem 6.2.3 we come to the following:

$$\begin{array}{rrcl} & |X(A)| = nt(X(A)) + |T_A| & = & |D| - p(A \otimes B) \\ \iff & nt(X(A)) & = & |D| - p(A \otimes B) - |T_A| \end{array}$$

Thus, we get:

$$\begin{array}{rcl} s(A \otimes B) - nt(X(A)) \ = \ s(D) & = & s(A \otimes B) - |D| + p(A \otimes B) + |T_A| \\ \iff \quad s(A \otimes B) & = & s(D) + |D| - p(A \otimes B) - |T_A| \\ \Rightarrow \quad s(A \otimes B) & \geq & s(B) + p(B) - p(A \otimes B) - |T_A| \end{array}$$

The implication follows, as by Lemma 6.4.3 it holds $s(D) + |D| \geq s(B) + p(B)$.

4. Can be shown similar to the previous item.

5. To show that $s(A \oslash B) \leq s(A) + s(B)$ holds, we consider optimal covers C and D for A and B. We then build a cover E for $A \oslash B$ in such a way that any path of C is appended by a path of D, see Lemma 6.2.1(3). Since $|E| = p(A \oslash B)$ applies, $s(A \oslash B) \leq s(E) = s(C) + s(D) = s(A) + s(B)$ follows. That is because an optimal cover has at most as many Steiner vertices as a concrete cover.
   To show that $s(A \oslash B) \geq s(A) + s(B)$ holds, we consider an optimal cover C for $A \oslash B$. Then, we have $s(A \oslash B) = s(C_{|A}) + s(C_{|B}) \geq s(A) + s(B)$, as $|C_{|A}| = p(A) = p(A \oslash B) = p(B) = |C_{|B}|$.

6. We distinguish two cases. At first, let $s(A) > p(B) - p(A)$.
   To show that $s(A \oslash B) \leq s(A) + s(B) - (p(B) - p(A))$ holds, we consider optimal covers C and D for A and B. We construct a cover E for $A \oslash B$ as follows. We first split $p(B) - p(A)$ many paths of C at Steiner vertices as stated in Remark 6.4.2. After, we combine each of the resulting paths with a path of D. On this way, it holds that $|E| = p(A \oslash B) = p(B)$ and therefore, $s(A \oslash B) \leq s(C) + s(D) - (p(B) - p(A)) = s(A) + s(B) - (p(B) - p(A))$.
   Please note, a Steiner path cover C for $A \oslash B$ with $s(C_{|A}) > 0$ is not optimal if $|C_{|A}| < |C| = p(A \oslash B)$ holds. By Remark 6.4.2 a path of $C_{|A}$ could be split up at a Steiner vertex and the number of Steiner vertices could be reduced.
   To show $s(A \oslash B) \geq s(A) + s(B) - (p(B) - p(A))$, we look at an optimal cover C for $A \oslash B$. Thus, we have $s(A \oslash B) = s(C) = s(C_{|A}) + s(C_{|B})$. Further, by the previous note it holds $|C| = p(A \oslash B) = p(B) = |C_{|A}|$. By Lemma 6.4.3 we come to $s(C_{|A}) + |C_{|A}| \geq s(A) + p(A)$. Summing up these equations, we get $s(A \oslash B) + p(A \oslash B) = s(C_{|A}) + |C_{|A}| + s(C_{|B})$. Finally, we come to:

$$\begin{aligned} s(A \oslash B) &= s(C_{|A}) + |C_{|A}| - p(A \oslash B) + s(C_{|B}) \\ &\geq s(A) + p(A) - p(B) + s(C_{|B}) \;\geq\; s(A) + p(A) - p(B) + s(B) \end{aligned}$$

   The last step holds since $p(B) = |C_{|B}|$ and by Remark 6.4.4.
   Now we consider the case in which $s(A) \leq p(B) - p(A)$. To show that $s(A \oslash B) \leq s(B)$ applies, we consider optimal covers C and D for A and B. We then build a cover E for $A \oslash B$ such that we first split as many paths of C at Steiner vertices as possible in a way described in Remark 6.4.2. After this, all Steiner vertices of C have been removed and we combine each of the resulting paths with a path of D. Consequently, it we get $|E| = p(A \oslash B) = p(B)$ and thus, $s(A \oslash B) \leq s(E) = s(B)$.
   To show that $s(A \oslash B) \geq s(B)$ applies, we consider an optimal cover C for $A \oslash B$. By the above note we know that $s(C_{|A}) = 0$, otherwise, C would not be optimal. Then, by $|C_{|B}| = p(B)$ and by Remark 6.4.4 it follows $s(A \oslash B) = s(C_{|B}) \geq s(B)$.

7. Can be shown similar to the previous item.

□

A directed co-tree can be computed in linear time from the input directed co-graph, see Theorem 3.3.4. Combining this with Lemma 6.3.1 and 6.4.5, we come to the following result.

Theorem 6.4.6. *The value of a directed Steiner path cover of minimum cost for a directed co-graph can be computed in linear time with respect to the size of the directed co-expression.*

Lemma 6.4.3 allows us to minimize the following additional cost function.

Corollary 6.4.7. *The value of a directed Steiner path cover C for a directed co-graph G such that $|C| + s(C)$ is minimal can be computed in linear time with respect to the size of the directed co-expression.*

6.5 Computing an Optimal Directed Steiner Path Cover

A detailed exact algorithm to compute an optimal directed Steiner path cover for a directed co-graph can be read in [GKR^{+}22]. By algorithm DIRECTEDSTEINERPATHCOVER from [GKR^{+}22] and the results presented in this chapter we obtain the following result.

Theorem 6.5.1 ([GKR^{+}22])**.** *A directed Steiner path cover of minimum cost for a directed co-graph can be computed in linear time with respect to the size of the directed co-expression.*

6.6 Conclusion

To sum up we could show that there is a linear time solution for computing the minimum number of paths within a directed Steiner path cover and the minimum number of Steiner vertices in such a directed Steiner path cover in directed co-graphs. Thus, we conclude with a linear time computation of an optimal directed Steiner path, if it exists, for directed co-graphs.

For an undirected co-graph G, we can solve the Steiner path cover problem in linear time by the following transformation. We replace every edge $\{u, v\}$ of G by two directed edges (u, v) and (v, u) and apply our solution for directed co-graphs. This reproves our result of [GHK^{+}20b].

The directed Hamiltonian path problem can be solved by an XP-algorithm w.r.t. the parameter directed clique-width [GHO13]. Since directed co-graphs have directed clique-width at most two [GWY16] a polynomial time solution for the directed Hamiltonian path problem follows. Such an algorithm is also given in [Gur17]. A directed Hamiltonian path exists if and only if we have $T = V(G)$ and $p(G) = 1$. Thus, our results lead to the first linear time algorithm for the directed Hamiltonian path problem on directed co-graphs, which is a generalization of the known results for undirected co-graphs of Lin et al. [LOP95].

In our future work we want to investigate whether these results can be carried over to other graph classes such as chordal graphs, interval graphs, or distance-hereditary graphs.

7 NP-hard Problems on Various Recursive Digraph Classes: Digraph Coloring

7.1 Introduction

The following part is taken from [GKL20]. Coloring problems are among the most famous problems in graph theory, since they allow to model many real-life problems under a graph theoretical formalism. Graph coloring is an assignment of labels, which we call colors, to the objects of a graph subject to some constraints. Usually, vertices or edges are considered as objects. Such problems have multiple applications [Bys04, HKdW97, JP04, dWELS02]. The coloring problem on undirected graphs has been well studied, but there are not many results for coloring problems on directed graphs.

In undirected graphs we have the well known Chromatic number problem, where no two vertices that are adjacent get the same color. The target is to get the minimum number of colors for coloring the whole graph. As even the problem whether a graph has a 3-coloring, is NP-complete, finding the chromatic number of an undirected graph is an NP-hard problem. However there are many efficient solutions for the coloring problem on special graph classes, like chordal graphs [Gol80], comparability graphs [Hoà94], and co-graphs [CLSB81].

In the coloring of digraphs we differentiate between the following types of coloring. For the oriented chromatic number (OCN) we have the same condition as for the chromatic number, but additionally we set a condition to the colors with respect to the directions of the arcs in the digraph. Thus, if there is an arc from a vertex colored by 1 to a vertex colored by 2, it is not allowed to have also an arc in the oriented graph from color 2 to color 1. The oriented chromatic number is the minimum number of colors we need to color the oriented graph following these conditions. The oriented chromatic number was often used for undirected graphs, such that we look at every possible orientation of an undirected graph. Then, the maximum number of the optimal oriented colorings of every orientation leads us to the oriented coloring of the undirected graph. In this sense, there are already interesting results of the oriented coloring of special undirected graph classes. E.g., every tree has oriented chromatic number of at most 3. Oriented coloring arises in scheduling models where incompatibilities are oriented [CD06]. For several special undirected graph classes the oriented chromatic number is also bounded e.g. for outerplanar graphs [Sop97], planar graphs [Mar13], and Halin graphs [DS14]. There exists also an FPT-algorithm for OCN w.r.t. the parameter tree-width of the underlying undirected graph, see [Gan09]. Further, it is shown in the same paper that OCN is DET-hard[1] for classes of oriented graphs with bounded rank-width of the underlying undirected class. But it does also make sense to look at the oriented coloring of oriented graph classes with the following justification. If r is a constant and not part of the input, we call the corresponding problem OCN_r, which is an NP-hard problem. Nevertheless, for $r \leq 3$ we can decide OCN_r in polynomial time. While the undirected problem is easy to solve on trees, OCN_4 is NP-complete on DAGs [CD06], which are the cycle free graphs in digraphs. Thus, it is reasonable to look at the oriented chromatic number of other special oriented graph classes which we do in this chapter.

The following part is taken from [GKR21a]. The dichromatic number (DCN) is about acyclic coloring. This means we want to divide the digraph into color classes such that the vertices in each color class induces an acyclic subdigraph. Acyclic colorings of digraphs received a lot of attention several years ago [BFJ$^+$04, Moh03, NL82] but also in recent works [LM17, MSW19, SW20]. The dichromatic number is one of two basic concepts for the class of perfect digraphs [AH15] and can be regarded as a natural counterpart of the well known chromatic number for undirected graphs. If r is a constant and not part of the input we call the corresponding problem DCN_r. Even DCN_2 is NP-complete [FHM03], which motivates to consider the Dichromatic number problem on special graph classes, as well as to search for parameterized algorithms for a solution. Up to now, only few classes of digraphs are known for which the dichromatic number can be found in polynomial time. The set of DAGs is obviously equal to the set of digraphs of dichromatic number 1. Further, every odd-cycle free digraph [NL82] and every non-even digraph [MSW19] has dichromatic number at most 2. We consider the dichromatic number restricted to directed co-graphs and show a parameterized algorithm for solving the Dichromatic number problem w.r.t. the parameter directed clique-width.

The upcoming part is taken from [LGK21]. While the previously mentioned problems are considering vertex colorings, the oriented Chromatic index problem (OCI) is about arc coloring. Within the problem we have an oriented graph G and an integer r and we have to decide whether there is an oriented r-arc-coloring for G. If r is a constant and not part of the input, the corresponding problem is denoted by OCI_r. Even OCI_4 is NP-complete [OPS08]. This persuades to examine the problem in the context of special digraph classes.

In this chapter we consider the above mentioned coloring problems on different digraph classes, namely on msp and esp-digraphs, transitive DAGs, oriented, and directed co-graphs. The content of this whole chapter is from [LGK21] and [GKL20], while the dichromatic number parts as well as the parameterization in Subsection 7.6.2 are from [GKR21a]. Within

[1]DET is the class of decision problems that are reducible in logarithmic space to the problem of computing the determinant of an integer valued $n \times n$-matrix.

the latter one, we show how acyclic coloring can be parameterized my the parameter directed clique-width. Table 7.1 provides an overview in advance.

$G \in$ digraph class	coloring	bound	computation
msp-digraphs	$\chi_o(G)$	≤ 7	linear time
esp-digraphs	$\chi_o(G)$	≤ 7	linear time
transitive DAGs	$\chi_o(G)$	$= \ell(G)+1$	linear time
DAGs	$\chi_o(G)$	$\leq \ell(G)+1$	NP-hard
oriented co-graphs	$\chi_o(G)$	$= \ell(G)+1$	linear time
msp-digraphs	$\chi'_o(G)$	≤ 7	linear time
esp-digraphs	$\chi'_o(G)$	≤ 7	linear time
directed co-graphs	$\vec{\chi}(G)$	unbounded	linear time

Table 7.1: Overview about the results on coloring different digraph classes given in this chapter. Linear time means always w.r.t. number of vertices and edges in G. Notice that, $\ell(G)$ is the length of the longest oriented path in G. The last column shows how much time is needed for the calculation of the respective coloring number (χ_o = oriented chromatic number, χ'_o = oriented chromatic index, $\vec{\chi}$ = dichromatic number).

7.2 Undirected Graph Coloring

This section is taken from [GKL20]. Graph coloring of undirected graphs can be defined as follows.

Definition 7.2.1 (Graph Coloring)**.** A r-coloring of a graph G is a mapping $c : V(G) \to \{1,\ldots,r\}$ such that:

- $c(u) \neq c(v)$ for every $\{u,v\} \in E(G)$

The *chromatic number* of G, denoted by $\chi(G)$, is the smallest integer r such that G has a r-coloring.

Thus, the graph coloring problem is equivalent to the partition into independent sets problem, where we separate the vertices into subsets, such that no two vertices in a subset are connected by an edge. Accordingly, vertices with the same color build an independent set.

Name: Chromatic number problem (*CN*)
Instance: A graph G and a positive integer $r \leq |V(G)|$.
Question: Is there an r-coloring for G?

It is well known that bipartite graphs are exactly these graphs which allow a 2-coloring as well as planar graphs are graphs that allow a 4-coloring. The problem whether graph G has a 3-coloring is already NP-complete, such that the Chromatic number problem is NP-hard in general. However, coloring problems on undirected graphs allow efficient solutions when restricted to special graph classes such as chordal graphs [Gol80], comparability graphs [Hoà94], and co-graphs [CLSB81]. On undirected co-graphs, the graph coloring problem can be solved in linear time, see [CLSB81]. If r is a constant and not part of the input we get the corresponding r-Chromatic Number problem.

Name: r-Chromatic number problem (CN_r)
Instance: A graph G.
Question: Is there a r-coloring for G?

Even on 4-regular planar graphs CN_3 is NP-complete, see [Dai80]. A graph G can colored by a greedy algorithm. For some given ordering π of $V(G)$, the vertices are ordered as a sequence in which each vertex is assigned to the minimum possible value that is not forbidden by the colors of its neighbors, see Algorithm 2. Obviously, different orders can lead to different numbers of colors. But there is always an ordering yielding to the minimum number of colors, which is hard to find in general.

Algorithm 2 GREEDY COLORING

```
Data: A graph $G$ and an ordering $\pi : v_1 < \ldots < v_n$ of its vertices.
Result: Admitted vertex coloring $c : \{v_1, \ldots, v_n\} \mapsto \mathbb{N}$ of $G$.
for ($i = 1$ to $n$) do $c(v_i) = \infty$
end for
$c(v_1) = 1$;
for ($i = 2$ to $n$) do $c(v_i) = \min\{\mathbb{N} - \{c(v) \mid v \in N(v_i)\}\}$
end for
```

For the set of perfectly orderable graphs the greedy algorithm leads to an ordering with an optimal coloring, not only for the graph itself but also for all of its induced subgraphs.

Definition 7.2.2 (Perfectly orderable graph [Chv84])**.** For a graph G a linear ordering on $V(G)$ is *perfect* if a greedy coloring algorithm with that ordering optimally colors every induced subgraph of G. A graph G is *perfectly orderable* if it admits a perfect order.

Theorem 7.2.3 ([Chv84])**.** *A linear ordering π of a graph G is perfect if and only if there is no induced $P_4 = (\{a,b,c,d\},\{\{a,b\},\{b,c\},\{c,d\}\})$ in G such that $\pi(a) < \pi(b)$, $\pi(b) < \pi(c)$, and $\pi(d) < \pi(c)$.*

Since they do not contain a P_4 at all, co-graphs are perfectly orderable. For the coloring of series-parallel graphs there exists the following result.

