algorithms

MDPI

Article

Oriented Coloring on Recursively Defined Digraphs

Frank Gurski *, Dominique Komander and Carolin Rehs

Heinrich Heine University Düsseldorf, Institute of Computer Science, Algorithmics for Hard Problems Group, 40225 Düsseldorf, Germany; dominique.komander@hhu.de (D.K.); carolin.rehs@hhu.de (C.R.)

* Correspondence: frank.gurski@hhu.de

Received: 21 February 2019; Accepted: 22 April 2019; Published: 25 April 2019

Abstract: Coloring is one of the most famous problems in graph theory. The coloring problem on undirected graphs has been well studied, whereas there are very few results for coloring problems on directed graphs. An oriented k-coloring of an oriented graph $G = (V, A)$ is a partition of the vertex set V into k independent sets such that all the arcs linking two of these subsets have the same direction. The oriented chromatic number of an oriented graph G is the smallest k such that G allows an oriented k-coloring. Deciding whether an acyclic digraph allows an oriented 4-coloring is NP-hard. It follows that finding the chromatic number of an oriented graph is an NP-hard problem, too. This motivates to consider the problem on oriented co-graphs. After giving several characterizations for this graph class, we show a linear time algorithm which computes an optimal oriented coloring for an oriented co-graph. We further prove how the oriented chromatic number can be computed for the disjoint union and order composition from the oriented chromatic number of the involved oriented co-graphs. It turns out that within oriented co-graphs the oriented chromatic number is equal to the length of a longest oriented path plus one. We also show that the graph isomorphism problem on oriented co-graphs can be solved in linear time.

Keywords: oriented graphs; oriented co-graphs; oriented coloring; graph isomorphism

1. Introduction

Graph coloring is one of the basic problems in graph theory, which has already been considered in the 19th century. A k-coloring for an undirected graph G is a k-labeling of the vertices of G such that no two adjacent vertices have the same label. The smallest k such that a graph G has a k-coloring is named the chromatic number of G. 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 [1], comparability graphs [2], and co-graphs [3].

Oriented coloring has been introduced much later by Courcelle [4]. One could easily apply the definition of graph coloring to directed graphs, but as this would not take the direction of the arcs into account, this would not be very interesting. For such a definition, the coloring of a directed graph would be the coloring of the underlying undirected graph.

Oriented coloring also considers the direction of the arcs. An oriented k-coloring of an oriented graph $G = (V, A)$ is a partition of the vertex set V into k independent sets, such that all the arcs linking two of these subsets have the same direction. In the oriented chromatic number problem (OCN for short) there is given some oriented graph G and some integer c and one has to decide whether there is an oriented c-coloring for G. Even the restricted problem, when c is constant and does not belong to the input (OCN_c for short), is hard. OCN_4 is NP-complete even for DAGs [5], whereas the undirected problem is easy for trees.

Right now, the definition of oriented coloring is mostly considered for undirected graphs. There the maximum value $\chi_o(G')$ of all possible orientations G' of an undirected graph G is considered.

For several special undirected graph classes the oriented chromatic number has been bounded. Among these are outerplanar graphs [6], planar graphs [7], and Halin graphs [8]. In [9], Ganian has shown an FPT-algorithm for OCN w.r.t. the parameter tree-width (of the underlying undirected graph). Further, he has shown that OCN is DET-hard (DET is the class of decision problems which are reducible in logarithmic space to the problem of computing the determinant of an integer valued $n \times n$-matrix.) for classes of oriented graphs, such that the underlying undirected class has bounded rank-width.

Oriented coloring of special digraph classes seems not to be investigated up to now. The main reason is that the oriented chromatic number of the disjoint union of two oriented graphs can be larger than the maximum oriented chromatic number of the involved graphs (cf. Figure 1 and Example 2). In this paper, we consider the oriented coloring problem restricted to oriented co-graphs, which are obtained from directed co-graphs [10] by omitting the series operation. Oriented co-graphs were already analyzed by Lawler in [11] and [3] (Section 5) using the notation of transitive series parallel (TSP) digraphs. We give several characterizations for oriented co-graphs and show that for oriented co-graphs, the oriented chromatic number of the disjoint union of oriented graphs is equal to the maximum oriented chromatic number of the involved graphs. Further, we show that for every oriented graph the oriented chromatic number of the order composition of oriented graphs is equal to the sum of the oriented chromatic numbers of the involved graphs. To show this, we introduce an algorithm that computes an optimal oriented coloring and thus, the oriented chromatic number of oriented co-graphs in linear time. We also consider the longest oriented path problem on oriented co-graphs. It turns out that within oriented co-graphs the oriented chromatic number is equal to the length of a longest oriented path plus one. Further, we give a linear time algorithm for the graph isomorphism problem on oriented co-graphs. Since oriented co-graphs have a directed NLC-width of one [12], our results provide a useful basis for exploring the complexity of OCN related to width parameters (cf. Section 7).

Figure 1. Special oriented graphs: oriented cycle $\overrightarrow{C_3}$ and transitive tournament $\overrightarrow{T_3}$.

2. Preliminaries

2.1. Graphs and Digraphs

We use the notations of Bang-Jensen and Gutin [13] for graphs and digraphs.

For some given digraph $G = (V, A)$, we define its underlying undirected graph by ignoring the directions of the edges, i.e. $und(G) = (V, \{\{u, v\} \mid (u, v) \in A, u, v \in V\})$ and for some class of digraphs X, let $und(X) = \{und(G) \mid G \in X\}$. For some (di)graph class F we define Free(F) as the set of all (di)graphs G, such that no induced sub(di)graph of G is isomorphic to a member of F.

An *oriented graph* is a digraph with no loops and no opposite arcs. We recall some special oriented graphs. By

$$\overrightarrow{P_n} = (\{v_1, \ldots, v_n\}, \{(v_1, v_2), \ldots, (v_{n-1}, v_n)\}),$$

$n \geq 2$, we denote the oriented path on n vertices, by

$$\overrightarrow{C_n} = (\{v_1, \ldots, v_n\}, \{(v_1, v_2), \ldots, (v_{n-1}, v_n), (v_n, v_1)\}),$$

$n \geq 3$, we denote the oriented cycle on n vertices and by $\overrightarrow{T_n}$ we denote the transitive tournament on n vertices.

