A Parameterized Algorithmics Framework for Digraph Degree Sequence Completion Problems

Robert Bredereck, Vincent Froese, Marcel Koseler, Marcelo Garlet Millani, André Nichterlein\textsuperscript{*}, and Rolf Niedermeier

Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany, \{robert.bredereck,vincent.froese,andre.nichterlein,rolf.niedermeier\}@tu-berlin.de

Abstract

There has been intensive work on the parameterized complexity of the typically NP-hard task to edit undirected graphs into graphs fulfilling certain given vertex degree constraints. In this work, seemingly for the first time, we lift the investigations to the case of directed graphs; herein, we focus on arc insertions. To this end, our general two-stage framework consists of efficiently solving a problem-specific number problem transferring the solution to a solution for the graph problem by applying flow computations. In this way, we obtain fixed-parameter tractability and polynomial kernelizability results, with the central parameter being the maximum vertex degree. Although there are certain similarities with the much better studied undirected case, the flow computation used in the directed case seems not to work for the undirected case while \( f \)-factor computations as used in the undirected case seem not to work for the directed case.

1 Introduction

Modeling real-world networks (e.g., communication, ecological, social) often requests directed graphs (digraphs for short). We study a fundamental class of specific “network design” (in the sense of constructing a specific network topology) or “graph realization” problems. Here, our focus is on inserting arcs into a given digraph in order to fulfill some degree constraints. These problems are typically NP-hard, so we choose parameterized algorithm design for identifying relevant tractable special cases. The main parameter we work with is the maximum in- or out-degree of the newly constructed digraph.

To motivate the main digraph arc insertion (or completion) problems we deal with, consider the following three application scenarios.

First, assume we are given a directed network representing a system’s current state. Then, each individual node might have certain desired connection states in terms of the numbers of in- and outgoing arcs which we want to satisfy by adding arcs between the nodes. For instance, in a peer-review network we have an arc from one author reviewing a paper of another author. Depending on research experience, the authors might have different requests with respect to the number of own papers to be reviewed by others and other papers which

\textsuperscript{*}From February 2016 to January 2017 on postdoctoral leave to Durham University (GB), funded by DAAD.
they are reviewing. This leads to the **Digraph Degree Constraint Completion** problem as studied in Section 4.1.

Second, assume now that we have two different data sources: A network which is an incomplete measurement of some unreliable source and the true degree sequence of the target network. The goal is to reconstruct the original network by inserting a given number of arcs such that we obtain the target degree sequence (in a sense, the network matches the given degree sequence). In the presence of labeled input networks this might for example reveal communication patterns between users in social networks. The corresponding problem is called **Digraph Degree Sequence Completion** and studied in Section 4.2.

Third, assume we want to "$k$-anonymize" a social network, that is, after inserting a minimum number of arcs each in- and each out-degree occurs either zero or at least $k$ times. This leads to the **Digraph Degree Anonymity** problem as studied in Section 4.3. All three problems are NP-hard and based on a general framework presented in Section 3, we derive several fixed-parameter tractability results for these, mainly exploiting the parameter "maximum vertex degree".

All three problems are special cases of the **Digraph Degree Constraint Sequence Completion** problem which we will define next. Before doing so, however, we want to go into a little more detail concerning the roots of the underlying graph-theoretic problems studied here. Since early computer science and algorithmic graph theory days, studies on the graph realizability of degree sequences (that is, multisets of positive integers or integer pairs) have played a prominent role, being performed both for undirected graphs [8, 16] as well as digraphs [5, 11, 17, 22]. Lately, the graph modification view gained more and more attention: given a graph, can it be changed by a minimum number of graph modifications such that the resulting graph adheres to specific constraints for its degree sequence? Herein, in the most basic version a degree sequence is a sequence of positive integers specifying (requested) vertex degrees for a fixed ordering of the vertices. Typically, the corresponding computational problems are NP-hard. In recent years, research in this direction focused on undirected graphs [10, 13, 14, 28, 30]. In this work, we investigate the field of parameterized algorithms on digraphs. As already Gutin and Yeo [15] observed, much less is known about the structure of digraphs than of undirected graphs making the design of parameterized algorithms for digraphs more challenging. In particular, we present a general framework for a class of degree sequence problems, focusing on the case of arc insertions (that is, completion problems).

The most general degree completion problem for digraphs we consider in this work is as follows.

**Digraph Degree Constraint Sequence Completion (DDConSeqC)**

**Input:** A digraph $D = (V,A)$, an integer $s$, a "degree list function" $\tau: V \rightarrow 2^{[0,\ldots,r]} \times [0,\ldots,r]$, and a "sequence property" $\Pi$.