Proposition 7.2.4 ([Sey90])**.** *Let G be some series-parallel graph. Then, it holds that $\chi(G) \leq 3$.*

7.3 Oriented Coloring

Courcelle introduced *oriented coloring* in 1994 [Cou94]. The definition of the oriented coloring of undirected graphs is of course easily transferable on oriented graphs. The following part is taken from [GKL20].

An oriented r-coloring of an oriented graph G is a partition of $V(G)$ into r independent sets, such that all arcs linking two of these r subsets have the same direction.

Definition 7.3.1 (Oriented vertex-coloring [Cou94])**.** An *oriented r-vertex-coloring*[2] of an oriented graph $G = (V,E)$ is a mapping $c : V \to \{1,\dots,r\}$ such that:

- $c(u) \neq c(v)$ for every $(u,v) \in E$,
- $c(u) \neq c(y)$ for every two arcs $(u,v) \in E$ and $(x,y) \in E$ with $c(v) = c(x)$.

The *oriented chromatic number* of G, denoted by $\chi_o(G)$, is the smallest r such that there exists an oriented r-vertex-coloring for G. Then, $V_i = \{v \in V \mid c(v) = i\}$, $1 \leq i \leq r$, is a partition of V, which we call *color classes*.

Name: Oriented chromatic number problem (*OCN*)
Instance: A graph G and an integer r.
Question: Is there an oriented coloring with r or less colors for G?

If r is a constant and not part of the input, the corresponding problem is denoted by OCN_r, which is an NP-hard problem. For two oriented graphs $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$ a *homomorphism* from G_1 to G_2, $G_1 \to G_2$ for short, is a mapping $h : V_1 \to V_2$ such that $(u,v) \in E_1$ implies $(h(u),h(v)) \in E_2$. A homomorphism from G_1 to G_2 can be regarded as an oriented coloring of G_1 that uses the vertices of G_2 as colors classes. Therefore, digraph G_2 is called a *color graph* of G_1. So, there is an oriented r-coloring of an oriented graph G_1 if and only if there is a homomorphism from G_1 to an oriented graph G_2 with r vertices. That is, the oriented chromatic number of G_1 is the minimum number of vertices in an oriented graph G_2 such that there is a homomorphism from G_1 to G_2. Obviously, it is advisable to choose G_2 as a tournament.

Observation 7.3.2. *There is an oriented r-coloring of an oriented graph G_1 if and only if there is a homomorphism from G_1 to a tournament G_2 with r vertices. The oriented chromatic number of G_1 is the minimum number of vertices in a tournament G_2 such that there is a homomorphism from G_1 to G_2.*

We define the sum of two color graphs $H_1 = (V_1,E_1)$ and $H_2 = (V_2,E_2)$ as $H_1 + H_2 = (V_1 \cup V_2, E_1 \cup E_2)$, which we use in some proofs later on.

An interesting property of oriented coloring is that the oriented chromatic number can increase when combining two graphs without adding edges. So, the oriented chromatic number of the disjoint union of two oriented graphs can grow larger than the maximum oriented chromatic number of these graphs, as the direction between the different colors play a role.

Example 7.3.3. Look at the $\overrightarrow{C_3}$ and the transitive tournament on 3 vertices, denoted by $\overrightarrow{T_3}$. When considering both digraphs on their own they both have the same oriented chromatic

[2]We say oriented coloring for short.

number $\chi_o(\overrightarrow{C_3}) = \chi_o(\overrightarrow{T_3}) = 3$. If we combine both graphs by uniting them without creating any new edges, we need more colors to fulfill the conditions of oriented coloring, otherwise the directions of the color classes collapse.

The following relation holds for oriented coloring of a digraph with respect to the oriented coloring of its underlying undirected graph.

Observation 7.3.4. *For every oriented graph G it holds that $\chi(und(G)) \leq \chi_o(G)$.*

As well, we can keep an existing oriented coloring for subdigraphs.

Lemma 7.3.5. *Let G be an oriented graph and H be a subdigraph of G. Then, it holds that $\chi_o(H) \leq \chi_o(G)$.*

7.3.1 Oriented Coloring on MSP-digraphs

The following part is from [GKL20], where also the missing proofs of the theorems can be read. As indicated before, the disjoint union of two msp-digraphs can be larger than the maximum oriented chromatic number of the involved digraphs, see Example 7.3.6. The digraphs defined by expressions X_1 and X_2 both have oriented chromatic number 4 but their disjoint union leads to a digraph with oriented chromatic number 5.

Example 7.3.6. In the following two msp-expressions we assume that the $\times$ operation binds more strongly than the $\cup$ operation.

$$X_1 = v_1 \times (v_2 \cup v_3 \times v_4) \times v_5 \times v_6$$

$$X_2 = w_1 \times (w_2 \cup w_3 \times (w_4 \cup w_5 \times w_6)) \times w_7$$

In [Sop97] there is given a bound for the oriented chromatic number of undirected series-parallel graphs. As well they have shown that this bound is tight.

Theorem 7.3.7 ([Sop97])**.** *Let G' be some orientation of a series-parallel graph G. Then, it holds that $\chi_o(G') \leq 7$.*

In [PS06] this was strengthened by giving a triangle-free orientation of a series-parallel graph of order 15 and oriented chromatic number 7. This bound can not be applied to msp-digraphs, since the set of all $\overrightarrow{K_{n,m}}$ is a subset of msp-digraphs while the underlying graphs are even of unbounded tree-width and thus, no series-parallel graphs. Nevertheless, for msp-digraphs we get the same sharp bound by 7.

Theorem 7.3.8 ([GKL20])**.** *Let G be an msp-digraph. Then, it holds that $\chi_o(G) \leq 7$.*

Digraph G on 27 vertices defined in Example 7.3.9 satisfies $\chi_o(G) = 7$, which was found by a computer program with the following procedure. We implemented an algorithm which takes an oriented graph G and an integer k as an input and which decides whether $\chi_o(G) \leq k$. The existence of such an msp-digraph implies that the bound given in Theorem 7.3.8 is best possible.

Example 7.3.9. In the following msp-expression we assume that the $\times$ operation binds more strongly than the $\cup$ operation.

$$X = v_1 \times (v_2 \cup v_3 \times (v_4 \cup v_5 \times v_6)) \times (v_7 \cup (v_8 \cup v_9 \times v_{10}) \times (v_{11} \cup v_{12} \times v_{13}))\times$$
$$(v_{14} \cup (v_{15} \cup (v_{16} \cup v_{17} \times v_{18}) \times (v_{19} \cup v_{20} \times v_{21})) \times (v_{22} \cup (v_{23} \cup v_{24} \times v_{25}) \times v_{26})) \times v_{27}$$

In order to compute the oriented chromatic number of an msp-digraph G defined by an msp-expression X, we recursively compute the set $F(X)$ of all triples (H,L,R) such that H is a color graph for G, where L and R are the sets of colors of all sinks and all sources in G with respect to the coloring by H. The number of vertex labeled, i.e., the vertices are distinguishable from each other, oriented graphs on n vertices is $3^{n(n-1)/2}$. By Theorem 7.3.8 we can conclude that

$$|F(X)| \leq 3^{7(7-1)/2} \cdot 2^7 \cdot 2^7 \in O(1)$$

which is independent of the size of G.

Lemma 7.3.10. *1. For every $v \in V$ it holds $F(v) = \{((\{i\},\emptyset),\{i\},\{i\}) \mid 0 \leq i \leq 6\}$.*

2. *For every two msp-expressions X_1 and X_2 we obtain $F(X_1 \cup X_2)$ from $F(X_1)$ and $F(X_2)$ as follows. For every $(H_1,L_1,R_1) \in F(X_1)$ and every $(H_2,L_2,R_2) \in F(X_2)$ such that graph $H_1 + H_2$ is oriented, we put $(H_1 + H_2, L_1 \cup L_2, R_1 \cup R_2)$ into $F(X_1 \cup X_2)$.*

3. *For every two msp-expressions X_1 and X_2 we obtain $F(X_1 \times X_2)$ from $F(X_1)$ and $F(X_2)$ as follows. For every $(H_1,L_1,R_1) \in F(X_1)$ and every $(H_2,L_2,R_2) \in F(X_2)$ such that graph $H_1 + H_2$ together with the arcs in $R_1 \times L_2$ is oriented, we put $((V_1 \cup V_2, E_1 \cup E_2 \cup R_1 \times L_2), L_1, R_2)$ into $F(X_1 \times X_2)$.*

Proof. 1. $F(v)$ includes obviously all possible solutions to color every vertex on its own with the seven given colors.

2. Let (H_1,L_1,R_1) be any possible solution for coloring digraph(X_1), which therefore is included in $F(X_1)$, as well as a possible solution (H_2,L_2,R_2) for coloring digraph(X_2) which is included in $F(X_2)$. Let further $H_1 + H_2$ be an oriented graph. Since the operation $\cup$ creates no additional edges in digraph($X_1 \cup X_2$), the vertices of digraph(X_1) can still be colored with H_1 and the vertices of digraph(X_2) can still be colored with H_2 such that all vertices from digraph($X_1 \cup X_2$) are colored correctly. Further, all sinks in digraph(X_1) and digraph(X_2) are also sinks in digraph($X_1 \cup X_2$). The same holds for the sources. For an oriented graph $H_1 + H_2$ this leads to $(H_1 + H_2, L_1 \cup L_2, R_1 \cup R_2) \in F(X_1 \cup X_2)$.

   Let $(H,L,R) \in F(X_1 \cup X_2)$, then there is an induced subdigraph H_1 of the color graph H which colors digraph(X_1), an induced subdigraph of digraph($X_1 \cup X_2$). Since H is oriented, H_1 is oriented. Let $L_1 \subseteq L$ be the sources with vertices in digraph(X_1) and $R_1 \subseteq R$ be the sinks for vertices in digraph(X_1). Then, it holds that $(H_1,L_1,R_1) \in F(X_1)$. The same arguments hold for X_2, such that $(H_2,L_2,R_2) \in F(X_2)$.

3. Let (H_1,L_1,R_1) be any possible solution for coloring digraph(X_1), which therefore is included in $F(X_1)$, as well as a possible solution (H_2,L_2,R_2) for coloring X_2 which

is included in $F(X_2)$. Further, let $H_1 + H_2$ together with edges from $R_1 \times L_2$ be an oriented graph. Then, $H = (V_1 \cup V_2, E_1 \cup E_2 \cup R_1 \times L_2)$ is an oriented coloring for $X = X_1 \times X_2$. Since the sinks of digraph(X_1) are connected with the sources of digraph(X_2) in digraph(X) the sources of L_1 are the only sources left in digraph(X) as well as the sinks in R_2 are the only sinks left in digraph(X). This leads to $(H, L_1, R_2) \in F(X)$.

Let $(H, L, R) \in F(X_1 \times X_2)$, then there is an induced subdigraph H_1 of the color graph H which colors digraph(X_1) which is an induced subdigraph of digraph$(X_1 \times X_2)$. Since H is oriented, H_1 is also oriented. Since all the sources of digraph$(X_1 \times X_2)$ are in digraph(X_1) it holds that $L_1 = L$ are also sources of digraph(X_1). Let R_1 be the vertices in digraph(X_1) which only have out-going neighbors in digraph(X_2) but not in digraph(X_1), then R_1 are the sinks of digraph(X_1). Thus, it holds that $(H_1, L_1, R_1) \in F(X_1)$. Simultaneously, there is an induced subdigraph H_2 of the color graph H which colors digraph(X_2) which is an induced subdigraph of digraph$(X_1 \times X_2)$. Since H is oriented, H_2 is also oriented. Since all the sinks of digraph$(X_1 \times X_2)$ are in digraph(X_2) it holds that $R_2 = R$ are also sinks of digraph(X_2). Let L_2 be the vertices in digraph(X_2) which only have in-coming neighbors in digraph(X_1) but not in digraph(X_2), then L_2 are the sources of digraph(X_2). Thus, it holds that $(H_2, L_2, R_2) \in F(X_2)$.

This shows the statements of the lemma. □

Since every possible coloring of G is part of the set $F(X)$, where X is an msp-expression for G, it is easy to find a minimum coloring for G.

Corollary 7.3.11 ([GKL20]). *There is an oriented r-coloring for some msp-digraph G which is given by an msp-expression X if and only if there is $(H, L, R) \in F(X)$ such that color graph H has r vertices. Therefore, it holds that $\chi_o(G) = \min\{|V| \mid ((V, E), L, R) \in F(X)\}$.*

Theorem 7.3.12. *For an msp-digraph G the oriented chromatic number can be computed in linear time.*

Proof. Let G be an msp-digraph on n vertices and m edges and let T be an msp-tree for G with root r. For a vertex u of T we denote by T_u the subtree rooted at u while X_u is the msp-expression defined by T_u.

In order to solve OCN for an msp-digraph G, we traverse the msp-tree T into bottom-up order. For every vertex u of T we compute $F(X_u)$ following the rules given in Lemma 7.3.10. By Corollary 7.3.11 we can solve our problem by $F(X_r) = F(X)$.

An msp-tree T can be computed in linear time from a minimal series-parallel digraph, see Observation 3.6.3. Our rules given in Lemma 7.3.10 show the following running times.

- For every $v \in V$ set $F(v)$ is computable in $O(1)$ time.
- Every set $F(X_1 \cup X_2)$ can be computed in $O(1)$ time from $F(X_1)$ and $F(X_2)$.
- Every set $F(X_1 \times X_2)$ can be computed in $O(1)$ time from $F(X_1)$ and $F(X_2)$.

Since we have n leaves and $n - 1$ inner vertices in msp-tree T, the total running time is in $O(n + m)$. □

As we see later in Corollary 7.3.15 for every oriented co-graph G it holds that the oriented chromatic number of G is equal to the chromatic number of its underlying undirected graph. This does not hold for msp-digraphs by Example 7.3.9 and the next result, which can be shown on a similar way as the bound for msp-digraphs.

Proposition 7.3.13 ([GKL20]). *For an msp-digraph G it holds that $\chi(und(G)) \leq 3$.*

7.3.2 Oriented Coloring on Transitive Acyclic Digraphs

Next, we will apply the concept of perfectly orderable graphs and Theorem 7.2.3 in order to color transitive acyclic digraphs. This part is as well taken from [GKL20].

Theorem 7.3.14. *Every greedy coloring along a topological ordering of a transitive DAG G leads to an optimal oriented coloring of G while $\chi_o(G)$ can be computed in linear time.*

Proof. Since G is a DAG there is a topological ordering t for G. As G is transitive, it does not contain a special orientation of the P_4, namely the N graph (see Figure 3.5), as an induced subdigraph. Theorem 7.2.3 implies that every linear ordering and thus, also t is perfect on $und(G)$. Let $c : V(G) \to \{1,\ldots,k\}$ be a coloring for $und(G)$ obtained by the greedy Algorithm 2 for t on $V(G)$. It remains to show that c is an oriented coloring for G.

- $c(u) \neq c(v)$ holds for every $(u,v) \in E(G)$ since $c(u) \neq c(v)$ holds for every $\{u,v\} \in E(und(G))$.
- $c(u) \neq c(y)$ for every two arcs $(u,v) \in E(G)$ and $(x,y) \in E(G)$ with $c(v) = c(x)$ holds by the following argumentation. Assume there is an arc $(v_i,v_j) \in E(G)$ with $v_i < v_j$ in t but $c(v_i) > c(v_j)$. Then, when coloring v_i we would have taken $c(v_j)$ if possible, as we always take the minimum possible color value. Since this was not possible there must have been an other vertex $v_k < v_i$ which was colored before v_i with $c(v_k) = c(v_j)$ and $(v_k,v_i) \in E(G)$. But if $(v_k,v_i) \in E(G)$ and $(v_i,v_j) \in E(G)$, due to transitivity it must also hold that $(v_k,v_j) \in E(G)$ and consequently, $c(v_k) = c(v_j)$ is not possible. Therefore, the assumption was wrong and for every arc $(v_i,v_j) \in E(G)$ with $v_i < v_j$ in t it must hold that $c(v_i) < (c_j)$.

The optimality of oriented coloring c holds since the lower bound of Observation 7.3.4 is achieved. $\square$

The proof of Theorem 7.3.14 leads also to an optimal oriented coloring. In order to state the next result, let $\omega(H)$ be the number of vertices in a largest clique in the (undirected) graph H.