2.2. Undirected Co-Graphs

Let $G_1 = (V_1, E_1), \ldots, G_k = (V_k, E_k)$ be k vertex-disjoint graphs.

- The *disjoint union* of $G_1, \ldots, G_k$, denoted by $G_1 \cup \ldots \cup G_k$, is the graph with vertex set $V_1 \cup \ldots \cup V_k$ and edge set $E_1 \cup \ldots \cup E_k$.
- The *join composition* of $G_1, \ldots, G_k$, denoted by $G_1 \times \ldots \times G_k$, is defined by their disjoint union plus all possible edges between vertices of G_i and G_j for all $1 \leq i, j \leq k$, $i \neq j$.

The set of all graphs, which can be defined from a single vertex graph by applying the disjoint union and join composition, is characterized as the set of all co-graphs. It is well known that co-graphs are precisely the P_4-free graphs [3].

2.3. Undirected Graph Coloring

Definition 1 (Graph Coloring). *A k-coloring of a graph $G = (V, E)$ is a mapping $c : V \to \{1, \ldots, k\}$ such that:*

- *$c(u) \neq c(v)$ for every $\{u, v\} \in E$*

The chromatic number of G, denoted by $\chi(G)$, is the smallest k such that G has a k-coloring.

On undirected co-graphs, the graph coloring problem is easy to solve by the following result proven by Corneil et al.:

Lemma 1 ([3]). *Let $G_1, \ldots, G_k$ be k vertex-disjoint graphs.*

1. $\chi(G_1 \cup \ldots \cup G_k) = \max(\chi(G_1), \ldots, \chi(G_k))$
2. $\chi(G_1 \times \ldots \times G_k) = \chi(G_1) + \ldots + \chi(G_k)$

Proposition 1. *Let G be a co-graph. Then, $\chi(G)$ can be computed in linear time.*

The undirected coloring problem, i.e., computing $\chi(G)$, can be solved by an FPT-algorithm w.r.t. the tree-width of the input graph [14]. In contrast, this is not true for clique-width, since it has been shown in [15], that the undirected coloring problem is W[1]-hard w.r.t. the clique-width of the input graph. That is, under reasonable assumptions an XP-algorithm is the best one can hope for. Such algorithms are known, see [16].

2.4. Directed Co-Graphs

The following operations for digraphs have already been considered by Bechet et al. in [10]. Let $G_1 = (V_1, E_1), \ldots, G_k = (V_k, E_k)$ be k vertex-disjoint digraphs.

- The *disjoint union* of $G_1, \ldots, G_k$, denoted by $G_1 \oplus \ldots \oplus G_k$, is the digraph with vertex set $V_1 \cup \ldots \cup V_k$ and arc set $E_1 \cup \ldots \cup E_k$.
- The *series composition* of $G_1, \ldots, G_k$, denoted by $G_1 \otimes \ldots \otimes G_k$, is defined by their disjoint union plus all possible arcs between vertices of G_i and G_j for all $1 \leq i, j \leq k$, $i \neq j$.
- The *order composition* of $G_1, \ldots, G_k$, denoted by $G_1 \oslash \ldots \oslash G_k$, is defined by their disjoint union plus all possible arcs from vertices of G_i to vertices of G_j for all $1 \leq i < j \leq k$.

The set of all digraphs which can be defined by the disjoint union, series composition, and order composition is characterized as the set of all directed co-graphs [10]. Obviously, for every directed co-graph we can define a tree structure, denoted as the *di-co-tree*. The leaves of the di-co-tree represent the vertices of the graph and the inner nodes of the di-co-tree correspond to the operations applied on the subexpressions defined by the subtrees. For every directed co-graph one can construct a di-co-tree in linear time, see [17].

In [18] it is shown that the weak k-linkage problem can be solved in polynomial time for directed co-graphs. By the recursive structure there exist dynamic programming algorithms to compute the size of a largest edgeless subdigraph, the size of a largest subdigraph which is a tournament, the

size of a largest semicomplete subdigraph, and the size of a largest complete subdigraph for every directed co-graph in linear time. Also the Hamiltonian path, Hamiltonian cycle, regular subdigraph, and directed cut problem are polynomial on directed co-graphs [19]. Calculs of directed co-graphs were also considered in connection with pomset logic in [20]. Further, the directed path-width and directed tree-width can be computed in linear time for directed co-graphs [21].

In [17], it has been shown that directed co-graphs can be characterized by the eight forbidden induced subdigraphs shown in Figure 2.

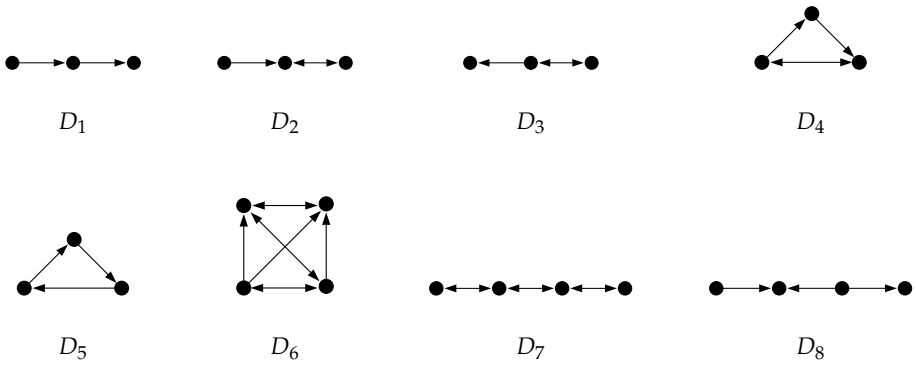

Figure 2. The eight forbidden induced subdigraphs for directed co-graphs.

3. Oriented Co-Graphs

Oriented colorings are defined on oriented graphs, which are digraphs with no bidirected edges. Therefore we introduce oriented co-graphs by omitting the series operation from the definition of directed co-graphs, as given in [10].