**Question:** Is it possible to obtain a digraph $D'$ by adding at most $s$ arcs to $D$ such that the degree sequence of $D'$ fulfills $\Pi$ and $\deg_D(v) \in \tau(v)$ for all $v \in V$?

We emphasize that there are two types of constraints—one for the individual vertices specified by the function $\tau$, and one for the list of degree values as a whole, specified by $\Pi$. For instance, a common $\Pi$ as occurring in the context of data privacy applications is to request that the list is $k$-anonymous, that is, every combination of in- and outdegree that occurs in the list occurs at least $k$ times.

Since DDConSeqC and its special cases as studied here all turn out to be NP-hard [23].

2
a parameterized complexity analysis seems the most natural fit for understanding the computational complexity landscape of these kinds of problems—this has also been observed in the above mentioned studies for the undirected case. Our main findings are mostly on the positive side. That is, although seemingly more intricate to deal with due to the existence of in- and outdegrees, many positive algorithmic results as holding for undirected graphs can also be achieved for digraphs (albeit using significantly different techniques). In particular, we present a maximum-flow-based framework that, together with the identification and solution of certain number problems, helps to derive several fixed-parameter tractability results with respect to the parameter maximum possible in- or outdegree $\Delta^*$ in the resulting digraph. Notably, the corresponding result in the undirected case was based on $f$-factor computations [10] which do not transfer to the directed case, and, vice versa, the flow computation approach we present for the directed case seemingly does not transfer to the undirected case. For special cases of DDCONSEQC, we can move further and even derive some polynomial-size problem kernels, again for the parameter $\Delta^*$.

We consider the parameter $\Delta^*$ for the following reasons. First, it is smaller than $r$, a natural parameter in the input. Second, in combination with $\Pi$, we might get an even smaller upper bound for $\Delta^*$. Third, bounded degree graphs are well studied and our work extends this since we only require $\Delta^*$ to be small, not to be constant. Fourth, in practice, the maximum degree is significantly smaller than the number of vertices: Leskovec and Horvitz [25] studied a huge instant-messaging network (180 million vertices) with maximum degree 600. For the DBLP co-author graph generated in February 2012 containing more than 715,000 vertices one has a maximum degree of 804 and an H-index of 208, that is, there are not more than 208 vertices with degree larger than 208 [19]. Thus, the parameter maximum degree is interesting when studying kernelization as we do. In the context of anonymization we expect that the maximum degree will not increase during the anonymization process [18].

Related Work. Most of the work on graph modification problems for realizing degree constraints has focused on undirected graphs [10, 13, 14, 19, 28, 30]. Closest to our work is the framework for deriving polynomial-size problem kernels for undirected degree sequence completion problems [10], which we complement by our results for digraphs. Generally, we can derive similar results, but the technical details differ quite significantly and the landscape of problems is richer in the directed case. As to digraph modification problems in general, we are aware of surprisingly little work. We mention work studying arc insertion for making a digraph transitive [32] or for making a graph Eulerian [7], both employing the toolbox of parameterized complexity analysis. Somewhat related is also work about the insertion of edges into a mixed graph to satisfy local edge-connectivity constraints [1] or about orienting edges in a partially oriented graph to make it an oriented graph [2].

Our Results. In Section 3, we present our general framework for DDCONSEQC. That is, based on flow computations, in a two-stage approach we show that it is fixed-parameter tractable for the parameter $\Delta^*$. To this end, we identify a pure number problem that needs to be solved in polynomial time and is problem-specific. Next, presenting applications of the framework, in Section 4.1 we show that if there is no constraint $\Pi$ concerning the degree sequence (that is, DIGRAPh DEGREE CONSTRAINT COMPLETION), then one can not only obtain fixed-parameter tractability but also a polynomial-size problem kernel for parameter $\Delta^*$. Then, in Section 4.2 we show the analogous result if there is an exactly specified degree se-
sequence to be fulfilled (Digraph Degree Sequence Completion). Finally, in Section 4.3 we show that if we request the degree sequence to be $k$-anonymous (that is, Digraph Degree Anonymity), then we can at least derive a polynomial-size problem kernel for the combined parameter $(s, \Delta_D)$, where $\Delta_D$ denotes the maximum in- or outdegree of the input graph. Also, we take a first step outlining the limitations of our framework with respect to the parameter $\Delta^*$.