Corollary 7.3.15 ([GKL20]). *Let G be a transitive DAG. Then, it holds that $\chi_o(G) = \chi(und(G)) = \omega(und(G))$ and all values can be computed in linear time.*

For some oriented graph G we denote by $\ell(G)$ the length of a longest oriented path in G.

Proposition 7.3.16 ([GKL20]). *Let G be a transitive acyclic digraph. Then, it holds that $\chi_o(G) = \ell(G) + 1$.*

Next, we consider oriented colorings of oriented graphs with bounded vertex degree. For every oriented graph its oriented chromatic number can be bounded (exponentially) by its maximum degree Δ according to [KSZ97]. For small values $\Delta \leq 7$ there are better bounds in [Duf19] and [DOPS20]. By Proposition 7.3.16 and the observation that the first vertex of a longest path within a transitive digraph G has a minimum outdegree of $\ell(G)$, it follows that the oriented chromatic number of G can be estimated as follows.

Corollary 7.3.17 ([GKL20])**.** *Let G be a transitive acyclic digraph. Then, it holds that* $\chi_o(G) \leq \Delta(G)+1$.

The next proposition can be proved over the coloring of the transitive closure of a digraph.

Proposition 7.3.18 ([GKL20])**.** *Let G be an acyclic digraph. Then, it holds that* $\chi_o(G) \leq \ell(G)+1$.

7.3.3 Oriented Coloring on Oriented Co-graphs

We continue with the following results about the oriented coloring of oriented co-graphs which can be found in [GKR19c].

Proposition 7.3.19 ([GKR19c])**.** *For two vertex-disjoint oriented co-graphs G_1 and G_2, the following equations hold.*

1. $\chi_o(v) = 1$
2. $\chi_o(G_1 \oplus G_2) = \max(\chi_o(G_1), \chi_o(G_2))$
3. $\chi_o(G_1 \oslash G_2) = \chi_o(G_1) + \chi_o(G_2)$

Theorem 7.3.20 ([GKR19c])**.** *For an oriented co-graph G, an optimal oriented coloring and $\chi_o(G)$ can be computed in linear time.*

The coloring algorithms in [GKR19c] are based on a dynamic programming along the di-co-tree of a given oriented co-graph. Since every oriented co-graph is transitive and acyclic, Theorem 7.3.14 leads to the next result, which re-proves Theorem 7.3.20.

Corollary 7.3.21. *For an oriented co-graph G every greedy coloring along a topological ordering leads to an optimal oriented coloring and $\chi_o(G)$ can be computed in linear time.*

Note that, Theorem 7.3.14 is more general than Corollary 7.3.21 since it does not exclude N which is a forbidden induced subdigraph for oriented co-graphs. With Theorem 3.3.10, it holds that

$$\mathrm{OC} = \mathrm{Free}\{\overleftrightarrow{P_2}, \overrightarrow{P_3}, \overrightarrow{C_3}, N\} \subseteq \mathrm{Free}\{\overleftrightarrow{P_2}, \overrightarrow{P_3}, \overrightarrow{C_3}\}$$

and $\mathrm{Free}\{\overleftrightarrow{P_2}, \overrightarrow{P_3}, \overrightarrow{C_3}\}$ is equivalent to the set of all acyclic transitive digraphs. Since every oriented co-graph is transitive and acyclic, Corollary 7.3.17 leads to the following bound.

Corollary 7.3.22. *For an oriented co-graph G it holds that* $\chi_o(G) \leq \Delta(G)+1$.

There are classes of oriented co-graphs, e.g., the class of all $\overrightarrow{K_{1,n}}$, for which the oriented chromatic number is even bounded by a constant and thus, is smaller than the shown bound. Considering transitive tournaments we conclude that the bound given in Corollary 7.3.22 is best possible.

7.3.4 Oriented Coloring on ESP-digraphs

The content from this subsection is taken from [LGK21]. Since every esp-digraph is an orientation of a series-parallel graph by Theorem 7.3.7 and from [Sop97] we get the following bound.

Corollary 7.3.23. *Let G be an esp-digraph. Then, it holds that $\chi_o(G) \leq 7$.*

There is an alternative way of proving this bound by using the recursive structure of the class of esp-digraphs, see [LGK21]. For the optimality of the shown bound, we give the following example.

Example 7.3.24. The esp-expression

$$\begin{aligned} X_6 \quad = \quad & ((v_1,v_4)\cup((v_1,v_2)\times((v_2,v_4)\cup((v_2,v_3)\times(v_3,v_4)))))\times \\ & ((((v_4,v_6)\cup((v_4,v_5)\times(v_5,v_6)))\times(v_6,v_7))\cup(v_4,v_7)) \end{aligned}$$

defines the esp-digraph on 7 vertices shown in Figure 7.1 and it obviously holds that $\chi_o(\text{digraph}(X_6)) = 7$. This implies that the bound of Corollary 7.3.23 is best possible.

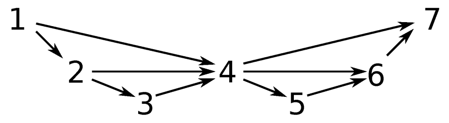


Figure 7.1: Digraph(X_6) in Example 7.3.24.

In order to compute the oriented chromatic number of an esp-digraph G defined by an esp-expression X, we recursively compute the set $F(X)$ of all triples (H,ℓ,r) such that H is a color graph for G, where ℓ and r are the colors of the source and sink, in G with respect to the coloring by H. The number of vertex labeled oriented graphs (i.e., the vertices are distinguishable from each other) on n vertices is $3^{n(n-1)/2}$. By Corollary 7.3.23 we can conclude that

$$|F(X)| \leq 3^{7(7-1)/2}\cdot 7\cdot 7 \in O(1)$$

which is independent of the size of G.

Lemma 7.3.25. *Let $G=(V,E)$ be an esp-digraph. Then,*

1. *For every $(u,v)\in E$ it holds*

$$F((u,v)) = \{((\{i,j\},\{(i,j)\}),i,j) \mid 1\leq i,j\leq 7,\ i\neq j\}.$$

2. *For every two esp-expressions X_1 and X_2 we obtain $F(X_1\cup X_2)$ from $F(X_1)$ and $F(X_2)$ as follows. For every $(H_1,\ell_1,r_1)\in F(X_1)$ and every $(H_2,\ell_2,r_2)\in F(X_2)$ such that graph H_1+H_2 is oriented, $\ell_1=\ell_2$, and $r_1=r_2$, we put (H_1+H_2,ℓ_1,r_1) into $F(X_1\cup X_2)$.*

3. *For every two esp-expressions X_1 and X_2 we obtain $F(X_1\times X_2)$ from $F(X_1)$ and $F(X_2)$ as follows. For every $(H_1,\ell_1,r_1)\in F(X_1)$ and every $(H_2,\ell_2,r_2)\in F(X_2)$ such that graph H_1+H_2 is oriented, and $r_1=\ell_2$, we put $((V_1\cup V_2,E_1\cup E_2),\ell_1,r_2)$ into $F(X_1\times X_2)$.*

Proof. We show for each operation that the stated formulas hold.

1. Set $F((u,v))$ includes obviously all possible solutions to color the end vertices of every arc on its own with the 7 given colors.

2. Set $F(X_1)$ includes all possible solutions for coloring X_1, just as $F(X_2)$ for X_2. In particular we have solutions included, that are equal but a permutation of the colors. Since the sources and sinks are each identified with each other, we only keep solutions where $\ell_1 = \ell_2$ and $r_1 = r_2$. In this step it is essential that we kept all possible solutions before, even if they are just permutations of the different colors. $H_1 + H_2$ is oriented and has by construction at most 7 vertices. Since in $X_1 \cup X_2$ there are no additional edges compared to $E_1 \cup E_2$, every vertex can get the same color as in the individual solutions, such that all vertices are legally colored. So $(H_1 + H_2, \ell_1, r_1)$ is a possible solution to color and thus $(H_1 + H_2, \ell_1, r_1) \in F(X_1 \cup X_2)$.

   Let $(H, \ell, r) \in F(X_1 \cup X_2)$, then we can take an induced subdigraph H_1 which colors all the vertices of digraph(X_1) as well as H_2 which colors all the vertices of H_2. Let $\ell_1 = \ell$ be the color of the source in digraph(X_1) and $r_1 = r$ the color of the sink in digraph(X_2). It holds that $(H_1, \ell_1, r_1) \in F(X_1)$. The same arguments hold for X_2 such that $(H_2, \ell_2, r_2) \in F(X_2)$.

3. Set $F(X_1)$ includes all possible solutions for coloring X_1, just as $F(X_2)$ for X_2. In particular we have solutions included, that are equal but a permutation of the colors. Since the source and the sink are identified with each other, we only keep solutions where $r_1 = \ell_2$. In this step it is essential that we kept all possible solutions before, even if they are just permutations of the different colors. $H_1 + H_2$ is oriented and has by construction at most 7 vertices. In $X_1 \times X_2$ there no additional edges compared to $E_1 \cup E_2$, every vertex can get the same color as in the individual solutions, such that all vertices are legally colored. So $(H_1 + H_2, \ell_1, r_2)$ is a possible solution and thus, $(H_1 + H_2, \ell_1, r_2) \in F(X_1 \times X_2)$.

   Let $(H, \ell, r) \in F(X_1 \times X_2)$, then we can take an induced subdigraph H_1 which colors all the vertices of digraph(X_1) as well as H_2 which colors all the vertices of H_2. Let $\ell_1 = \ell$ be the color of the source in digraph(X_1) and r_1 the color of the sink in digraph(X_2). It holds that $(H_1, \ell_1, r_1) \in F(X_1)$. The same arguments hold for X_2, if ℓ_2 is the color of the source of digraph(X_1) and $r_1 = r$ is the color of the sink of digraph(X_2), such that $(H_2, \ell_2, r_2) \in F(X_2)$.

This shows the statements of the lemma. □

The optimal solution for digraph G given by some esp-expression X is always included in $F(X)$ since all possible subsolutions are maintained in the process and not only the optimal solutions. We can show this shortly by contradiction. Assumed there exists an optimal solution (H, ℓ, r) for $X_1 \times X_2$, but $(H, \ell, r) \notin F(X_1 \times X_2)$ such that (H, ℓ, r) was not taken into the solution. Thus, for either X_1 or X_2 (which we call X_i in the following), the solution of coloring the vertices with color graph H', which is an induced subdigraph of H and which only contains the colors we need for coloring X_i, was not part of the solution $F(X)$. But since there are all the possible solutions in $F(X)$ and not only minimal solutions, this is a contradiction to our procedure. The same holds for the parallel composition $X_1 \cup X_2$. Thus, we find a minimal coloring for G.

Corollary 7.3.26. *There is an oriented vertex r-coloring for some esp-digraph G which is given by some esp-expression X if and only if there is some* $(H,\ell,r) \in F(X)$ *such that color graph H has r vertices. Therefore,* $\chi_o(G) = \min\{|V| \mid ((V,E),\ell,r) \in F(X)\}$.

Theorem 7.3.27. *Let G be an esp-digraph. Then, the oriented chromatic number of G can be computed in linear time.*

Proof. Let $G = (V,E)$ be an esp-digraph with $n = |V|$ vertices and $m = |E|$ edges and let T be an esp-tree for G with root r. For a vertex u of T we denote by T_u the subtree rooted at u and by X_u the esp-expression defined by T_u.
For computing the oriented chromatic number for some esp-digraph G, we traverse esp-tree T in bottom-up order. For every vertex u of T we can compute $F(X_u)$ by following the rules given in Lemma 7.3.25. By Corollary 7.3.26 we can solve our problem using $F(X_r) = F(X)$. An esp-tree T can be computed in $O(n+m)$ time from G, see [Val78]. By Lemma 7.3.25 we obtain the following running times.

- For every arc $(u,v) \in E$ set $F((u,v))$ is computable in $O(1)$ time.
- For every two esp-expressions X_1 and X_2 set $F(X_1 \cup X_2)$ can be computed in $O(1)$ time from $F(X_1)$ and $F(X_2)$.
- For every two esp-expressions X_1 and X_2 set $F(X_1 \times X_2)$ can be computed in $O(1)$ time from $F(X_1)$ and $F(X_2)$.

Since T consists of n leaves and $n-1$ inner vertices, the overall running time is in $O(n+m)$. □

7.4 g-oriented r-coloring

In [GKL21b] the concept of oriented coloring excluding homomorphisms to digraphs with short cycles is introduced. A g-oriented r-coloring of an oriented graph G is a homomorphism from G to some digraph H on r vertices of girth at least $g+1$. The g-oriented chromatic number of G is the smallest integer r such that G allows a g-oriented r-coloring. It holds that for every msp-digraph the g-oriented chromatic number is at most $2^{g+1}-1$. This bound together with the recursive structure of msp-digraphs lead to a linear time solution for computing the g-oriented chromatic number of msp-digraphs. This reproves the already known result, that every msp-digraph has oriented chromatic number at most 7. A well known concept in graph theory is the concept of graph powers, see [BJG18] and [BLS99]. A k-power graph G^k of a digraph G is a digraph with the same vertex set as G and an arc (u,v) is in G^k if and only if $u \neq v$ and there exists a directed path from u to v in G of length at most k. Thus, G^1 is G. As well, k-power digraphs of msp-digraphs have oriented chromatic number at most $2^{2k+1}-1$. More about this can be found in [GKL21b].

7.5 Oriented Arc-coloring

While the previous sections deal with vertex-coloring, we continue now with arc-coloring. Therefore, we introduce the oriented chromatic index problem and show bounds for special graph classes. The content from this whole arc-coloring section is taken from [LGK21].

Definition 7.5.1 (Oriented arc-coloring [OPS08])**.** An *oriented r-arc-coloring* of an oriented graph $G = (V,E)$ is a mapping $c : E \to \{1,\dots,r\}$ such that:

- $c((u,v)) \neq c((v,w))$ for every two arcs $(u,v) \in E$ and $(v,w) \in E$.
- $c((u,v)) \neq c((y,z))$ for every four arcs $(u,v) \in E$, $(v,w) \in E$, $(x,y) \in E$, and $(y,z) \in E$, with $c((v,w)) = c((x,y))$.

The *oriented chromatic index* of G, denoted with $\chi'_o(G)$, is the smallest r such that G has an oriented r-arc-coloring. Then, $E_i = \{e \in E \mid c(e) = i\}$, $1 \leq i \leq r$, is a partition of E which we call again *color classes*.

There is an oriented r-arc-coloring of an oriented graph G if and only if there is a homomorphism from line digraph $LD(G)$ to some oriented graph H on r vertices. Then, the oriented chromatic index of G is the minimum number of vertices in an oriented graph H such that there is a homomorphism from line digraph $LD(G)$ to H. The oriented chromatic index problem can be defined as follows.

Name: Oriented chromatic index problem (OCI)
Instance: An oriented graph G and a positive integer $r \leq |E(G)|$.
Question: Is there an oriented r-arc-coloring for G?

If r is a constant, i.e., not part of the input, the corresponding problem is denoted by OCI_r. If $r \leq 3$, then we can decide OCI_r in polynomial time, but even OCI_4 is NP-complete [OPS08]. The hardness of OCI_4 motivates to consider the oriented chromatic index of special graph classes. The definition of oriented arc-coloring was often used for undirected graphs, where the maximum value $\chi'_o(G')$ of all possible orientations G' of a graph G is considered. There are already bounds on the oriented chromatic index for special graph classes, e.g. for planar graphs [OPS08] and outerplanar graphs [PS06].

Observation 7.5.2 ([OPS08])**.** *Let G be an oriented graph. Then, it holds that $\chi'_o(G) = \chi_o(LD(G))$.*

Observation 7.5.3 ([OPS08])**.** *Let G be an oriented graph. Then, it holds that $\chi'_o(G) \leq \chi_o(G)$.*

One can find an equivalent characterizations for OCI using binary integer programs.

Remark 7.5.4. To formulate OCI for some oriented graph $G = (V,E)$ with $n = |V|$ and $m = |E|$ as a binary integer program, we introduce a binary variable $y_k \in \{0,1\}$, $k \in \{1,\dots,n\}$, such that $y_k = 1$ if and only if color k is used.[3] Further, we use $m \cdot n \leq n^3$ variables $x_{i,j,k} \in \{0,1\}$,

[3]By Observation 7.5.3 we need at most n colors.