Definition 2 (Oriented Co-Graphs)**.** *The class of oriented co-graphs is recursively defined as follows.*

1. *Every digraph on a single vertex* $(\{v\}, \emptyset)$*, denoted by* $\bullet$*, is an oriented co-graph.*
2. *If* $G_1, \ldots, G_k$ *are* k *vertex-disjoint oriented co-graphs, then*
   - *(a)* $G_1 \oplus \ldots \oplus G_k$ *and*
   - *(b)* $G_1 \oslash \ldots \oslash G_k$ *are oriented co-graphs.*

The class of oriented co-graphs was already analyzed by Lawler in [11] and [3] (Section 5) using the notation of *transitive series parallel (TSP) digraphs*. A digraph $G = (V, A)$ is called *transitive*, if for every pair $(u, v) \in A$ and $(v, w) \in A$ of arcs with $u \neq w$ the arc (u, w) also belongs to A.

Theorem 1 ([3])**.** *A graph* G *is a co-graph if and only if there exists an orientation* G' *of* G*, such that* G' *is an oriented co-graph.*

A di-co-tree T is *canonical* if on every path from the root to the leaves of T, the labels disjoint union and order operation strictly alternate. Since the disjoint union $\oplus$ and the order composition $\oslash$ are associative, we always can assume canonical di-co-trees.

Lemma 2. *Let* G *be an oriented co-graph and* T *be a di-co-tree for* G*. Then,* T *can be transformed in linear time into a canonical di-co-tree for* G*.*

The recursive definitions of oriented and undirected co-graphs lead to the following observation.

Observation 1. *For every oriented co-graph* G *the underlying undirected graph* $\mathrm{und}(G)$ *is a co-graph.*

The reverse direction of this observation only holds under certain conditions, see Theorem 2. By $\overleftrightarrow{P_2} = (\{v_1, v_2\}, \{(v_1, v_2), (v_2, v_1)\})$ we denote the complete biorientation of a path on two vertices.

Lemma 3. *Let G be a digraph, such that $G \in Free(\{\overleftrightarrow{P_2}, D_1, D_5\})$. Then, it holds that G is transitive.*

Proof. Let $(u, v), (v, w) \in A$ be two arcs of $G = (V, A)$. Since $G \in \text{Free}(\{\overleftrightarrow{P_2}\})$, we know that $(v, u), (w, v) \notin A$. Further, since $G \in \text{Free}(\{D_1, D_5\})$, we know that u and w are connected either only by $(u, w) \in A$ or by $(u, w) \in A$ and $(w, u) \in A$, which implies that G is transitive. □

Oriented co-graphs can be characterized by forbidden subdigraphs as follows.

Theorem 2. *Let G be a digraph. The following properties are equivalent:*

1. *G is an oriented co-graph.*
2. *$G \in Free(\{D_1, D_5, D_8, \overleftrightarrow{P_2}\})$.*
3. *$G \in Free(\{D_1, D_5, \overleftrightarrow{P_2}\})$ and $und(G) \in Free(\{P_4\})$.*
4. *$G \in Free(\{D_1, D_5, \overleftrightarrow{P_2}\})$ and $und(G)$ is a co-graph.*
5. *G has directed NLC-width 1 and $G \in Free(\{\overleftrightarrow{P_2}\})$.*
6. *G has directed clique-width at most 2 and $G \in Free(\{\overleftrightarrow{P_2}\})$.*
7. *G is transitive and $G \in Free(\{\overleftrightarrow{P_2}, D_8\})$.*

Proof. (1) ⇒ (2) If G is an oriented co-graph, then G is a directed co-graph and by [17] it holds that $G \in \text{Free}(\{D_1, \ldots, D_8\})$. Furthermore, $G \in \text{Free}(\{\overleftrightarrow{P_2}\})$ because of the missing series composition. This leads to $G \in \text{Free}(\{D_1, D_5, D_8, \overleftrightarrow{P_2}\})$. (2) ⇒ (1) If $G \in \text{Free}(\{D_1, D_5, D_8, \overleftrightarrow{P_2}\})$, then $G \in \text{Free}(\{D_1, \ldots, D_8\})$ and is G a directed co-graph. Since $G \in \text{Free}(\{\overleftrightarrow{P_2}\})$, there is no series operation in any construction of G which implies that G is an oriented co-graph. (3) ⇔ (4) Since co-graphs are precisely the P_4-free graphs [3]. (2) ⇒ (7) By Lemma 3. (7) ⇒ (2) If G is transitive, then $G \in \text{Free}(\{D_1, D_5\})$. (1) ⇔ (5) and (1) ⇔ (6) By [12]. (1)&(2) ⇒ (4) By Observation 1. (3) ⇒ (2) If $und(G)$ does not contain a P_4, then G can not contain any orientation of P_4. □

Among others are two subclasses of oriented co-graphs, which will be of interest within our results. By restricting within Definition 2 (2) to $k = 2$ and graph G_1 or G_2 to an edgeless graph or to a single vertex, we obtain the class of all *oriented simple co-graphs* or *oriented threshold graphs*, respectively. The class of oriented threshold graphs has been introduced by Boeckner in [22].

4. Graph Coloring on Recursively Defined Digraphs

4.1. Oriented Graph Coloring Problem

Oriented graph coloring has been introduced by Courcelle [4] in 1994. Most results on this problem consider orientations of undirected graphs. Now, we consider oriented graph coloring on recursively defined oriented graph classes.

Definition 3 (Oriented Graph Coloring [4])**.** *An oriented k-coloring of an oriented graph $G = (V, A)$ is a mapping $c : V \to \{1, \ldots, k\}$, such that:*

- *$c(u) \neq c(v)$ for every $(u, v) \in A$*
- *$c(u) \neq c(y)$ for every two arcs $(u, v) \in A$ and $(x, y) \in A$ with $c(v) = c(x)$*

The oriented chromatic number of G, denoted by $\chi_o(G)$, is the smallest k, such that G has an oriented k-coloring. The vertex sets $V_i = \{v \in V \mid c(v) = i\}$, $1 \leq i \leq k$, divide a partition of V into so called color classes.

For two oriented graphs $G_1 = (V_1, A_1)$ and $G_2 = (V_2, A_2)$ 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 $(u, v) \in A_1$ implies that $(h(u), h(v)) \in A_2$. The oriented

graphs G_1 and G_2 are *homomorphically equivalent*, if there is a homomorphism from G_1 to G_2 and one from G_2 to G_1. 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. This leads to equivalent definitions for oriented coloring and oriented chromatic number. There is an oriented k-coloring of an oriented graph G_1 if and only if there is a homomorphism from G_1 to some oriented graph G_2 on k vertices. That is, the oriented chromatic number of G 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, G_2 can be chosen as a tournament.

Observation 2. *There is an oriented k-coloring of an oriented graph G_1 if and only if there is a homomorphism from G_1 to some tournament G_2 on k vertices. Further, the oriented chromatic number of G is the minimum number of vertices in a tournament G_2, such that there is a homomorphism from G_1 to G_2.*

Lemma 4. *Let G be an oriented graph and H be a subdigraph of G, then $\chi_o(H) \leq \chi_o(G)$.*

Example 1. *For oriented paths and oriented cycles we know: $\chi_o(\overrightarrow{P_2}) = 2$, $\chi_o(\overrightarrow{P_3}) = 3$, $\chi_o(\overrightarrow{C_4}) = 4$, $\chi_o(\overrightarrow{C_5}) = 5$.*

An oriented graph $G = (V, A)$ is an *oriented clique* (*o-clique*) if $\chi_o(G) = |V|$. Thus all graphs given in Example 1 are oriented cliques.

Name Oriented Chromatic Number (OCN)
Instance An oriented graph $G = (V, A)$ and a positive integer $c \leq |V|$.
Question Is there an oriented c-coloring for G?

If c is constant and not part of the input, the corresponding problem is denoted by OCN_c. Even for DAGs OCN_4 is NP-complete [5].

The definition of oriented coloring is also used for undirected graphs. For an undirected graph G the maximum value $\chi_o(G')$ of all possible orientations G' of G is considered. In this sense, every tree has oriented chromatic number at most 3. For several further graph classes there exist bounds on the oriented number. Among these are outerplanar graphs [6], planar graphs [7], and Halin graphs [8].

4.2. Oriented Graph Coloring for Oriented Graphs

Oriented graph coloring has not yet been considered for recursively defined graphs, though it has been analyzed for some graph operations. In this section we show results of oriented graph coloring on some graph operations and provide algorithms for recursively defined oriented graph classes. This will also be useful for the following section.

First, we give some results on the oriented graph coloring for general recursively defined oriented graphs. These results will be very useful to prove our results for oriented co-graphs in the next section.

Lemma 5. *Let $G_1, \ldots, G_k$ be k vertex-disjoint oriented graphs. Then the following equations holds:*

1. $\chi_o(G_1 \oplus \bullet) = \chi_o(G_1)$
2. $\chi_o(G_1 \oplus \ldots \oplus G_k) \geq \max(\chi_o(G_1), \ldots, \chi_o(G_k))$
3. $\chi_o(G_1 \oslash \ldots \oslash G_k) = \chi_o(G_1) + \ldots + \chi_o(G_k)$

Proof.

1. $\chi_o(G_1 \oplus \bullet) \leq \chi_o(G_1)$

   Since no new arcs are inserted G_1 can keep its colors. The added isolated vertex gets a color of G_1 in order to obtain a valid coloring for $G_1 \oplus \bullet$.

   $\chi_o(G_1 \oplus \bullet) \geq \chi_o(G_1)$

This relation holds by Lemma 4, since G_1 is an induced subdigraph of $G_1 \oplus \bullet$.

2. $\chi_o(G_1 \oplus \ldots \oplus G_k) \geq \max(\chi_o(G_1), \ldots, \chi_o(G_k))$

   Since the digraphs $G_1, \ldots, G_k$ are induced subdigraphs of digraph $G_1 \oplus \ldots \oplus G_k$, all values $\chi_o(G_1), \ldots, \chi_o(G_k)$ lead to a lower bound for the number of necessary colors of the combined graph by Lemma 4.

3. $\chi_o(G_1 \oslash \ldots \oslash G_k) \leq \chi_o(G_1) + \ldots + \chi_o(G_k)$

   For $1 \leq i \leq k$ let $G_i = (V_i, A_i)$ and $c_i : V_i \to \{1, \ldots, \chi_o(G_i)\}$ a coloring for G_i. For $G_1 \oslash \ldots \oslash G_k = (V, A)$ we define a mapping $c : V \to \{1, \ldots, \sum_{j=1}^{k} \chi_o(G_i)\}$ as follows.

$$c(v) = \begin{cases} c_1(v) & \text{if } v \in V_{G_1} \\ c_i(v) + \sum_{j=1}^{i-1} \chi_o(G_i) & \text{if } v \in V_{G_i},\ 2 \leq i \leq k. \end{cases}$$

   The mapping c satisfies the definition of an oriented coloring, because no two adjacent vertices from G_i, $1 \leq i \leq k$, have the same color by assumption and by definition of c. For $1 \leq i \neq j \leq k$ a vertex of G_i and a vertex of G_j are always adjacent, but never colored equally by definition of c.

   Further, the arcs between two color classes of every G_i, $1 \leq i \leq k$, have the same direction by definition of c. For $1 \leq i \neq j \leq k$ the arcs between a color class of G_i and a color class of G_j have the same direction by definition of the order operation.

   $\chi_o(G_1 \oslash \ldots \oslash G_k) \geq \chi_o(G_1) + \ldots + \chi_o(G_k)$

   Since every G_i, $1 \leq i \leq k$, is an induced subdigraph of the combined graph, all values $\chi_o(G_1), \ldots, \chi_o(G_k)$ lead to a lower bound for the number of necessary colors of the combined graph by Lemma 4. Further, the order operations implies that for every $1 \leq i \neq j \leq k$ no vertex in G_i can be colored in the same way as a vertex in G_j. Thus, $\chi_o(G_1) + \ldots + \chi_o(G_k)$ leads to a lower bound for the number of necessary colors of the combined graph.

This shows the statements of the lemma. □

By Lemma 5, we can solve oriented coloring for oriented simple co-graphs and thus, also for subclasses, such as oriented threshold graphs and transitive tournaments, in linear time.