$i,j,k \in \{1,\dots,n\}$, such that $x_{i,j,k} = 1$ if and only if edge (v_i,v_j) receives color k. The main idea is to ensure the two conditions of Definition 7.5.1 within conditions (7.3) and (7.5). W.l.o.g. we assume that E has at least two arcs belonging to a directed path of length two.

$$\text{Minimize} \sum_{k=1}^{n} y_k \tag{7.1}$$

subject to

$$\begin{aligned}
\sum_{k=1}^{n} x_{i,j,k} &= 1 \text{ for every } (v_i,v_j) \in E && (7.2)\\
x_{i_0,i_1,k} + x_{i_1,i_2,k} &\leq y_k \text{ for every } (v_{i_0},v_{i_1}),(v_{i_1},v_{i_2}) \in E,\ k \in \{1,\dots,n\} && (7.3)\\
\bigvee_{k=1}^{n} x_{i_1,i_2,k} \wedge x_{i_3,i_4,k} &\leq 1 - \bigvee_{k=1}^{n} x_{i_0,i_1,k} \wedge x_{i_4,i_5,k} && (7.4)\\
&\quad \text{for every } (v_{i_0},v_{i_1}),(v_{i_1},v_{i_2}),(v_{i_3},v_{i_4}),(v_{i_4},v_{i_5}) \in E && (7.5)\\
y_k &\in \{0,1\} \text{ for every } k \in \{1,\dots,n\} && (7.6)\\
x_{i,j,k} &\in \{0,1\} \text{ for every } i,j,k \in \{1,\dots,n\} && (7.7)
\end{aligned}$$

Equations (7.5) are not in propositional logic. In order to reformulate them for binary integer programming, one can use the results of [Gur14].

7.5.1 Oriented Arc-coloring of ESP-digraphs

This subsection is taken from [LGK21]. For the chromatic index of orientations of undirected series-parallel graphs Observation 7.5.3 and Theorem 7.3.7 lead to the following bound.

Corollary 7.5.5 ([PS06])**.** *Let G' be some orientation of a series-parallel graph G. Then, it holds that $\chi'_o(G') \leq 7$.*

They even showed that the bound is tight (even for an orientation of an outerplanar graph). Since every esp-digraph is an orientation of a series-parallel graph by Corollary 7.5.5 we have the following bound.

Corollary 7.5.6. *Let G be an esp-digraph. Then, it holds that $\chi'_o(G) \leq 7$.*

Alternatively, the last result can be obtained from Theorem 7.3.8, Lemma 3.6.15 and Observation 7.5.2.

Remark 7.5.7. We can also bound the oriented chromatic index of an esp-digraph G using the corresponding line digraph $LD(G)$ which is a vertex series-parallel digraph by Lemma 3.6.15.

$$\begin{aligned}
\chi'_o(G) &= \chi_o(LD(G)) && \text{Observation7.5.2}\\
&\leq 7 && \text{Lemma 3.6.15 and Theorem 7.3.8}
\end{aligned}$$

The results of [PS06] even show that 7 is a tight upper bound for the oriented chromatic index of every orientation of series-parallel graphs (even for an orientation of an outerplanar graph). In order to show that this bound is also tight for the subclass of esp-digraphs we give the next example.

Example 7.5.8. The esp-expression

$$\begin{aligned} X_5 \;=\;& (v_1,v_2)\times((v_2,v_5)\cup(v_2,v_3)\times((v_3,v_5)\cup(v_3,v_4)\times(v_4,v_5)))\times\\ & ((v_5,v_9)\cup((v_5,v_7)\cup(v_5,v_6)\times(v_6,v_7))\times((v_7,v_9)\cup(v_7,v_8)\times(v_8,v_9)))\times\\ & ((v_9,v_{16})\cup((v_9,v_{13})\cup((v_9,v_{11})\cup(v_9,v_{10})\times(v_{10},v_{11}))\times\\ & ((v_{11},v_{13})\cup(v_{11},v_{12})\times(v_{12},v_{13})))\times\\ & ((v_{13},v_{16})\cup((v_{13},v_{15})\cup(v_{13},v_{14})\times(v_{14},v_{15}))\times(v_{15},v_{16})))\times(v_{16},v_{17}) \end{aligned}$$

defines the esp-digraph on 17 vertices shown in Figure 7.2. Further, by Observation 7.5.2 and since digraph(X_2),where X_2 is defined in Example 7.3.9, is the line digraph of digraph(X_5) it holds that

$$\chi'_o(\text{digraph}(X_5)) = \chi_o(LD(\text{digraph}(X_5))) = \chi_o(\text{digraph}(X_2)) = 7.$$

This implies that the bound of Corollary 7.5.6 is best possible.

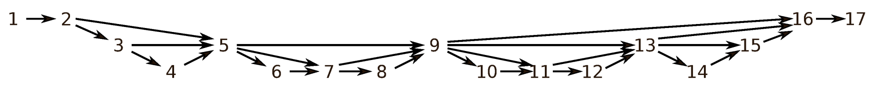


Figure 7.2: Digraph(X_5) in Example 7.5.8.

By Theorem 7.3.12 and Observation 7.5.2 we obtain the following result.

Theorem 7.5.9. *Let G be an esp-digraph. Then, the oriented chromatic index of G can be computed in linear time.*

7.5.2 Oriented Arc-coloring of MSP-digraphs

This subsection is taken from [LGK21]. By Theorem 7.3.8 and Observation 7.5.3 we know the following bound.

Corollary 7.5.10. *Let G be an msp-digraph. Then, it holds that* $\chi'_o(G) \leq 7$.

For the optimality of the shown bound, we next give an example.

Example 7.5.11. We recursively define msp-expressions Y_i as follows. Y_0 defines a single vertex graph and for $i \geq 1$ we define

$$Y_i = (Y_0 \cup Y_{i-1} \times Y_{i-1})$$

in order to define

$$X_3 = Y_0 \times Y_0 \times Y_6 \times Y_0 \times Y_0.$$

Digraph(X_3) has 131 vertices and satisfies $\chi'_o(\text{digraph}(X_3)) = 7$, which was found by a computer program (Remark 7.5.4) using Gurobipy.

This implies that the bound of Corollary 7.5.10 is best possible.

Remark 7.5.12. In order to compute the oriented chromatic index we have given a binary integer program in Remark 7.5.4. From a theoretical point of view this implies the existence of an FPT-algorithm for OCI w.r.t. parameter n, since integer linear programming is fixed-parameter tractable for the parameter number of variables [Len83].

7.6 Acyclic Coloring

The content from this whole section is from [GKR20a] and is also partly published in [GKR21a]. Coming from the coloring of oriented graphs, we now continue with the coloring of digraphs, such that we include bidirectional arcs. The following approach for coloring digraphs is given in [NL82]. A set V' of vertices of a digraph G is called *acyclic* if $G[V']$ is acyclic.

Definition 7.6.1 (Acyclic graph coloring [NL82])**.** An *acyclic r-coloring* of a digraph G is a mapping $c : V(G) \to \{1,\dots,r\}$, such that the color classes $c^{-1}(i)$ for $1 \leq i \leq r$ are acyclic. The *dichromatic number* of G, denoted by $\vec{\chi}(G)$, is the smallest r, such that G has an acyclic r-coloring.

There are several works about acyclic graph coloring [BFJ+04, Moh03, NL82] including the recent works [LM17, MSW19, SW20]. The Dichromatic number problem can be defined as follows.

Name: Dichromatic number problem (DCN)
Instance: A digraph G and a positive integer $r \leq |V(G)|$.
Question: Is there an acyclic r-coloring for G?

If r is a constant and not part of the input, the corresponding problem is denoted by r-Dichromatic number problem (DCN_r). Even DCN_2 is NP-complete [FHM03].

The following observations support that the dichromatic number can be regarded as a natural counterpart of the well known chromatic number $\chi(G)$ for undirected graphs G.

Observation 7.6.2. *For every symmetric digraph G it holds that* $\vec{\chi}(G) = \chi(und(G))$.

Observation 7.6.3. *For every digraph G it holds that* $\vec{\chi}(G) \leq \chi(und(G))$.

Observation 7.6.4. *Let G be a digraph and H be a subdigraph of G, then* $\vec{\chi}(H) \leq \vec{\chi}(G)$.

7.6.1 Acyclic Coloring on Directed Co-graphs

As recently mentioned in [SW19], only few classes of digraphs for which the dichromatic number can be found in polynomial time are known. The set of DAGs is obviously equal to the set of digraphs of dichromatic number 1. Every odd-cycle free digraph [NL82] and every non-even digraph [MSW19] has dichromatic number at most 2. Thus, for DAGs, odd-cycle free digraphs, and non-even digraphs the dichromatic number can be computed in linear time. Furthermore, for every perfect digraph the dichromatic number can be found in polynomial time [AH15]. We next show how to find an optimal acyclic coloring for directed co-graphs, which are defined below, in linear time. Given the ex-di-co-tree, an optimal acyclic coloring can as well be computed in linear time on extended directed co-graphs.

Lemma 7.6.5. *Let G_1 and G_2 be two vertex-disjoint directed graphs. Then, the following equations hold:*

1. $\vec{\chi}(\bullet) = 1$

2. $\vec{\chi}(G_1 \oplus G_2) = \vec{\chi}(G_1 \oslash G_2) = \vec{\chi}(G_1 \ominus G_2) = \max(\vec{\chi}(G_1), \vec{\chi}(G_2))$

3. $\vec{\chi}(G_1 \otimes G_2) = \vec{\chi}(G_1) + \vec{\chi}(G_2)$

Proof. 1. $\vec{\chi}(\bullet) = 1$ is obviously clear.

2. $\vec{\chi}(G_1 \oplus G_2) \geq \max(\vec{\chi}(G_1), \vec{\chi}(G_2))$ Since the digraphs G_1 and G_2 are induced subdigraphs of digraph $G_1 \oplus G_2$, both values $\vec{\chi}(G_1)$ and $\vec{\chi}(G_2)$ lead to a lower bound for the number of necessary colors of the combined digraph by Observation 7.6.4.

   $\vec{\chi}(G_1 \oplus G_2) \leq \max(\vec{\chi}(G_1), \vec{\chi}(G_2))$ Since the disjoint union operation does not create new arcs, we can combine the color classes of G_1 and G_2.

   The results for the order composition and directed union follow by the same arguments.

3. $\vec{\chi}(G_1 \otimes G_2) \geq \vec{\chi}(G_1) + \vec{\chi}(G_2)$

   Since both digraphs G_1 and G_2 are induced subdigraphs of digraph $G_1 \otimes G_2$, both values $\vec{\chi}(G_1)$ and $\vec{\chi}(G_2)$ lead to a lower bound for the number of necessary colors of the combined digraph by Observation 7.6.4. Further, the series composition implies that every vertex in G_1 is on a cycle of length two with every vertex of G_2. Thus, no vertex in G_1 can be colored in the same way as a vertex in G_2. So, $\vec{\chi}(G_1) + \vec{\chi}(G_2)$ leads to a lower bound for the number of necessary colors of the combined digraph.

   $\vec{\chi}(G_1 \otimes G_2) \leq \vec{\chi}(G_1) + \vec{\chi}(G_2)$

   For $1 \leq i \leq 2$ let $G_i = (V_i, E_i)$ and $c_i : V_i \to \{1, \ldots, \vec{\chi}(G_i)\}$ a coloring for G_i. For $G_1 \otimes G_2 = (V, E)$ we define a mapping $c_i : V_i \to \{1, \ldots, \vec{\chi}(G_1) + \vec{\chi}(G_2)\}$ as follows.

$$c(v) = \begin{cases} c_1(v) & \text{if } v \in V_1 \\ c_2(v) + \vec{\chi}(G_1) & \text{if } v \in V_2 \end{cases}$$

   The mapping c satisfies the definition of an acyclic coloring, because every color class $c^{-1}(j), j \in \{1, \ldots, \vec{\chi}(G_1) + \vec{\chi}(G_2)\}$ is a subset of V_1 or of V_2, such that $c^{-1}(j)$ induces an acyclic digraph in G_1 or G_2 by assumption. Since the series operation does not insert any further arcs between two vertices of G_1 and G_2, vertex set $c^{-1}(j)$ induces also an acyclic digraph in G.

This shows the statements of the lemma. □

With Lemma 7.6.5 we immediately get the following Theorem.

Theorem 7.6.6. *Let G be a directed co-graph. Then, an optimal acyclic coloring for G and $\vec{\chi}(G)$ can be computed in linear time.*

This holds as well for extended directed co-graphs if the ex-di-co-tree is given. The *clique number* $\omega_d(G)$ of a digraph G is the number of vertices in a largest complete bioriented subdigraph of G and the *clique number* $\omega(G)$ of a (-n undirected) graph G is the number of vertices in a largest complete subgraph of G. Since the results of Lemma 7.6.5 also hold for ω_d instead of $\vec{\chi}$ we obtain the following result.

Proposition 7.6.7. *Let G be an extended directed co-graph. Then, it holds that*

$$\vec{\chi}(G) = \chi(und(sym(G))) = \omega(und(sym(G))) = \omega_d(G)$$

and all values can be computed in linear time.

7.6.2 Acyclic Coloring Parameterized by Directed Clique-width

The Dichromatic number problem remains hard even for inputs of bounded directed feedback vertex set size [MSW19]. This result implies that there are no XP-algorithms for the Dichromatic number problem parameterized by directed width parameters such as directed path-width, directed tree-width, DAG-width or Kelly-width. The first positive result concerning structural parameterizations of the Dichromatic number problem is the existence of an FPT-algorithm for the Dichromatic number problem parameterized by directed modular width [SW19].

We show a polynomial-time algorithm for the Dichromatic number problem on digraphs of constant directed clique-width. Therefore, we consider a directed clique-width expression X of the input digraph G of directed clique-width k. For each node t of the corresponding rooted expression-tree T we use label-based reachability information about the subgraph G_t of the subtree rooted at t. For every partition of the vertex set of G_t into acyclic sets $V_1,\ldots,V_s$ we compute the multi set $\langle \text{reach}(V_1),\ldots,\text{reach}(V_s)\rangle$, where $\text{reach}(V_i)$, $1 \leq i \leq s$, is the set of all label pairs (a,b) such that the subgraph of G_t induced by V_i contains a vertex labeled by b, which is reachable by a vertex labeled by a. By using bottom-up dynamic programming along expression-tree T, we obtain an algorithm for the Dichromatic number problem of running time $n^{2^{O(k^2)}}$ where n denotes the number of vertices of the input digraph. Since any algorithm with running time in $n^{2^{o(k)}}$ would disprove the Exponential Time Hypothesis (ETH), the exponential dependence on k in the degree of the polynomial cannot be avoided, unless ETH fails.

From a parameterized point of view, the algorithm we present shows that the Dichromatic number problem is in XP when parameterized by directed clique-width. Further, we show that the Dichromatic number problem is W$[1]$-hard on symmetric digraphs when parameterized by directed clique-width. Inferring from this, there is no FPT-algorithm for the Dichromatic number problem parameterized by directed clique-width under reasonable assumptions. The best parameterized complexity, which can be achieved, is given by an XP-algorithm. Furthermore, we apply definability within monadic second order logic (MSO_1) in order to show that Dichromatic number problem is in FPT when parameterized by the directed clique-width and r, which implies that for every integer r it holds that DCN_r is in FPT when parameterized by directed clique-width.

Since the directed clique-width of a digraph is at most its directed modular width [SW20], we reprove the existence of an XP-algorithm for DCN and an FPT-algorithm for DCN_r parameterized by directed modular width [SW19]. On the other hand, there exist several classes of digraphs of bounded directed clique-width and unbounded directed modular width, which implies that directed clique-width is the more powerful parameter and thus, the results of [SW19] does not imply any parameterized algorithm for directed clique-width.