Proposition 2. *Let G be an oriented simple co-graph. Then, it holds that $\chi_o(G) = \chi(und(G)) = \omega(und(G))$ and all values can be computed in linear time.*

It is not easy to generalize these results to oriented co-graphs. To do so, we would need to compute the oriented chromatic number of the disjoint union of two oriented co-graphs with at least two vertices. But it is not possible to compute this oriented chromatic number of the disjoint union of general oriented graphs from the oriented chromatic numbers of the involved graphs. In Lemma 5 (2) we only show a lower bound. The following example proves that in general this can not be strengthened to equality.

Example 2. *The two graphs $\overrightarrow{C_3}$ and $\overrightarrow{T_3}$ in Figure 1 have the same oriented chromatic number $\chi_o(\overrightarrow{C_3}) = \chi_o(\overrightarrow{T_3}) = 3$, but their disjoint union needs more colors.*

On the other hand, there are several examples for which the disjoint union does not need more than $\max(\chi_o(G_1), \chi_o(G_2))$ colors, such as the union of two isomorphic oriented graphs. By Theorem 2, we know that $\overrightarrow{T_3}$, shown in Figure 1, is an oriented co-graph, but $\overrightarrow{C_3}$, shown in Figure 1, is not an oriented co-graph. Consequently, the question arises whether oriented coloring could be closed under disjoint union, when restricted to oriented co-graphs.

4.3. Oriented Graph Coloring for Oriented Co-Graphs

In order to solve OCN restricted to oriented co-graphs G we created a procedure, which is shown in Algorithm 1. The method traverses a canonical di-co-tree T for G using a depth-first search, such that for every inner vertex the children are visited from left to right. For every inner vertex u of T, we store two values $\text{in}[u]$ and $\text{out}[u]$. These values ensure that the vertices of G, corresponding to the leaves of the subtree, rooted at u will we labeled by labels ℓ, such that $\text{in}[u] \leq \ell \leq \text{out}[u]$. For every leaf vertex u of T, we additionally store the label of the corresponding vertex of G in $\text{color}[u]$. These values lead to an optimal oriented coloring of G by the next theorem.

Algorithm 1: Computing an oriented coloring for an oriented co-graph.

```
procedure LABEL(G, u, i)
if (u is a leaf of T) {
    color[u] = i; in[u] = i; out[u] = i;
}
else {
    in[u] = i; out[u] = 0;
    for all children v of u from left to right do {
        j =LABEL(G, v, i);
        if (out[u] < j)
            out[u] = j;
        if (u corresponds to a disjoint union)
            i = in[u];
        else        ▶ u corresponds to an order operation
            i = out[v] + 1;
    }
}
return out[u];
```

Theorem 3. *Let G be an oriented co-graph. Then, an optimal oriented coloring for G and $\chi_o(G)$ can be computed in linear time.*

Proof. Let $G = (V, A)$ be an oriented co-graph. Using the method of [17] we can build a di-co-tree T with root r for G in linear time. Further by Lemma 2, we can assume that T is a canonical di-co-tree. For some node u of T we define by T_u the subtree of T which is rooted at u and by G_u the subgraph of G which is defined by T_u. Obviously, for every vertex u of T the tree T_u is a di-co-tree for the digraph G_u which is also an oriented co-graph.

Next, we verify that procedure $\text{LABEL}(G, r, 1)$, shown in Algorithm 1, returns the value $\chi_o(G)$ and computes an oriented coloring for G within array $\text{color}[u]$. Therefore, we recursively show for every vertex u of T that after performing $\text{LABEL}(G, u, i)$ for all leaves u of T_u the value $\text{color}[u]$ leads to an oriented coloring of G_u using the colors $\{i = \text{in}[u], \ldots, \text{out}[u]\}$ (Please note that using colors starting at values greater than 1 is not a contradiction to Definition 3.) and the value $\text{out}[u] - \text{in}[u] + 1$ leads to the oriented chromatic number of G_u.

We distinguish the following three cases depending on the type of operation corresponding to the vertices u of T.

- If u is a leaf of T, then $\text{color}[u] = \text{out}[u] = \text{in}[u]$ by the algorithm leads to an oriented coloring of G_u.

  Further, $\text{out}[u] - \text{in}[u] + 1 = 1$, which obviously corresponds to the oriented chromatic number of G_u.
- Let u be an inner vertex of T which corresponds to an order operation and $u_1, \ldots, u_\ell$ are the children of u in T.

We already know that the oriented colorings of G_{u_i}, $1 \leq i \leq \ell$, are feasible. Further, for $1 \leq i \neq j \leq \ell$, the algorithm's way of working ensures that a vertex from G_{u_i} and a vertex from G_{u_j} are never colored equally in G_u. For $1 \leq i \neq j \leq \ell$, the arcs between a color class of G_{u_i} and a color class of G_{u_j} have the same direction by the definition of the order operation.

By the algorithm, value $\text{out}[u] - \text{in}[u] + 1$ is equal to $\sum_{i=1}^{\ell} \chi_o(G_{u_i})$. By Lemma 5, we conclude that $\text{out}[u] - \text{in}[u] + 1$ is equal to $\chi_o(G_{i_1} \oslash \ldots \oslash G_{i_\ell}) = \chi_o(G_u)$.

- Let u be an inner vertex of T which corresponds to a disjoint union operation and $u_1, \ldots, u_\ell$ are the children of u in T.

  We already know that the oriented colorings of G_{u_i}, $1 \leq i \leq \ell$, are feasible. Since a disjoint union operation does not create any arcs, no two adjacent vertices have the same color in G_u. Further, our method ensures that for every arc (u, v) in G it holds that $\text{color}[u] < \text{color}[v]$. Thus, all arcs between two color classes in G_u have the same direction.

  By the algorithm, value $\text{out}[u] - \text{in}[u] + 1$ is equal to $\max(\chi_o(G_1), \ldots, \chi_o(G_\ell))$. By Lemma 5, we conclude that $\text{out}[u] - \text{in}[u] + 1 \leq \chi_o(G_1 \oplus \ldots \oplus G_\ell) = \chi_o(G_u)$. The relation $\text{out}[u] - \text{in}[u] + 1 \geq \chi_o(G_1 \oplus \ldots \oplus G_\ell) = \chi_o(G_u)$ holds by the feasibility of our oriented coloring.