In Table 7.2 we summarize the known results for DCN and DCN$_r$ parameterized by width parameters.

parameter	DCN		DCN$_r$	
directed modular width	FPT	[SW19]	FPT	[SW19]
directed clique-width	W[1]-hard XP	Corollary 7.6.10 Corollary 7.6.17	FPT	Corollary 7.6.20
directed clique-width $+ r$	FPT	Theorem 7.6.19	///	
directed tree-width	$\not\in$ XP	[MSW19]	$\not\in$ XP	[MSW19]
directed path-width	$\not\in$ XP	[MSW19]	$\not\in$ XP	[MSW19]
DAG-width	$\not\in$ XP	[MSW19]	$\not\in$ XP	[MSW19]
Kelly-width	$\not\in$ XP	[MSW19]	$\not\in$ XP	[MSW19]
clique-width of $und(G)$	$\not\in$ FPT	by Corollary 7.6.10	open	

Table 7.2: Complexity of DCN and DCN$_r$ parameterized by width parameters. We assume that P $\neq$ NP. The "///" entries indicate that by taking r out of the instance the considered parameter makes no sense.

The first positive result concerning structural parameterizations of DCN was recently given in [SW19] using the directed modular width (dmw).

Theorem 7.6.8 ([SW19])**.** *The Dichromatic number problem is in FPT when parameterized by directed modular width.*

By [GHK$^+$14], directed clique-width performs much better than directed path-width, directed tree-width, DAG-width, and Kelly-width from the parameterized complexity point of view. Hence, we consider the parameterized complexity of DCN parameterized by directed clique-width.

Directed clique-width is not comparable to the directed variants of tree-width mentioned above, which can be observed by the set of all complete biorientations of cliques and the set of all acyclic orientations of grids, see Figure 2.6 for an example. The relation of directed clique-width and directed modular width [SW20] is as follows.

Lemma 7.6.9 ([SW20])**.** *For every digraph G it holds that d-cw$(G) \leq$ dmw(G).*

On the other hand, there exist several classes of digraphs of bounded directed clique-width and unbounded directed modular width, e.g. even the set of all directed paths $\{\overrightarrow{P_n} \mid n \geq 1\}$, the set of all directed cycles $\{\overrightarrow{C_n} \mid n \geq 1\}$, and the set of all minimal series-parallel digraphs [VTL82]. Thus, the result of [SW19] does not imply any XP-algorithm or FPT-algorithm for directed clique-width.

Corollary 7.6.10. *The Dichromatic number problem is W$[1]$-hard on symmetric digraphs and thus, on all digraphs when parameterized by directed clique-width.*

Thus, under reasonable assumptions there is no FPT-algorithm for the Dichromatic number problem parameterized by directed clique-width and an XP-algorithm is the best that can be achieved. Next, we introduce such an XP-algorithm. The results are, like the rest of this section, from [GKR21a].

Let $G=(V,E)$ be a digraph which is given by some directed clique-width k-expression X. For some vertex set $V'\subseteq V$, we define reach(V') as the set of all pairs (a,b) such that there is a vertex $u\in V'$ labeled by a and there is a vertex $v\in V'$ labeled by b and v is reachable from u in $G[V']$. In the following we use a slightly different notation for the creation of a new vertex v with label a for a directed clique-width expression: We denote this by $a(v)$.

Within a construction of a digraph by directed clique-width operations only the edge insertion operation can change the reachability between the present vertices. Next, we show which acyclic sets remain acyclic when performing an edge insertion operation and how the reachability information of these sets have to be updated due to the edge insertion operation.

Lemma 7.6.11. *Let $G=(V,E)$ be a vertex labeled digraph defined by some directed clique-width k-expression X, $a\neq b$, $a,b\in\{1,\dots,k\}$, and $V'\subseteq V$ be an acyclic set in G. Then, vertex set V' remains acyclic in digraph$(\alpha_{a,b}(X))$ if and only if $(b,a)\notin$ reach(V').*

Proof. If $(b,a)\in$ reach(V'), then we know that in digraph(X) there is a vertex y labeled by a which is reachable by a vertex x labeled by b. That is, in digraph(X) there is a directed path P from x to y. The edge insertion $\alpha_{a,b}$ leads to the arc (y,x) which together with path P brings us to a cycle in digraph$(\alpha_{a,b}(X))$. If $(b,a)\notin$ reach(V') and $V'\subseteq V$ is an acyclic set in digraph(X), then there is a topological ordering of digraph$(X)[V']$ such that every vertex labeled by a is before every vertex labeled by b in the ordering. The same ordering is a topological ordering for digraph$(\alpha_{a,b}(X))[V']$ which implies that V' remains acyclic for digraph$(\alpha_{a,b}(X))$. □

Lemma 7.6.12. *Let $G=(V,E)$ be a vertex labeled digraph defined by some directed clique-width k-expression X, $a\neq b$, $a,b\in\{1,\dots,k\}$, $V'\subseteq V$ be an acyclic set in G, and $(b,a)\notin$ reach(V'). Then, reach(V') for digraph$(\alpha_{a,b}(X))$ can be obtained from reach(V') for digraph(X) as follows:*

- *For every pair $(x,a)\in$ reach(V') and every pair $(b,y)\in$ reach(V'), we extend reach(V') by (x,y).*

Proof. Let R_1 be the set reach(V') for digraph(X), R_2 be the set reach(V') for digraph$(\alpha_{a,b}(X))$, and R be the set of pairs constructed in the lemma starting with reach(V') for digraph(X). Then, it holds $R_1\subseteq R$. Furthermore, the rule given in the lemma obviously puts feasible pairs into reach(V') which implies $R\subseteq R_2$. It remains to show $R_2\subseteq R$. Let $(c,d)\in R_2$ then, if $(c,d)\in R_1$ then also $(c,d)\in R$ as mentioned above. Thus, let $(c,d)\notin R_1$, which implies that there is a vertex $u\in V'$ labeled by c and a vertex $v\in V'$ labeled by d and v is reachable from u in digraph$(\alpha_{a,b}(X))$. Since digraph(X) is a spanning subdigraph of digraph$(\alpha_{a,b}(X))$ and the labels of the vertices are not changed by the performed edge insertion operation, there is a non-empty set $V_u\subseteq V'$ of vertices labeled by c and there is a non-empty set $V_v\subseteq V'$ of vertices labeled by d and no vertex of V_v is reachable from a vertex of V_u in digraph(X). By the definition of the edge insertion operation we know that in digraph(X) there is a vertex u' labeled by a and a vertex V' labeled by b such that u' is reachable from u and v is reachable from v'. Thus, it holds that $(c,a) \in R_1$ and $(b,d) \in R_1$. Our rule given in the statement leads to $(c,d) \in R$. □

We use the notion of a *multi set*, i.e., a set that may have several equal elements. For a multi set with elements $x_1,\dots,x_n$ we write $\mathcal{M} = \langle x_1,\dots,x_n\rangle$. There is no order on the elements of $\mathcal{M}$. The number how often an element x occurs in $\mathcal{M}$ is denoted by $\psi(\mathcal{M},x)$. Two multi sets $\mathcal{M}_1$ and $\mathcal{M}_2$ are *equal* if for each element $x \in \mathcal{M}_1 \cup \mathcal{M}_2$, $\psi(\mathcal{M}_1,x) = \psi(\mathcal{M}_2,x)$, otherwise they are called *different*. The empty multi set is denoted by $\langle\rangle$.

For a disjoint partition of the vertex set V into acyclic sets $V_1,\dots,V_s$, let $\mathcal{M}$ be the multi set $\langle \text{reach}(V_1),\dots,\text{reach}(V_s)\rangle$. Let $F(X)$ be the set of all mutually different multi sets $\mathcal{M}$ for all disjoint partitions of vertex set V into acyclic sets. Every multi set in $F(X)$ consists of nonempty subsets of $\{1,\dots,k\}\times\{1,\dots,k\}$. Each subset can occur 0 times and not more than $|V|$ times. Thus, $F(X)$ has at most

$$(|V|+1)^{2^{k^2}-1} \in |V|^{2^{O(k^2)}}$$

mutually different multi sets and is polynomially bounded in the size of X.

In order to give a dynamic programming solution along the recursive structure of a directed clique-width k-expression, we show how to compute $F(a(v))$, $F(X\oplus Y)$ from $F(X)$ and $F(Y)$, as well as $F(\alpha_{a,b}(X))$ and $F(\rho_{a\to b}(X))$ from $F(X)$.

Lemma 7.6.13. *Let $a,b \in \{1,\dots,k\}$, $a \neq b$.*

1. *$F(a(v)) = \{\langle\{(a,a)\}\rangle\}$.*

2. *Starting with set $D = \{\langle\rangle\}\times F(X)\times F(Y)$ extend D by all triples that can be obtained from some triple $(\mathcal{M},\mathcal{M}',\mathcal{M}'') \in D$ by removing a set L' from $\mathcal{M}'$ or a set L'' from $\mathcal{M}''$ and inserting it into $\mathcal{M}$, or by removing both sets and inserting $L'\cup L''$ into $\mathcal{M}$. Finally, we choose $F(X\oplus Y) = \{\mathcal{M} \mid (\mathcal{M},\langle\rangle,\langle\rangle) \in D\}$.*

3. *$F(\alpha_{a,b}(X))$ can be obtained from $F(X)$ as follows. First, we remove from $F(X)$ all multi sets $\langle L_1,\dots,L_s\rangle$ such that $(b,a) \in L_t$ for some $1 \le t \le s$. Afterwards, we modify every remaining multi set $\langle L_1,\dots,L_s\rangle$ in $F(X)$ as follows:*

   - *For every L_i which contains a pair (x,a) and a pair (b,y), we extend L_i by (x,y).*

4. *$F(\rho_{a\to b}(X)) = \{\langle\rho_{a\to b}(L_1),\dots,\rho_{a\to b}(L_s)\rangle \mid \langle L_1,\dots,L_s\rangle \in F(X)\}$,
   where we use $\rho_{a\to b}(L_i) = \{(\rho_{a\to b}(c),\rho_{a\to b}(d)) \mid (c,d)\in L_i\}$ and $\rho_{a\to b}(c) = b$, if $c = a$, and $\rho_{a\to b}(c) = c$, if $c \neq a$.*

Proof. 1. In digraph$(a(v))$ there is exactly one vertex v labeled by a and thus, the only partition of V into one acyclic set of the vertex set of digraph$(a(v))$ is $\{v\}$. The corresponding multi set is $\langle\text{reach}(\{v\})\rangle = \langle\{(a,a)\}\rangle$.

2. $F(X\oplus Y) \subseteq \{\mathcal{M} \mid (\mathcal{M},\langle\rangle,\langle\rangle) \in D\}$:
   Every acyclic set of digraph$(X\oplus Y)$ is either an acyclic set in digraph(X) or an acyclic set in digraph(Y) or is the union of two acyclic sets from digraph(X) and digraph(Y).

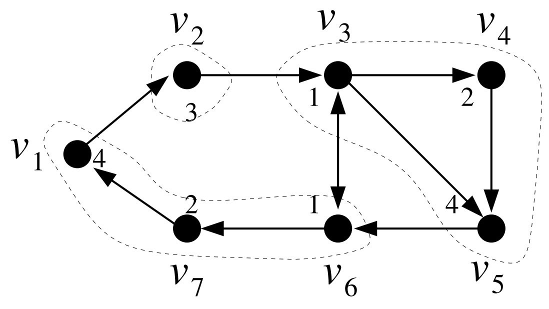


Figure 7.3: Digraph from Example 7.6.15. The dashed lines indicate a partition of the vertex set into three acyclic sets. The small numbers at the vertices represent their labels.

All three possibilities are considered when computing $\{\mathcal{M} \mid (\mathcal{M}, \langle\rangle, \langle\rangle) \in D\}$ from $F(X)$ and $F(Y)$.
$F(X \oplus Y) \supseteq \{\mathcal{M} \mid (\mathcal{M}, \langle\rangle, \langle\rangle) \in D\}$:
Since the operation $\oplus$ does not create any new edges, the acyclic sets from digraph(X) and from digraph(Y) as well as the union of acyclic sets from digraph(X) and digraph(Y) remain acyclic sets for digraph$(X \oplus Y)$.

3. By Lemma 7.6.11 we have to remove all multi sets $\langle L_1, \dots, L_s\rangle$ from $F(X)$ for which holds that $(b,a) \in L_t$ for some $1 \leq t \leq s$. The remaining multi sets are updated correctly by Lemma 7.6.12.

4. In digraph(X) there is a vertex labeled by d which is reachable from a vertex labeled with c if and only if in digraph$(\rho_{a\to b}(X))$ there is a vertex labeled by $\rho_{a\to b}(d)$ which is reachable from a vertex labeled with $\rho_{a\to b}(c)$.

This shows the statement of the lemma. □

Since every possible coloring of G is realized in the set $F(X)$, where X is a directed clique-width k-expression for G, it is easy to find a minimum coloring for G.

Corollary 7.6.14. *Let $G = (V,E)$ be a digraph given by a directed clique-width k-expression X. There is a partition of V into r acyclic sets if and only if there is some $\mathcal{M} \in F(X)$ consisting of r sets of label pairs.*

Example 7.6.15. We consider the digraph $G = (V,E)$ in Figure 7.3. The given partition into three acyclic sets $V = V_1 \cup V_2 \cup V_3$, where $V_1 = \{v_1, v_6, v_7\}$, $V_2 = \{v_2\}$ and $V_3 = \{v_3, v_4, v_5\}$ leads to the multi set $\mathcal{M} = \langle \text{reach}(V_1), \text{reach}(V_2), \text{reach}(V_3)\rangle$, where $\text{reach}(V_1) = \text{reach}(V_3) = \{(1,1),(2,2),(4,4),(1,2),(2,4),(1,4)\}$ and $\text{reach}(V_2) = \{(3,3)\}$.

Theorem 7.6.16. *The Dichromatic number problem on digraphs on n vertices given by a directed clique-width k-expression can be solved in $n^{2^{O(k^2)}}$ time.*

Proof. Let $G = (V,E)$ be a digraph of directed clique-width at most k and T be a k-expression-tree for G with root w. For some vertex u of T we denote by T_u the subtree rooted at u and X_u

the k-expression defined by T_u. In order to solve the Dichromatic number problem for G, we traverse k-expression-tree T into bottom-up order. For every vertex u of T we compute $F(X_u)$ following the rules given in Lemma 7.6.13. By Corollary 7.6.14 we can solve our problem by $F(X_w) = F(X)$.

Our rules given Lemma 7.6.13 show the following running times. For every $v \in V$ and $a \in \{1,\dots,k\}$ set $F(a(v))$ can be computed in $O(1)$. The set $F(X \oplus Y)$ can be computed in time $(n+1)^{3(2^{k^2}-1)} \in n^{2^{O(k^2)}}$ from $F(X)$ and $F(Y)$. The sets $F(\alpha_{a,b}(X))$ and $F(\rho_{a\to b}(X))$ can be computed in time $(n+1)^{2^{k^2}-1} \in n^{2^{O(k^2)}}$ from $F(X)$.

In order to bound the number and order of operations within directed clique-width expressions, we can use the normal form for clique-width expressions defined in [EGW03]. The proof of Theorem 4.2 in [EGW03] shows that also for directed clique-width expression X, we can assume that for every subexpression, after a disjoint union operation first there is a sequence of edge insertion operations followed by a sequence of relabeling operations, i.e., between two disjoint union operations there is no relabeling before an edge insertion. Since there are n leaves in T, we have $n-1$ disjoint union operations, at most $(n-1)\cdot(k-1)$ relabeling operations, and at most $(n-1)\cdot k(k-1)$ edge insertion operations. This leads to an overall running time of $n^{2^{O(k^2)}}$. □

The running time shown in Theorem 7.6.16 leads to the following result.

Corollary 7.6.17. *The Dichromatic number problem is in XP when parameterized by directed clique-width.*

Up to now there are only very few digraph classes for which we can compute a directed clique-width expression in polynomial time. This holds for directed co-graphs, digraphs of bounded directed modular width, and orientations of trees. For such classes we can apply the result of Theorem 7.6.16. In order to find directed clique-width expressions for general digraphs one can use results on the related parameter bi-rank-width [KR13]. By [BJG18, Lemma 9.9.12] we can use approximate directed clique-width expressions obtained from rank-decomposition with the drawback of a single-exponential blow-up on the parameter.

Next, we give a lower bound for the running time of parameterized algorithms for Dichromatic number problem parameterized by the directed clique-width.

Corollary 7.6.18. *The Dichromatic number problem on digraphs on n vertices parameterized by the directed clique-width k cannot be solved in time $n^{2^{o(k)}}$, unless ETH fails.*

Proof. In order to show the statement we apply the following lower bound for the Chromatic number problem parameterized by clique-width given in [FGL$^+$18]. Any algorithm for the Chromatic Number problem parameterized by clique-width with running in $n^{2^{o(k)}}$ would disprove the Exponential Time Hypothesis. By Observation 7.6.2 and since for every undirected graph G its clique-width equals the directed clique-width of $\overleftrightarrow{G}$ [GWY16], any algorithm for the Dichromatic number problem parameterized by directed clique-width can be used to solve the Chromatic number problem parameterized by clique-width. □

In order to show fixed parameter tractability for DCN_r w.r.t. the parameter directed clique-width one can use its definability within monadic second order logic (MSO). We restrict to MSO_1-logic, which allows propositional logic, variables for vertices and vertex sets of digraphs, the predicate $arc(u,v)$ for arcs of digraphs, and quantification over vertices and vertex sets [CE12]. In [GHK$^+$14, Theorem 4.2] it has been shown that for every integer k and MSO_1 formula ψ, every ψ-LinEMSO$_1$ optimization problem (see [GHK$^+$14]) is fixed-parameter tractable on digraphs of clique-width k w.r.t. the parameters k and length of the formula $|\psi|$. Next, we will apply this result to DCN.

Theorem 7.6.19. *The Dichromatic number problem is in FPT when parameterized by directed clique-width and r.*

Proof. Let $G=(V,E)$ be a digraph. We can define DCN_r by an MSO_1 formula

$$\psi = \exists V_1,\ldots,V_r : \left(\mathrm{Partition}(V,V_1,\ldots,V_r) \wedge \bigwedge_{1\le i\le r} \mathrm{Acyclic}(V_i)\right)$$

with

$$\begin{aligned}\mathrm{Partition}(V,V_1,\ldots,V_r) \quad = \quad & \forall v\in V : (\textstyle\bigvee_{1\le i\le r} v\in V_i)\wedge \\ & \nexists v\in V : (\textstyle\bigvee_{i\ne j,\ 1\le i,j\le r}(v\in V_i \wedge v\in V_j))\end{aligned}$$

and

$$\mathrm{Acyclic}(V_i) \quad = \quad \forall V'\subseteq V_i, V'\ne\emptyset : \exists v\in V'(\mathrm{outdegree}(v)=0 \vee \mathrm{outdegree}(v)\ge 2)$$

For the correctness we note the following: For every induced cycle V' in G it holds that for every vertex $v\in V'$ we have $\mathrm{outdegree}(v)=1$ in G. This does not hold for non-induced cycles. But since for every cycle V'' in G there is a subset $V'\subseteq V''$, such that $G[V']$ is a cycle, we can verify by $\mathrm{Acyclic}(V_i)$ whether $G[V_i]$ is acyclic. Since it holds that $|\psi|\in O(r)$, the statement follows by the result of [GHK$^+$14] stated above. □

Corollary 7.6.20. *For every integer r the r-Dichromatic number problem is in FPT when parameterized by directed clique-width.*

7.7 Conclusions and Outlook

This part is taken from [LGK21]. The bound of 7 for the oriented chromatic number and the oriented chromatic index of series-parallel digraphs can be followed in different ways. These bounds are tight even for series-parallel digraphs. As well we can get linear time solutions for the oriented chromatic number and the oriented chromatic index of series-parallel digraphs. The existence of graph classes of arbitrary large vertex degree but bounded oriented chromatic index, such as msp-digraph and esp-digraphs, implies that Vizings Theorem [Viz64] can not be carried over to the oriented chromatic index.

In future work we analyze the existence of polynomial or even linear time algorithms for computing the oriented chromatic index on msp-digraphs. Furthermore, it remains open whether it is possible to compute oriented chromatic index and oriented chromatic number of orientations of series-parallel graphs efficiently which would lead to generalizations of Theorem 7.5.9 and Theorem 7.3.27.

The following part is taken from [GKR21a]. The methods presented in Subsection 7.6.2 allow us to compute the dichromatic number on directed co-graphs in linear time and on graph classes of bounded directed clique-width in polynomial time.

The shown parameterized solutions of Corollary 7.6.17 and Theorem 7.6.19 also hold for any parameter which is larger or equal than directed clique-width, such as the parameter directed modular width [SW20] (which even allows an FPT-algorithm by [SW19, SW20]) and directed linear clique-width [GR19a].

Further, the hardness result of Corollary 7.6.10 rules out FPT-algorithms for the Dichromatic number problem parameterized by width parameters which can be bounded by directed clique-width. Among these are the clique-width and rank-width of the underlying undirected graph, which also have been considered in [Gan09] on the Oriented chromatic number problem.

From a parameterized point of view width parameters are so-called structural parameters, which are measuring the difficulty of decomposing a graph into a special tree-structure. Beside these, the standard parameter, i.e., the threshold value given in the instance, is well studied. Unfortunately, for the Dichromatic number problem the standard parameter is the number of necessary colors r and does even not allow an XP-algorithm, since DCN_2 is NP-complete [MSW19]. A positive result can be obtained for parameter "number of vertices" n. Since integer linear programming is fixed-parameter tractable for the parameter "number of variables" [Len83] the existence of an integer program for DCN using $O(n^2)$ variables implies an FPT-algorithm for parameter n, see [GKR20a].

It remains to verify whether the running time of our XP-algorithm for DCN can be improved to $n^{2^{O(k)}}$, which is possible for the Chromatic number problem by [EGW01]. Further, it remains open whether the hardness of Corollary 7.6.10 also holds for special digraph classes and for directed linear clique-width [GR19a]. Additionally, the existence of an FPT-algorithm for DCN_r w.r.t. parameter clique-width of the underlying undirected graph is open.

8 Conclusions and Outlook

We conclude with a short summary of this work and give some outlook to further possible follow-up research and open questions.

We presented different classes of digraphs and characterized them by different properties. Further, we showed how to use these properties such as for example bounded directed clique-width. By introducing the new class of twin-dh digraphs we have opened up some possibilities for further research. Multiple problems were shown to be not NP-hard anymore when restricted to directed co-graphs. Since twin-dh digraphs are a superclass of directed co-graphs there could be a possibility to expand these results. More precisely, twin-dh digraphs are also a subclass of extended directed co-graphs. Thus, for further research especially those problems are interesting to consider on this digraph class, which are easily solvable on directed co-graphs but still NP-hard on extended directed co-graphs. These could be problems, where we need to know about the connectivity between several vertices. This neighborhood information is still given in twin-dh digraphs, while it is missing in extended directed co-graphs. In a di-co-tree for an extended directed co-graph this explicit neighborhood information is lost. On this topic, it is also an open problem whether an extended directed co-graph can be recognized in linear time and whether we can build its ex-di-co-tree in polynomial or even linear time for any digraph. In order to generalize twin-dh digraphs, the very recent work of [BKTW20] seems to give a promising approach with their definition of twin-width. For future work it might also be interesting to investigate problems which are solvable on undirected distance-hereditary graphs on the corresponding digraphs. Accordingly, we could study, for example, whether the (directed) Steiner tree problem can also be solved on twin-dh digraphs.

A question still open is the hardness of the directed path-width problem on semicomplete digraphs. Although there are several FPT-algorithms solving this problem, there is still no proof that the problem is NP-hard on semicomplete digraphs at all. Considering directed graph parameters there is also the open Conjecture from Hunter and Kreutzer for general digraphs, namely that Kelly-width and the DAG-width of a digraph are equivalent within a constant factor [HK08]. From the results about twin-dh digraphs and directed co-graphs at least we know that DAG-width bounds Kelly-width of a digraph which belongs to one of these classes.

The pseudo-polynomial solutions for the subset sum problems with (weak) digraph constraints are given for directed co-graphs and minimal series-parallel digraphs. Thus, research on other superclasses of directed co-graphs or minimal series-parallel digraphs could be interesting, e.g., on extended directed co-graphs. Moreover, one could look at more general digraph classes of bounded width. In addition, there are other NP-hard digraph problems that can be considered on various recursive digraph classes. Furthermore, it seems likely that there are possibilities of constructing a polynomial or even linear time algorithm for the oriented chromatic index problem on minimal series-parallel digraphs. To sum up, research in the field of digraphs is still very young and offers many open questions and opportunities for new results in the future.

9 Bibliography

References

[AACKS14] A.K. Abu-Affash, P. Carmi, M.J. Katz, and M. Segal. The euclidean bottleneck steiner path problem and other applications of (α,β)-pair decomposition. *Discrete & Computational Geometry*, 51(1):1–23, 2014.

[ACP87] S. Arnborg, D.G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal of Algebraic and Discrete Methods*, 8(2):277–284, 1987.

[AGU72] A.V. Aho, M.R. Garey, and J.D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.

[AH15] S.D. Andres and W. Hochstättler. Perfect digraphs. *Journal of Graph Theory*, 79(1):21–29, 2015.

[AHU74] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Company, Massachusetts, 1974.

[AKK$^+$15] S.A. Amiri, L. Kaiser, S. Kreutzer, R. Rabinovich, and S. Siebertz. Graph Searching Games and Width Measures for Directed Graphs. In *32nd International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 30 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34–47. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.

[AKK17] I. Adler, M.M. Kanté, and O. Kwon. Linear rank-width of distance-hereditary graphs i. a polynomial-time algorithm. *Algorithmica*, 78(1):342–377, 2017.

[AKR16] S. A. Amiri, S. Kreutzer, and R. Rabinovich. DAG-width is PSPACE-complete. *Theoretical Computer Science*, 655:78–89, 2016.

[ALLM16] K. Agrawal, J. Li, K. Lu, and B. Moseley. Scheduling parallel dag jobs online to minimize average flow time. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, page 176–189, 2016.

[Bar06] J. Barát. Directed pathwidth and monotonicity in digraph searching. *Graphs and Combinatorics*, 22:161–172, 2006.

[BCKN15] H.L. Bodlaender, M. Cygan, S. Kratsch, and J. Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243:86–111, 2015.

[BdGR97] D. Bechet, P. de Groote, and C. Retoré. A complete axiomatisation of the inclusion of series-parallel partial orders. In *Rewriting Techniques and Applications*, volume 1232 of *LNCS*, pages 230–240. Springer-Verlag, 1997.

[BDH$^+$12] D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, and J. Obdrzálek. The dag-width of directed graphs. *Journal of Combinatorial Theory, Series B*, 102(4):900–923, 2012.

[BDHK06] D. Berwanger, A. Dawar, P. Hunter, and S. Kreutzer. The dag-width and parity games. In *Proceedings of the Annual Symposium of Theoretical Aspects of Computer Science (STACS)*, volume 3884 of *LNCS*, pages 524–536. Springer-Verlag, 2006.

[BDK00] H.J. Broersma, E. Dahlhaus, and T. Kloks. A linear time algorithm for minimum fill-in and treewidth for distance hereditary graphs. *Discrete Applied Mathematics*, 99(1-3):367–400, 2000.

[BFJ$^+$04] D. Bokal, G. Fijavz, M. Juvan, P.M. Kayll, and B. Mohar. The circular chromatic number of a digraph. *Journal of Graph Theory*, 46(3):227–240, 2004.

[BHW11] G. Borradailea, B. Heeringa, and G. Wilfong. The 1-neighbour knapsack problem. In *Proceedings of International Workshop on Combinatorial Algorithms (IWOCA)*, volume 7056 of *LNCS*, pages 71–84. Springer-Verlag, 2011.

[BHW12] G. Borradailea, B. Heeringa, and G. Wilfong. The knapsack problem with neighbour constraints. *Journal of Discrete Algorithms*, 16:224–235, 2012.

[BJG09] J. Bang-Jensen and G. Gutin. *Digraphs. Theory, Algorithms and Applications*. Springer-Verlag, Berlin, 2009.

[BJG18] J. Bang-Jensen and G. Gutin, editors. *Classes of Directed Graphs*. Springer-Verlag, Berlin, 2018.

[BJM14] J. Bang-Jensen and A. Maddaloni. Arc-disjoint paths in decomposable digraphs. *Journal of Graph Theory*, 77:89–110, 2014.

[BJT92] J. Bang-Jensen and C. Thomassen. A polynomial algorithm for the 2-path problem for semicomplete digraphs. *SIAM Journal on Discrete Mathematics*, 5(3):366–376, 1992.

[BKK95] H.L. Bodlaender, T. Kloks, and D. Kratsch. Treewidth and pathwidth of permutation graphs. *SIAM Journal on Discrete Mathematics*, 8(4):606–616, 1995.

[BKTW20] E. Bonnet, E. J. Kim, S. Thomassé, and R. Watrigant. Twin-width I: tractable FO model checking. *CoRR*, abs/2004.14789, 2020.

[BLS99] A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 1999.

[BM86] H.-J. Bandelt and H.M. Mulder. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41:182–208, 1986.

[BM93] H.L. Bodlaender and R.H. Möhring. The pathwidth and treewidth of cographs. *SIAM J. Disc. Math.*, 6(2):181–188, 1993.

[Bod96] H.L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[Bod98] H.L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998.

[Boe18] D. Boeckner. Oriented threshold graphs. *Australasian Journal of Combinatorics*, 71(1):43–53, 2018.

[BPT09] R. B. Borie, R. G. Parker, and C. A. Tovey. Solving problems on recursively constructed graphs. 41(1), 2009.

[Bys04] J. M. Byskov. Enumerating maximal independent sets with applications to graph colouring. *Operations Research Letters*, 32(6):547–556, 2004.

[CD06] J.-F. Culus and M. Demange. Oriented coloring: Complexity and approximation. In *Proceedings of the Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume 3831 of *LNCS*, pages 226–236. Springer-Verlag, 2006.

[CE12] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic. A Language-Theoretic Approach.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, 2012.

[CH77] V. Chvátal and P.L. Hammer. Aggregation of inequalities in integer programming. *Annals of Discrete Math.*, 1:145–162, 1977.

[Chv84] V. Chvátal. Perfectly ordered graphs. In C. Berge and V. Chvátal, editors, *Topics on Perfect Graphs*, volume 88 of *North-Holland Mathematics Studies*, pages 63–65. North-Holland, 1984.

[CL18] A. Custic and S. Lendl. On streaming algorithms for the steiner cycle and path cover problem on interval graphs and falling platforms in video games. *ACM Computing Research Repository (CoRR)*, abs/1802.08577:9 pages, 2018.

[CLMS14] B. Cloteaux, M.D. LaMar, E. Moseman, and J. Shook. Threshold Digraphs. *J Res Natl Inst Stand Technol*, 119:227–234, 2014.

[CLMS19] V. Campos, R. Lopes, A.K. Maia, and I. Sau. Adapting the directed grid theorem into an fpt algorithm. *Electronic Notes in Theoretical Computer Science*, 346:229–240, 2019.

[CLSB81] D.G. Corneil, H. Lerchs, and L. Stewart-Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3:163–174, 1981.

[CMR00] B. Courcelle, J.A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

[CMZ12] M. Chimani, P. Mutzel, and B. Zey. Improved steiner tree algorithms for bounded treewidth. *Journal of Discrete Algorithms*, 16:67–78, 2012.

[CO00] B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101:77–114, 2000.

[Coh68] R.S. Cohen. Transition graphs and the star height problem. In *Proceedings of the 9th Annual Symposium on Switching and Automata Theory*, pages 383–394. IEEE Computer Society, 1968.

[Cou94] B. Courcelle. The monadic second-order logic of graphs VI: On several representations of graphs by relational structures. *Discrete Applied Mathematics*, 54:117–149, 1994.

[Cou18] B. Courcelle. From tree-decompositions to clique-width terms. *Discrete Applied Mathematics*, 248:125 – 144, 2018.

[CP06] C. Crespelle and C. Paul. Fully dynamic recognition algorithm and certificate for directed cographs. *Discrete Applied Mathematics*, 154(12):1722–1741, 2006.

[CR05] D.G. Corneil and U. Rotics. On the relationship between clique-width and treewidth. *SIAM Journal on Computing*, 4:825–847, 2005.

[CS11] M. Chudnovsky and P.D. Seymour. A well-quasi-order for tournaments. *Journal of Combinatorial Theory, Series B*, 101(1):47–53, 2011.

[CSS15] M. Chudnovsky, A. Scot, and P.D. Seymour. Disjoint paths in tournaments. *Advances in Mathematics*, 270:582–597, 2015.