By applying the invariant for $u = r$, the statements of the theorem follow. □

Example 3. *We illustrate the method given in Algorithm 1 by computing an oriented coloring for the oriented co-graph G, which is given by the canonical di-co-tree T of Figure 3. On the left of each vertex u of T, the values* $\text{in}[u]$ *and* $\text{out}[u]$ *are given. An optimal oriented coloring for G is given in blue letters below the leaves of T. The root r of T leads to* $\chi_o(G) = \text{out}[r] = 5$.

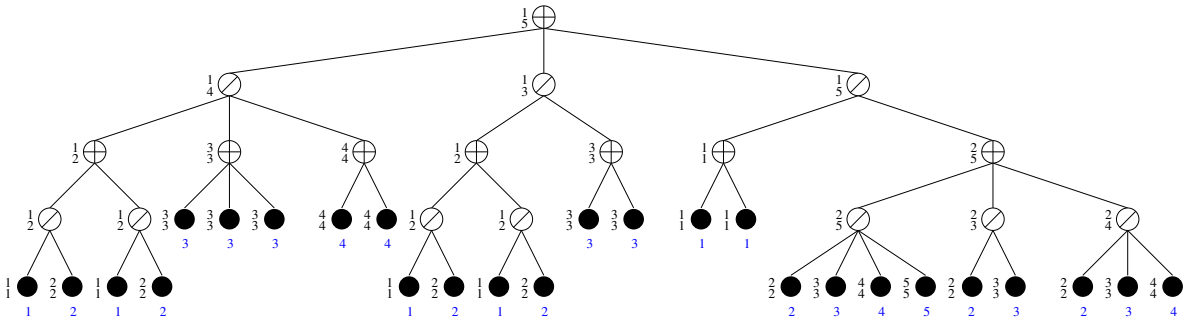

Figure 3. Canonical di-co-tree T for oriented co-graph G.

Next, we can improve the result of Lemma 5 (2) for oriented co-graphs.

Corollary 1. *Let $G_1, \ldots, G_k$ be k vertex-disjoint oriented co-graphs. Then, it holds that*

$$\chi_o(G_1 \oplus \ldots \oplus G_k) = \max(\chi_o(G_1), \ldots, \chi_o(G_k)).$$

Proof. Let $G = G_1 \oplus \ldots \oplus G_k$ be an oriented co-graph and T be a di-co-tree with root r for G. The method given in Algorithm 1 computes an oriented coloring using $\chi_o(G) = \chi_o(G_1 \oplus \ldots \oplus G_k)$ colors. Further, the proof of Theorem 3 shows that $\chi_o(G_1 \oplus \ldots \oplus G_k) = \max(\chi_o(G_1), \ldots, \chi_o(G_k))$. □

Corollary 2. *Let G be an oriented co-graph. The following properties are equivalent:*

1. *G is an oriented clique.*
2. *G has a di-co-tree, which does not use any disjoint union operation.*
3. *G is a transitive tournament.*

Further characterizations for transitive tournaments and oriented co-graphs, which are oriented cliques, can be found in [23] (Chapter 9).

As mentioned in Observation 2, oriented coloring of an oriented graph G can be characterized by the existence of homomorphisms to tournaments. These tournaments are not necessarily transitive and G is not necessarily homomorphically equivalent to some tournament. For oriented co-graphs we can show a deeper result.

Corollary 3. *There is an oriented k-coloring of an oriented co-graph G if and only if there is a homomorphism from G to some transitive tournament $\overrightarrow{T_k}$ on k vertices. Further, the oriented chromatic number of an oriented co-graph G is the minimum number k, such that G is homomorphically equivalent with the transitive tournament $\overrightarrow{T_k}$.*

Proof. Within an oriented co-graph $G = (V, A)$ the color classes $V_1, \ldots, V_k$ of an oriented k-coloring define a transitive tournament $\overrightarrow{T_k} = (\{V_1, \ldots, V_k\}, \{(V_i, V_j) \mid v_i \in V_i, v_j \in V_j, (v_i, v_j) \in A\})$. If $k = \chi_o(G)$, then there is a homomorphism from $\overrightarrow{T_k}$ to G. □

5. Longest Oriented Path for Oriented Graphs

Name Oriented Path (OP)
Instance An oriented graph $G = (V, A)$ and a positive integer $k \leq |V|$.
Question Is there an oriented path of length at least k in G?

We can bound the path length through the oriented chromatic number, when considering oriented co-graphs. Please note that in the subsequent results the oriented path $\overrightarrow{P_{k+1}}$ does not necessarily refer to an induced path (though, its definition implies no chord on the path).

Proposition 3 ([24]). *A directed graph G has a homomorphism to the transitive tournament $\overrightarrow{T_k}$ if and only if there is no homomorphism of the oriented path $\overrightarrow{P_{k+1}}$ to G.*

By Corollary 3 this leads to the next result.

Corollary 4. *The oriented chromatic number of an oriented co-graph G is the minimum number k such that there is no homomorphism of the oriented path $\overrightarrow{P_{k+1}}$ to G.*

In order to compute the length of a longest oriented path $\ell(G)$ for an oriented graph G, we give the next result.

Lemma 6. *Let $G_1, \ldots, G_k$ be k vertex-disjoint oriented graphs.*

1. $\ell(G_1 \oplus \ldots \oplus G_k) = \max(\ell(G_1), \ldots, \ell(G_k))$
2. $\ell(G_1 \oslash \ldots \oslash G_k) = \ell(G_1) + \ldots + \ell(G_k) + k - 1$

Theorem 4. *Let G be an oriented co-graph. Then, the length of a longest oriented path $\ell(G)$ can be computed in linear time.*

Proposition 4. *Let G be an oriented co-graph. Then, it holds that $\ell(G) = \chi_o(G) - 1$.*

Proof. The statement can be shown recursively on the structure of an oriented co-graph G.