[Dai80] D.P. Dailey. Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete. *Discrete Mathematics*, 30(3):289–293, 1980.

[DES14] M. Dehmer and F. Emmert-Streib, editors. *Quantitative Graph Theory: Mathematical Foundations and Applications*. Crc Pr Inc, New York, 2014.

[DF13] R.G. Downey and M.R. Fellows. *Fundamentals of Parameterized Complexity*. Springer-Verlag, New York, 2013.

[DHP01] G. Damiand, M. Habib, and C. Paul. A simple paradigm for graph recognition: application to cographs and distance hereditary graphs. *Theoretical Computer Science*, 263(1-2):99–111, 2001.

[DOPS20] J. Dybizbański, P. Ochem, A. Pinlou, and A. Szepietowski. Oriented cliques and colorings of graphs with low maximum degree. *Discrete Mathematics*, 343(5):111829, 2020.

[DS14] J. Dybizbański and A. Szepietowski. The oriented chromatic number of Halin graphs. *Information Processing Letters*, 114(1-2):45–49, 2014.

[Duf19] C. Duffy. A note on colourings of connected oriented cubic graphs. *ACM Computing Research Repository (CoRR)*, abs/1908.02883:8 pages, 2019.

[dWELS02] D. de Werra, C. Eisenbeis, S. Lelait, and E. Stöhr. Circular-arc graph coloring: On chords and circuits in the meeting graph. *European Journal of Operational Research*, 136(3):483 – 500, 2002.

[Egg63] L.E. Eggan. Transition graphs and the star height of regular events. *Michigan Math. J.*, 10:385–397, 1963.

[EGW01] W. Espelage, F. Gurski, and E. Wanke. How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In *Proceedings of Graph-Theoretical Concepts in Computer Science (WG)*, volume 2204 of *LNCS*, pages 117–128. Springer-Verlag, 2001.

[EGW03] W. Espelage, F. Gurski, and E. Wanke. Deciding clique-width for graphs of bounded tree-width. *Journal of Graph Algorithms and Applications - Special Issue of JGAA on WADS 2001*, 7(2):141–180, 2003.

[EL89] P. Eades and X. Lin. How to draw a directed graph. *[Proceedings] 1989 IEEE Workshop on Visual Languages*, pages 13–17, 1989.

[Epp92] D. Eppstein. Parallel recognition of series-parallel graphs. *Information and Computation*, 98(1):41–55, 1992.

[EST94] J.A. Ellis, I.H. Sudborough, and J.S. Turner. The vertex separation and search number of a graph. *Information and Computation*, 113(1):50–79, 1994.

[FGL$^+$18] F.V. Fomin, P. Golovach, D. Lokshtanov, S. Saurabh, and M. Zehavi. Cliquewidth III: The Odd Case of Graph Coloring Parameterized by Cliquewidth. *ACM Transactions on Algorithms*, 15(1):9:1–9:27, 2018.

[FHM03] T. Feder, P. Hell, and B. Mohar. Acyclic homomorphisms and circular colorings of digraphs. *SIAM Journal on Discrete Mathematics*, 17(1):161–163, 2003.

[FHP19] F. Foucaud, S. Heydarshahi, and A. Parreau. Domination and location in twin-free digraphs. *CoRR*, abs/1910.05311, 2019.

[FP13] F.V. Fomin and M. Pilipczuk. Jungles, bundles, and fixed parameter tractability. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 396–413. ACM-SIAM, 2013.

[FP19] F.V. Fomin and M. Pilipczuk. On width measures and topological problems on semi-complete digraphs. *Journal of Combinatorial Theory, Series B*, 138:78–165, 2019.

[FS13] A. Fradkin and P.D. Seymour. Tournament pathwidth and topological containment. *Journal of Combinatorial Theory, Series B*, 103:374–384, 2013.

[Gal14] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 296–303. ACM, 2014.

[Gan09] R. Ganian. The parameterized complexity of oriented colouring. In *Proceedings of Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, MEMICS*, volume 13 of *OASICS*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.

[GGK22] St. Goebbels, F. Gurski, and D. Komander. The knapsack problem with special neighbor constraints. *Mathematical Methods of Operations Research*, 2022. to appear.

[GHK$^+$09] R. Ganian, P. Hlinený, J. Kneis, A. Langer, J. Obdrzálek, and P. Rossmanith. On digraph width measures in parameterized algorithmics. In *Proceedings of the International Symposium on Parameterized and Exact Computation*, volume 5917 of *LNCS*, pages 185–197. Springer-Verlag, 2009.

[GHK$^+$14] R. Ganian, P. Hlinený, J. Kneis, A. Langer, J. Obdrzálek, and P. Rossmanith. Digraph width measures in parameterized algorithmics. *Discrete Applied Mathematics*, 168:88–107, 2014.

[GHK$^+$16] R. Ganian, P. Hlinený, J. Kneis, D. Meisters, J. Obdrzálek, P. Rossmanith, and S. Sikdar. Are there any good digraph width measures? *Journal of Combinatorial Theory, Series B*, 116:250–286, 2016.

[GHK$^+$20a] F. Gurski, S. Hoffmann, D. Komander, C. Rehs, J. Rethmann, and E. Wanke. Computing Directed Steiner Path Covers for Directed Co-Graphs. In *Proceedings of the Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume 12011 of *LNCS*, pages 556–565. Springer-Verlag, 2020.

[GHK$^+$20b] F. Gurski, S. Hoffmann, D. Komander, C. Rehs, J. Rethmann, and E. Wanke. Exact solutions for the steiner path problem on special graph classes. In *Operations Research Proceedings (OR 2019), Selected Papers*, pages 331–338. Springer-Verlag, 2020.

[GHO13] R. Ganian, P. Hlinený, and J. Obdrzálek. A unified approach to polynomial algorithms on graphs of bounded (bi-)rank-width. *European Journal of Combinatorics*, 34(3):680–701, 2013.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, 1979.

[GKL20] F. Gurski, D. Komander, and M. Lindemann. Oriented coloring of msp-digraphs and oriented co-graphs. In *Proceedings of the International Conference on Combinatorial Optimization and Applications (COCOA)*, volume 12577 of *LNCS*, pages 743–758. Springer-Verlag, 2020.

[GKL21a] F. Gurski, D. Komander, and M. Lindemann. Efficient computation of the oriented chromatic number of recursively defined digraphs. *Theoretical Computer Science*, 890:16–35, 2021.

[GKL21b] F. Gurski, D. Komander, and M. Lindemann. Homomorphisms to digraphs with large girth and oriented colorings of minimal series-parallel digraphs. In *Proceedings of the International Workshop on Algorithms and Computation (WALCOM)*, volume 12635 of *LNCS*, pages 182–194. Springer-Verlag, 2021.

[GKR19a] F. Gurski, D. Komander, and C. Rehs. Characterizations for special directed co-graphs. In *Proceedings of the International Conference on Combinatorial Optimization and Applications (COCOA)*, volume 11949 of *LNCS*, pages 252–264. Springer-Verlag, 2019.

[GKR19b] F. Gurski, D. Komander, and C. Rehs. Computing digraph width measures on directed co-graphs. In *Proceedings of International Symposium on Fundamentals of Computation Theory (FCT)*, volume 11651 of *LNCS*, pages 292–305. Springer-Verlag, 2019.

[GKR19c] F. Gurski, D. Komander, and C. Rehs. Oriented coloring on recursively defined digraphs. *Algorithms*, 12(4):87, 2019.

[GKR20a] F. Gurski, D. Komander, and C. Rehs. Acyclic coloring of special digraphs. *ACM Computing Research Repository (CoRR)*, abs/2006.13911:16 pages, 2020.

[GKR20b] F. Gurski, D. Komander, and C. Rehs. Solutions for subset sum problems with special digraph constraints. *Mathematical Methods of Operations Research*, 92(2):401–433, 2020.

[GKR20c] F. Gurski, D. Komander, and C. Rehs. Subset sum problems with special digraph constraints. In *Operations Research Proceedings (OR 2019), Selected Papers*, pages 339–346. Springer-Verlag, 2020.

[GKR21a] F. Gurski, D. Komander, and C. Rehs. Acyclic coloring parameterized by directed clique-width. In *Proceedings of the International Conference on Algorithms and Discrete Applied Mathematics (CALDAM)*, volume 12601 of *LNCS*, pages 95–108. Springer-Verlag, 2021.

[GKR21b] F. Gurski, D. Komander, and C. Rehs. How to compute digraph width measures on directed co-graphs. *Theoretical Computer Science*, 855:161–185, 2021.

[GKR21c] F. Gurski, D. Komander, and C. Rehs. On characterizations for subclasses of directed co-graphs. *Journal of Combinatorial Optimization*, 41(1):234–266, 2021.

[GKR$^+$22] F. Gurski, D. Komander, C. Rehs, J. Rethmann, and E. Wanke. Computing directed steiner path covers. *Journal of Combinatorial Optimization*, 2022. to appear.

[GKRW21] F. Gurski, D. Komander, C. Rehs, and S. Widerrecht. Directed width parameters on semicomplete digraphs. In *Proceedings of the International Conference on Combinatorial Optimization and Applications (COCOA)*, volume 13135 of *LNCS*, pages 615–628. Springer-Verlag, 2021.

[GMT18] L. Gourvès, J. Monnot, and L. Tlilane. Subset sum problems with digraph constraints. *Journal of Combinatorial Optimization*, 36(3):937–964, 2018.

[Gol78] M.C. Golumbic. Trivially perfect graphs. *Discrete Mathematics*, 24:105–107, 1978.

[Gol80] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[Gou12] R. Gould. *Graph Theory*. Dover Publications Inc., New York, NY, USA, 2012.

[GR99] M.C. Golumbic and U. Rotics. On the clique-width of some perfect graph classes. In *Proceedings of Graph-Theoretical Concepts in Computer Science (WG)*, volume 1665 of *LNCS*, pages 135–147. Springer-Verlag, 1999.

[GR00] M.C. Golumbic and U. Rotics. On the clique-width of some perfect graph classes. *International Journal of Foundations of Computer Science*, 11(3):423–443, 2000.

[GR18] F. Gurski and C. Rehs. Directed path-width and directed tree-width of directed co-graphs. In *Proceedings of the International Conference on Computing and Combinatorics (COCOON)*, volume 10976 of *LNCS*, pages 255–267. Springer-Verlag, 2018.

[GR19a] F. Gurski and C. Rehs. Comparing linear width parameters for directed graphs. *Theory of Computing Systems*, 63(6):1358–1387, 2019.

[GR19b] F. Gurski and C. Rehs. Forbidden directed minors, directed path-width and directed tree-width of tree-like digraphs. In *Proceedings of the Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume 11376 of *LNCS*, pages 234–246. Springer-Verlag, 2019.

[GRR18] F. Gurski, C. Rehs, and J. Rethmann. Directed pathwidth of sequence digraphs. In *Proceedings of the International Conference on Combinatorial Optimization and Applications (COCOA)*, volume 11346 of *LNCS*, pages 79–93. Springer-Verlag, 2018.

[Gru08] H. Gruber. Digraph complexity measures and applications in formal language theory. In *Proceedings of MEMICS'08*, pages 60–67, 2008.

[Gru12] H. Gruber. Digraph complexity measures and applications in formal language theory. *Discrete Mathematics and Theoretical Computer Science*, 14(2):189–204, 2012.

[Gur14] F. Gurski. Efficient binary linear programming formulations for boolean functions. *Statistics, Optimization and Information Computing*, 2(4):274–279, 2014.

[Gur17] F. Gurski. Dynamic programming algorithms on directed cographs. *Statistics, Optimization and Information Computing*, 5:35–44, 2017.

[Gus93] J. Gusted. On the pathwidth of chordal graphs. *Discrete Applied Mathematics*, 45(3):233–248, 1993.

[GW00] F. Gurski and E. Wanke. The tree-width of clique-width bounded graphs without $K_{n,n}$. In *Proceedings of Graph-Theoretical Concepts in Computer Science (WG)*, volume 1938 of *LNCS*, pages 196–205. Springer-Verlag, 2000.

[GW05] F. Gurski and E. Wanke. On the relationship between NLC-width and linear NLC-width. *Theoretical Computer Science*, 347(1-2):76–89, 2005.

[GWY16] F. Gurski, E. Wanke, and E. Yilmaz. Directed NLC-width. *Theoretical Computer Science*, 616:1–17, 2016.

[GY02] G. Z. Gutin and A. Yeo. Orientations of digraphs almost preserving diameter. *Discrete Applied Mathematics*, 121(1-3):129–138, 2002.

[HK08] P. Hunter and S. Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. *Theoretical Computer Science*, 399(3):206–219, 2008.

[HKdW97] P. Hansen, J. Kuplinsky, and D. de Werra. Mixed graph coloring. *Mathematical Methods of Operations Research*, 45:145–160, 1997.

[HMP11] P. Heggernes, D. Meister, and C. Papadopoulos. Graphs of linear clique-width at most 3. *Theoretical Computer Science*, 412(39):5466–5486, 2011.

[HN60] F. Harary and R.Z. Norman. Some properties of line digraphs. *Rend. Circ. Mat. Palermo*, 9(2):161–168, 1960.

[Hoà94] C.T. Hoàng. Efficient algorithms for minimum weighted colouring of some classes of perfect graphs. *Discrete Applied Mathematics*, 55:133–143, 1994.

[How77] E. Howorka. A characterization of distance-hereditary graphs. *The Quarterly Journal of Mathematics Ser. 2*, 28:417–420, 1977.

[HY87] X. He and Y. Yesha. Parallel recognition and decomposition of two terminal series parallel graphs. *Information and Computation*, 75:15–38, 1987.

[JKT21] L. Jaffke, O. Kwon, and J. A. Telle. Classes of intersection digraphs with good algorithmic properties. *ACM Computing Research Repository (CoRR)*, abs/2105.01413, 2021.

[JN83] D.S. Johnson and K.A. Niemi. On knapsacks, partitions, and a new dynamic programming technique for trees. *Mathematics of Operations Research*, 8(1):1–14, 1983.

[JP04] K. Jansen and L. Porkolab. Preemptive scheduling with dedicated processors: Applications of fractional graph coloring. *Journal of Scheduling*, 7:35–48, 2004.

[JRST01a] T. Johnson, N. Robertson, P.D. Seymour, and R. Thomas. Addentum to ”Directed tree-width”, 2001.

[JRST01b] T. Johnson, N. Robertson, P.D. Seymour, and R. Thomas. Directed tree-width. *Journal of Combinatorial Theory, Series B*, 82:138–155, 2001.

[Jun78] H.A. Jung. On a class of posets and the corresponding comparability graphs. *Journal of Combinatorial Theory, Series B*, 24:125–133, 1978.

[Kar72] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103, New York, 1972. Plenum Press.

[KBMK93] T. Kloks, H. Bodlaender, H. Müller, and D. Kratsch. Computing treewidth and minimum fill-in: All you need are the minimal separators. In *Proceedings of the Annual European Symposium on Algorithms (ESA)*, volume 726 of *LNCS*, pages 260–271. Springer-Verlag, 1993.

[KF79] T. Kashiwabara and T. Fujisawa. NP-completeness of the problem of finding a minimum-clique-number interval graph containing a given graph as a subgraph. In *Proceedings of the International Symposium on Circuits and Systems*, pages 657–660, 1979.

[KKK$^+$16] K. Kitsunai, Y. Kobayashi, K. Komuro, H. Tamaki, and T. Tano. Computing directed pathwidth in $O(1.89^n)$ time. *Algorithmica*, 75:138–157, 2016.

[KKT15] K. Kitsunai, Y. Kobayashi, and H. Tamaki. On the pathwidth of almost semi-complete digraphs. In *Proceedings of the Annual European Symposium on Algorithms (ESA)*, volume 9294 of *LNCS*, pages 816–827. Springer-Verlag, 2015.

[KL15] S. Kitaev and V. Lozin. *Words and Graphs*. Springer-Verlag, Berlin, 2015.

[KN09] B. Karrer and M. E. J. Newman. Random graph models for directed acyclic networks. *Phys. Rev. E*, 80:046110, 2009.

[Kob15] Y. Kobayashi. Computing the pathwidth of directed graphs with small vertex cover. *Information Processing Letters*, 115(2):310–312, 2015.

[Kom19] D. Komander. Characterization and algorithmic use of directed cographs. Master-Thesis, Heinrich-Heine Universität, Düsseldorf, Germany, 2019.

[KP04] H. Kellerer and U. Pferschy. A new fully polynomial time approximation scheme for the knapsack problem. *Journal of Combinatorial Optimization*, 8:5–11, 2004.

[KPP10] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer-Verlag, Berlin, 2010.

[KR09] M. Kanté and M. Rao. Directed rank-width and displit decomposition. In *Proceedings of Graph-Theoretical Concepts in Computer Science (WG)*, volume 5911 of *LNCS*, pages 214–225. Springer-Verlag, 2009.

[KR13] M. Kanté and M. Rao. The rank-width of edge-coloured graphs. *Theory Comput. Syst.*, 52(4):599–644, 2013.

[KR21] D. Komander and C. Rehs. Twin-distance-hereditary digraphs. *ACM Computing Research Repository (CoRR)*, abs/2112.04183:19 pages, 2021.

[KS15] I. Kim and P.D. Seymour. Tournament minors. *Journal of Combinatorial Theory, Series B*, 112(C):138–153, 2015.

[KSZ97] A.V. Kostochka, E. Sopena, and X. Zhu. Acyclic and oriented chromatic numbers of graphs. *Journal of Graph Theory*, 24(4):331–340, 1997.

[KZ15] S. Kintali and Q. Zhang. Forbidden directed minors and directed pathwidth. Research Report, 2015.

[Law76] E.L. Lawler. Graphical algorithms and their complexity. *Math. Centre Tracts*, 81:3–32, 1976.

[Len83] H.W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[Ler71] H. Lerchs. On cliques and kernels. Technical report, Dept. of Comput. Sci, Univ. of Toronto, 1971.

[LGK21] M. Lindemann, F. Gurski, and D. Komander. Oriented vertex and arc coloring of edge series-parallel digraphs (Abstract). International Conference on Operations Research (OR 2021), 2021.

[LM17] Z. Li and B. Mohar. Planar digraphs of digirth four are 2-colorable. *SIAM J. Discrete Math.*, 31:2201–2205, 2017.

[LOP95] R. Lin, S. Olariu, and G. Pruesse. An optimal path cover algorithm for cographs. *Comput. Math. Appl.*, 30:75–83, 1995.

[LS10] M. Lätsch and R. Schrader. Distance-hereditary digraphs. *Journal of Discrete Algorithms*, 8(2):231–240, 2010.

[Mar13] T.H. Marshall. Homomorphism bounds for oriented planar graphs of given minimum girth. *Graphs and Combin.*, 29:1489–1499, 2013.

[McN69] R. McNaughton. The loop complexity of regular events. *Information Sciences*, 1(3):305–328, 1969.

[MJV13] S.S. Moharana, A. Joshi, and S. Vijay. Steiner path for trees. *International Journal of Computer Applications*, 76(5):11–14, 2013.

[Moh03] B. Mohar. Circular colorings of edge-weighted graphs. *Journal of Graph Theory*, 43(2):107–116, 2003.

[MP95] N.V.R. Mahadev and U.N. Peled. *Threshold Graphs and Related Topics*. Annals of Discrete Math. 56. Elsevier, North-Holland, 1995.

[MS77] C.L. Monma and J.B. Sidney. A general algorithm for optimal job sequencing with series-parallel constraints. *Math. Oper. Res.*, 4:215–224, 1977.

[MS88] B. Monien and I.H. Sudborough. Min cut is NP-complete for edge weighted trees. *Theoretical Computer Science*, 58:209–229, 1988.

[MSW19] M.G. Millani, R. Steiner, and S. Wiederrecht. Colouring non-even digraphs. *ACM Computing Research Repository (CoRR)*, abs/1903.02872:37 pages, 2019.

[Nag12] H. Nagamochi. Linear layouts in submodular systems. In *Proceedings of the International Symposium on Algorithms and Computation*, volume 7676 of *LNCS*, pages 475–484. Springer-Verlag, 2012.

[NdM06] J. Nešetřil and P.O. de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Comb.*, 27:1022–1041, 2006.

[NL82] V. Neumann-Lara. The dichromatic number of a digraph. *Journal of Combinatorial Theory, Series B*, 33(2):265–270, 1982.

[NP11] S.D. Nikolopoulos and C. Papadopoulos. A simple linear-time recognition algorithm for weakly quasi-threshold graphs. *Graphs and Combinatorics*, 27(4):557–565, 2011.

[Obd06] J. Obdrzálek. Dag-width: Connectivity measure for directed graphs. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 814–821. ACM-SIAM, 2006.

[OPS08] P. Ochem, A. Pinlou, and E. Sopena. On the oriented chromatic index of oriented graphs. *Journal of Graph Theory*, 57(4):313–332, 2008.

[OS06] S. Oum and P.D. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006.

[Oum05] S. Oum. Rank-width and vertex-minors. *Journal of Combinatorial Theory, Series B*, 95:79–100, 2005.

[PS06] A. Pinlou and E. Sopena. Oriented vertex and arc coloring of outerplanar graphs. *Information Processing Letters*, 100:97–104, 2006.

[Ree99] B. Reed. Introducing directed tree width. *Electronic Notes in Discrete Mathematics*, 3:222–229, 1999.

[Ren86] F. Rendl. Quadratic assignment problems on series-parallel digraphs. *Z. Oper. Res. Ser. A-B*, 30(3):A161–A173, 1986.

[Ret98] C. Retoré. Pomset logic as a calculus of directed cographs. In *Proceedings of the Fourth Roma Workshop: Dynamic perspectives in Logic and Linguistics*, pages 221–247. CLUEB, 1998.

[Rig51] J. Riguet. Les relations de ferrers. *C.R. Acad. Sci. Paris*, 232:1729–1730, 1951.

[RS83] N. Robertson and P.D. Seymour. Graph minors I. Excluding a forest. *Journal of Combinatorial Theory, Series B*, 35:39–61, 1983.

[RS86] N. Robertson and P.D. Seymour. Graph minors II. Algorithmic aspects of tree width. *Journal of Algorithms*, 7:309–322, 1986.

[RS04] N. Robertson and P.D. Seymour. Graph minors XX. Wagners conjecture. *Journal of Combinatorial Theory, Series B*, 92:325–357, 2004.

[RW90] G. Reich and P. Widmayer. Beyond steiner's problem: a VLSI oriented generalization. In *Proceedings of Graph-Theoretical Concepts in Computer Science (WG)*, volume 411 of *LNCS*, pages 196–210. Springer-Verlag, 1990.

[Sch89] P. Scheffler. *Die Baumweite von Graphen als Mass für die Kompliziertheit algorithmischer Probleme*. Ph. D. thesis, Akademie der Wissenschaften in der DDR, Berlin, 1989.

[Sch21] R. Schrader. Personal communication, 2021.

[Sei74] D. Seinsche. On a property of the class of n-colorable graphs. *Journal of Combinatorial Theory B*, 16:191–193, 1974.

[Sey90] P.D. Seymour. Colouring series-parallel graphs. *Combinatorica*, 10(4):379–392, 1990.

[Sop97] E. Sopena. The chromatic number of oriented graphs. *Journal of Graph Theory*, 25:191–205, 1997.

[ST07] K. Suchan and I. Todinca. Pathwidth of circular-arc graphs. In *Proceedings of Graph-Theoretical Concepts in Computer Science (WG)*, volume 4769 of *LNCS*, pages 258–269. Springer-Verlag, 2007.

[Ste85] G. Steiner. A compact labeling scheme for series-parallel graphs. *Discrete Applied Mathematics*, 11(3):281–297, 1985.

[Sum74] P.D. Sumner. Dacey graphs. *Journal of Aust. Soc.*, 18:492–502, 1974.

[SW19] R. Steiner and S. Wiederrecht. Parameterized algorithms for directed modular width. *ACM Computing Research Repository (CoRR)*, abs/1905.13203:37 pages, 2019.

[SW20] R. Steiner and S. Wiederrecht. Parameterized algorithms for directed modular width. In *Proceedings of the International Conference on Algorithms and Discrete Applied Mathematics (CALDAM)*, volume 12016 of *LNCS*, pages 415–426. Springer-Verlag, 2020.

[Val78] J. Valdes. Parsing flowcharts and series-parallel graphs. Technical Report STAN-CS-78-682, Computer Science Department, Stanford University, Stanford, California, 1978.

[Viz64] V.G. Vizing. On an estimate of the chromatic class of a p-graph. *Metody Diskret. Analiz.*, 3:9–17, 1964.

[VSR$^+$18] M. M. Vazifeh, P. Santi, G. Resta, S. H. Strogatz, and C. Ratti. Addressing the minimum fleet problem in on-demand urban mobility. *Nature*, 557(7706):534–538, 2018.

[VTL82] J. Valdes, R.E. Tarjan, and E.L. Lawler. The recognition of series-parallel digraphs. *SIAM Journal on Computing*, 11:298–313, 1982.

[WC82] J.A. Wald and C.J. Colbourn. Steiner trees in outerplanar graphs. In *Thirteenth Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 15–22, 1982.

[WC83] J.A. Wald and C.J. Colbourn. Steiner trees, partial 2-trees, and minimum IFI networks. *Networks*, 13:159–167, 1983.

[Wie20] S. Wiederrecht. Digraphs of directed treewidth one. *Discrete Mathematics*, 343(12):112124, 2020.

[WY95] J. Westbrook and D. Yan. Approximation algorithms for the class steiner tree problem. Research Report, 1995.

[YC08] B. Yang and Y. Cao. Digraph searching, directed vertex separation and directed pathwidth. *Discrete Applied Mathematics*, 156(10):1822–1837, 2008.

10 Appendix

10.1 Contributions

As it is conventional in our field, the authors are in almost all cases in alphabetical order.

Directed width Parameters on Semicomplete Digraphs [GKRW21]

F. Gurski, D. Komander, C. Rehs, and S. Widerrecht

This paper was written in a close collaboration with my colleague Carolin Rehs. We developed the results and proofs together with help of Frank Gurski. Later in the process Sebastian Wiederrecht joined us and supported the proof of the directed path-width and directed tree-width comparison, as well as he supported us to optimize the result for Kelly-width and directed tree-width.

Twin-Distance-Hereditary Digraphs [KR21]

D. Komander and C. Rehs

This paper bases on discussions about directed twins with Carolin Rehs, Frank Gurski and Van Bang Le. The definitions and the proofs of the further properties of the class were developed in collaboration with Carolin Rehs. The characterizations by forbidden induced subdigraphs was completely my part.

Computing Directed Steiner Path Covers for Directed Co-Graphs [GKR$^+$22]

F. Gurski, D. Komander, C. Rehs, J. Rethmann, and E. Wanke

This is an extended version of the conference paper [GHK$^+$20a]. For this paper I played a role in the developing process of the ideas, which was mostly a collaboration of Egon Wanke, Jochen Rethmann, Frank Gurski and me. The proofs for computing a minimum Steiner path cover on directed co-graphs were developed by me and Jochen Rethmann in close collaboration, supported by Frank Gurski. Additionally, my part was especially giving the idea, how the forbidden subdigraph reduce the problem on directed co-graphs. Jochen Rethmann was responsible for the algorithms (such that I left them out in this work).

Computing Directed Steiner Path Covers [GHK$^+$20a]

F. Gurski, S. Hoffmann, D. Komander, C. Rehs, J. Rethmann, and E. Wanke

For my contribution see above.

Exact Solutions for the Steiner Path Problem on Special Graph Classes [GHK$^+$20b]

F. Gurski, S. Hoffmann, D. Komander, C. Rehs, J. Rethmann, and E. Wanke

This results were developed in discussion with all of the authors on which I took part. I also helped with the final completion of the work.

Oriented Vertex and Arc Coloring of Edge Series-parallel Digraphs [LGK21]

M. Lindemann, F. Gurski, and D. Komander

In this paper I was included in developing the edge coloring results (the oriented chromatic index), while the vertex coloring part was done by Marvin Lindemann and Frank Gurski. I developed the proof of Lemma 7.3.25 with support of Frank Gurski.

Homomorphisms to Digraphs with Large Girth and Oriented Colorings of Minimal series-parallel digraphs [GKL21b]

F. Gurski, D. Komander, and M. Lindemann

In this work I guided the research assistant Marvin Lindemann, who developed the results. Frank Gurski wrote it up and I did the proof reading work.

Acyclic Coloring Parameterized by Directed Clique-width [GKR21a]

F. Gurski, D. Komander, and C. Rehs

This work was mainly grown by Frank Gurski and me, we developed the results and proofs about coloring directed co-graphs together. Frank Gurski developed the main results about the directed clique-width parameterization and proofs and I supported him. I presented the results at the CALDAM 2021 conference.

Oriented Coloring of msp-digraphs and Oriented Co-graphs [GKL20]

F. Gurski, D. Komander, and M. Lindemann

In this paper I have done some important preliminary work for the main theorem 7.3.8, since we started with a hypothesis for the minimum oriented coloring number of 3, which I increased step by step up to 7. The final proof of this result was subsequently constructed by

Marvin Lindemann. Additionally, I was responsible for the proof of the lemma 7.3.10 with support of Frank Gurski.

Efficient Computation of the Oriented Chromatic Number of Recursively Defined Digraphs [GKL21a]

F. Gurski, D. Komander, and M. Lindemann

Same as in the corresponding conference paper [GKL20].

How to Compute Digraph Width Measures on Directed Co-Graphs [GKR21b]

F. Gurski, D. Komander, and C. Rehs

This journal version is an extended summary of the work of papers [GR18] and [GKR19b] which were presented at the conferences FCT 2019 and COCOON 2018. Since there was a major mistake in the proof of the directed tree-width results in paper [GR18], I was part of the intense correction process. We fixed it by developing some lemmata and claims, where I was part of the process of constructing and proving them. The directed path-width results were from [GR18], so this result is the work of Frank Gurski and Carolin Rehs. For the remaining results, which were from [GKR19b], read below.

Solutions for Subset Sum Problems with Special Digraph Constraints [GKR20b]

F. Gurski, D. Komander, and C. Rehs

This is an extended journal version of the conference paper [GKR20c]. The results and proofs of this work have been developed in close collaboration with Frank Gurski. Carolin Rehs only helped with some proof reading.

Subset Sum Problems with Special Digraph Constraints [GKR20c]

F. Gurski, D. Komander, and C. Rehs

The results and proofs of this work have been developed in collaboration with Frank Gurski. Carolin Rehs only helped with some proof reading.

The Knapsack Problem with Special Neighbor Constraints on Directed Cographs [GGK22]

St.J. Goebbels, F. Gurski, and D. Komander

I was part of the early development process of the ideas which based on our paper from [GKR20c].

Characterizations for Special Directed Co-graphs [GKR19a]

F. Gurski, D. Komander, and C. Rehs

The results and proofs in this paper are completely from my master thesis, such that they are not part of this work. Here, we just give a short overview about the results, since they are interesting compared to the further results in this work. Nevertheless, we did some further research on this topic, which is included in this work as well as it is published in the corresponding journal paper [GKR21c].

On Characterizations for Subclasses of Directed Co-Graphs [GKR21c]

F. Gurski, D. Komander, and C. Rehs

This is an extended journal version of our paper [GKR19a]. For the contribution of the results and proofs of the conference version of this paper [GKR19a], see above. The remaining part was proceeded later in close collaboration with Frank Gurski.

Computing Digraph Width Measures on Directed Co-Graphs [GKR19b]

F. Gurski, D. Komander, and C. Rehs

This work was written in collaboration with Carolin Rehs and Frank Gurski. Since the results and proofs are already part of my master thesis, I here only give an overview about the results as they are interesting in the context of the further results in this work.

Oriented Coloring on Recursively Defined Digraphs [GKR19c]

F. Gurski, D. Komander, and C. Rehs

My part of this work was the correctness proof of Theorem 3.3.15 (Theorem 7 in the original work) with some support of Carolin Rehs. Apart from that I only helped with some minor support at the end of the process.

10.2 Affirmation in lieu of an oath

Ich versichere an Eides statt, dass die Dissertation von mir selbständig und ohne unzulässige fremde Hilfe unter Beachtung der „Grundsätze zur Sicherung guter wissenschaftlicher Praxis an der Heinrich-Heine-Universität Düsseldorf“ erstellt worden ist.