- If $G = \bullet$, then it obviously holds $\ell(G) = 0 = \chi_o(G) - 1$.
- If $G = G_1 \oplus \ldots \oplus G_k$

$$\begin{array}{rcll} \ell(G) & = & \ell(G_1 \oplus \ldots \oplus G_k) & \\ & = & \max(\ell(G_1), \ldots, \ell(G_k)) & \text{by Lemma 6} \\ & = & \max(\chi_o(G_1) - 1, \ldots, \chi_o(G_k) - 1) & \text{by induction hypothesis} \\ & = & \max(\chi_o(G_1), \ldots, \chi_o(G_k)) - 1 & \\ & = & \chi_o(G_1 \oplus \ldots \oplus G_k) - 1 & \text{by Lemma 1} \\ & = & \chi_o(G) - 1 & \end{array}$$

- If $G = G_1 \oslash \ldots \oslash G_k$

$$\begin{array}{rcll} \ell(G) & = & \ell(G_1 \oslash \ldots \oslash G_k) & \\ & = & \ell(G_1) + \ldots + \ell(G_k) + k - 1 & \text{by Lemma 6} \\ & = & \chi_o(G_1) - 1 + \ldots + \chi_o(G_k) - 1 + k - 1 & \text{by induction hypothesis} \\ & = & \chi_o(G_1) + \ldots + \chi_o(G_k) - 1 & \\ & = & \chi_o(G_1 \oslash \ldots \oslash G_k) - 1 & \text{by Lemma 1} \\ & = & \chi_o(G) - 1 & \end{array}$$

This shows the statements of the lemma. □

The previous lemma implies that for every oriented co-graph the upper bound of Corollary 4 is strict.

In order to state the next result, let $\omega(G)$ be the number of vertices in a largest clique in graph G.

Corollary 5. *Let G be an oriented co-graph, then $\chi_o(G) = \ell(G) + 1 = \chi(und(G)) = \omega(und(G))$ and all values can be computed in linear time.*

Proof. The first equality holds by Proposition 4 and remaining equality follows by Lemma 1. □

6. Graph Isomorphism for Oriented Co-Graphs

The isomorphism problem for undirected co-graphs has been shown to be solvable in polynomial time in [3]. This result can be improved as follows. Two undirected co-graphs G_1 and G_1 are isomorphic if and only if their canonical co-trees T_1 and T_2 are isomorphic. A canonical co-tree for a co-graph can be determined in linear time. Thus, by applying a linear time isomorphism test for rooted labeled trees (cf. [25], Section 3.2) on canonical co-trees for G_1 and G_2, one can decide in linear time whether G_1 and G_2 are isomorphic.

We consider the corresponding problem for oriented co-graphs.

Name Oriented Co-Graph Isomorphism (OCI)
Instance Two oriented co-graphs $G_1 = (V_1, A_1)$ and $G_2 = (V_2, A_2)$.
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 $(u, v) \in A_1$ if and only if $(b(u), b(v)) \in A_2$?

For oriented co-graphs, the method using an isomorphism test for rooted labeled trees on the co-trees does not work, since the order of the vertices, which are representing order operations in the di-co-tree, must be preserved. The procedure in Algorithm 2 provides a solution for di-co-trees.

Theorem 5. *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 cographs with the corresponding di-co-trees T_1 and T_2 , which can be found in linear time with the method of [17]. Moreover we can assume, that the di-co-trees are

canonical by Lemma 2. 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 2. 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 [25] (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. □

Algorithm 2: 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 ℓ = h downto 0 do
    for all vertices v on level ℓ in T_1 from left to right do
        if (v is a leaf)
            label[v] = 0
        else
            let v_1, ..., 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 ℓ in T_1
    for all vertices v on level ℓ in T_2 from left to right do
        if (v is a leaf)
            label[v] = 0
        else
            let v_1, ..., 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 ℓ in T_2
    sort S_1 to obtain S'_1 and sort S_2 to obtain S'_2
    if (S'_1 ≠ S'_2)
        return false
    let V_ℓ be the set of all vectors on level ℓ in T_1
    find a bijection b : V_ℓ → {1, ..., |V_ℓ|}
    for all vertices v on level ℓ in T_1 do
        label[v] = b(v);
    for all vertices v on level ℓ in T_2 do
        label[v] = b(v);
}
return true;
```

7. Conclusions and Outlook

In this paper, we have considered vertex coloring on oriented graphs. We were able to introduce linear time solutions for the oriented coloring problem, longest oriented path problem and isomorphism problem on oriented co-graphs. Our solutions are based on computations along a (canonical) di-co-tree for the given input co-graphs. Furthermore, it turns out that within oriented co-graphs, the oriented chromatic number is equal to the length of a longest oriented path plus one. This is a quite interesting result, as within undirected co-graphs even for bipartite graphs the path length can not be bounded by the chromatic number.

Further, on oriented co-graphs an independent set of largest size can be computed in the same way as known for undirected co-graphs [3]. Additionally, a tournament subdigraph of largest size and a partition of the vertex set into a minimum number of tournaments can be computed by applying the method for independent set or oriented coloring problem on the reverse input graph.

It remains to consider the existence of an FPT-algorithm for OCN w.r.t. the parameter directed tree-width as given in [26]. Since the directed tree-width of a digraph is always less or equal the undirected tree-width of the corresponding underlying undirected graph [26], the FPT-algorithm of Ganian [9] (see also Section 1) does not imply such a result.

In [9], Ganian has shown that OCN is DET-hard for classes of oriented graphs, such that the underlying undirected class has bounded rank-width. He used a reduction from the isomorphism problem for tournaments, which has been shown to be DET-hard in [27]. The same reduction also works for several linear width parameters, since these can define the disjoint union of two arbitrarily large cliques. Consequently, OCN is DET-hard for classes of oriented graphs, such that the underlying undirected class has linear NLC-width at most 2, linear clique-width at most 3, neighbourhood-width at most 2, or linear rank-width 1. Further, OCN is DET-hard for classes of oriented graphs, such that the underlying undirected class has NLC-width 1 or equivalently clique-width 2. The complexity of OCN on oriented graphs, such that the underlying undirected class has linear NLC-width at most 1 (equivalently neighbourhood-width 1) or linear clique-width at most 2, remains open, since these classes do not contain the disjoint union of two arbitrarily large cliques.

It also remains to generalize the shown results for oriented coloring on oriented graphs of bounded directed clique-width as given in [12,28]. By the existence of a monadic second order logic formula it follows that for every c the problem OCN_c is fixed parameter tractable w.r.t. the parameter directed clique-width.

For the more general problem OCN the existence of an XP-algorithm or even an FPT-algorithm w.r.t. the directed clique-width of the input graph is still open. Since the directed clique-width of a digraph is always greater or equal the undirected clique-width of the corresponding underlying undirected graph [12], the result of Ganian [9] (see also Section 1) does not imply a hardness result.

Author Contributions: Conceptualization, C.R.; Writing—Original Draft Preparation, F.G. and C.R.; Writing—Review & Editing, D.K. and C.R.; Funding Acquisition, F.G.

Funding: This research was funded by the German Research Association (DFG) grant GU 970/7-1.

Acknowledgments: We acknowledge support by the Heinrich Heine University Duesseldorf.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Golumbic, M. *Algorithmic Graph Theory and Perfect Graphs*; Academic Press: Cambridge, MA, USA, 1980.
2. Hoàng, C. Efficient algorithms for minimum weighted coloring of some classes of perfect graphs. *Discret. Appl. Math.* **1994**, *55*, 133–143.
3. Corneil, D.; Lerchs, H.; Stewart-Burlingham, L. Complement Reducible Graphs. *Discret. Appl. Math.* **1981**, *3*, 163–174.
4. Courcelle, B. The monadic second-order logic of graphs VI: On several representations of graphs by relational structures. *Discret. Appl. Math.* **1994**, *54*, 117–149.

5. Culus, J.F.; Demange, M. Oriented Coloring: Complexity and Approximation. In Proceedings of the Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), Merin, Czech Republic, 21–27 January 2006; Springer: Berlin/Heidelberg, Germany, 2006; Volume 3831, pp. 226–236.
6. Sopena, É. The chromatic number of oriented graphs. *J. Graph Theory* **1997**, *25*, 191–205.
7. Marshall, T. Homomorphism bounds for oriented planar graphs of given minimum girth. *Graphs Comb.* **2013**, *29*, 1489–1499.
8. Dybizbański, J.; Szepietowski, A. The oriented chromatic number of Halin graphs. *Inf. Process. Lett.* **2014**, *114*, 45–49.
9. Ganian, R. The Parameterized Complexity of Oriented Coloring. In Proceedings of the Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS), Znojmo, Czech Republic, 13–15 November 2009; Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik: Wadern, Germany, 2009; Volume 13.
10. Bechet, D.; de Groote, P.; Retoré, C. A complete axiomatisation of the inclusion of series-parallel partial orders. In *Rewriting Techniques and Applications*; Springer: Berlin/Heidelberg, Germany, 1997; Volume 1232, pp. 230–240.
11. Lawler, E. Graphical algorithms and their complexity. *Math. Cent. Tracts* **1976**, *81*, 3–32.
12. Gurski, F.; Wanke, E.; Yilmaz, E. Directed NLC-width. *Theor. Comput. Sci.* **2016**, *616*, 1–17.
13. Bang-Jensen, J.; Gutin, G. *Digraphs. Theory, Algorithms and Applications*; Springer: Berlin, Germany, 2009.
14. Gurski, F. A comparison of two approaches for polynomial time algorithms computing basic graph parameters. *arXiv* **2008**, arXiv:0806.4073.
15. Fomin, F.; Golovach, P.; Lokshtanov, D.; Saurabh, S. Clique-width: On the price of generality. In Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), New York, NY, USA, 4–6 January 2009; ACM-SIAM: Philadelphia, PA, USA, 2009; pp. 825–834.
16. Espelage, W.; Gurski, F.; Wanke, E. How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In Proceedings of the Graph-Theoretical Concepts in Computer Science (WG), Boltenhagen, Germany, 14–16 June 2001; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2204, pp. 117–128.
17. Crespelle, C.; Paul, C. Fully dynamic recognition algorithm and certificate for directed cographs. *Discret. Appl. Math.* **2006**, *154*, 1722–1741.
18. Bang-Jensen, J.; Maddaloni, A. Arc-disjoint paths in decomposable digraphs. *J. Graph Theory* **2014**, *77*, 89–110.
19. Gurski, F. Dynamic Programming Algorithms on Directed Cographs. *Stat. Optim. Inf. Comput.* **2017**, *5*, 35–44.
20. Retoré, C. Pomset logic as a calculus of directed cographs. In Proceedings of the Fourth Roma Workshop: Dynamic perspectives in Logic and Linguistics, Roma, Italy, 15–17 October 1997; CLUEB: Bologna, Italy, 1998; pp. 221–247.
21. Gurski, F.; Rehs, C. Directed path-width and directed tree-width of directed co-graphs. In Proceedings of the International Conference on Computing and Combinatorics (COCOON), Qingdao, China, 2–4 July 2018; Springer: Berlin/Heidelberg, Germany, 2018; Volume 10976, pp. 255–267.
22. Boeckner, D. Oriented Threshold Graphs. *Australas. J. Comb.* **2018**, *71*, 43–53.
23. Gould, R. *Graph Theory*; Dover Publications Inc.: New York, NY, USA, 2012.
24. Hell, P.; Nesetril, J. *Graphs and Homomorphisms*; Oxford University Press: New York, NY, USA, 2004.
25. Aho, A.; Hopcroft, J.; Ullman, J. *The Design and Analysis of Computer Algorithms*; Addison-Wesley Publishing Company: Boston, MA, USA, 1974.
26. Johnson, T.; Robertson, N.; Seymour, P.; Thomas, R. Directed Tree-width. *J. Comb. Theory Ser. B* **2001**, *82*, 138–155.
27. Wagner, F. Hardness Results for Tournament Isomorphism and Automorphism. In Proceedings of the Mathematical Foundations of Computer Science (MFCS), Cesky Krumlov, Czech Republic, 27–31 August 2007; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4708, pp. 572–583.
28. Courcelle, B.; Olariu, S. Upper bounds to the clique width of graphs. *Discret. Appl. Math.* **2000**, *101*, 77–114.