2 Preliminaries

We consider simple digraphs (without multiarcs or self-loops) $D = (V, A)$ with $n := |V|$ and $m := |A|$. For a vertex $v \in V$, $\deg_D^-(v)$ denotes the indegree of $v$, that is, the number of arcs ingoing to $v$. Correspondingly, $\deg_D^+(v)$ denotes the outdegree. We define $\deg_D(v) := (\deg_D^-(v), \deg_D^+(v))$. The set $V(A') := \{v \in V \mid ((v, w) \in A' \lor (w, v) \in A') \land w \in V\}$ contains all vertices incident to an arc in $A'$. For a set of arcs $A' \subseteq V \times V$, $D + A'$ denotes the digraph $(V, A \cup A')$, while $D[A']$ denotes the subdigraph $(V(A'), A')$. Analogously, for a set of vertices $V' \subseteq V$, $D[V']$ denotes the induced subdigraph which only contains the vertices $V'$ and arcs between vertices from $V'$. The set $N_D^+(v) := \{w \in V \mid (v, w) \in A\}$ denotes the set of outneighbors of $v$. Analogously, $N_D^-(v) := \{w \in V \mid (w, v) \in A\}$ denotes the set of inneighbors. Furthermore, we define the maximum indegree $\Delta_D^- := \max_{v \in V} \deg_D^-(v)$, the maximum outdegree $\Delta_D^+ := \max_{v \in V} \deg_D^+(v)$, and $\Delta_D := \max\{\Delta_D^-, \Delta_D^+\}$.

A digraph degree sequence $\sigma = \{(d_1^- , d_1^+), \ldots , (d_n^- , d_n^+)\}$ is a multiset of nonnegative integer tuples. We define $\Delta^- := \max\{d_1^-, \ldots , d_n^-\}$, $\Delta^+ := \max\{d_1^+, \ldots , d_n^+\}$, and $\sigma := \max\{\Delta^-, \Delta^+\}$. For a digraph $D = (\{v_1, \ldots , v_n\}, A)$, we denote by $\sigma(D) := \{\deg_D(v_1), \ldots , \deg_D(v_n)\}$, the digraph degree sequence of $D$. Let $d = (d^-, d^+)$ be a nonnegative integer tuple. For a digraph $D$, the block $B_D(d)$ of degree $d$ is the set of all vertices having degree $d$, formally $B_D(d) := \{v \in V \mid \deg_D(v) = d\}$. We define $\lambda_D(d)$ as the number of vertices in $D$ with degree $d$, that is, $\lambda_D(d) := |B_D(d)|$. Similarly, we define $B_\sigma(t)$ as the multiset of all tuples of type $t$ and $\lambda_\sigma(t)$ as the number of occurrences of the tuple $t$ in the multiset $\sigma$. For two integer tuples $(x_1, y_1)$, $(x_2, y_2)$, we define the sum $(x_1, y_1) + (x_2, y_2) := (x_1 + x_2, y_1 + y_2)$.

Parameterized algorithms is a two-dimensional framework to design algorithms for NP-hard problems. The first part of a parametrized problem instance $(x, k)$ is the classical input $x$, and the second part is the parameter $k$. A parameterized problem is called fixed-parameter tractable if it is solvable in $f(k) \cdot |x|^{O(1)}$ time for some computable function $f$ only depending on $k$. A particularly important case of fixed-parameter tractability is the concept of kernelization. Here, the goal is to map an instance $(x, k)$ to an equivalent instance $(x', k')$ in polynomial time such that $k' \leq k$ and $|x'| \leq g(k)$, where $g$ is some computable function only depending on $k$. The resulting instance $(x', k')$ is called a problem kernel.

3 The Framework

Our goal is to develop a framework for deriving fixed-parameter tractability for a general class of digraph completion problems. To this end, recall our general setting for DDConSeqC which is as follows. We are given a digraph and want to add at most $s$ arcs to it such that the vertices satisfy certain degree constraints $\tau$, and additionally, the degree sequence of the digraph fulfills a certain property $\Pi$. Formally, the sequence property $\Pi$ is given as a function that maps a digraph degree sequence to 1 if the sequence fulfills the property and otherwise
to 0. We restrict ourselves to properties where the corresponding function can be encoded with only polynomially many bits in the number of vertices of the input digraph and can be decided efficiently.\footnote{All specific properties in this work can be easily decided in polynomial time. Indeed, in many cases even fixed-parameter tractability with respect to the maximum integer in the sequence would suffice.} We remark that it is not always the case that there are both vertex degree constraints (\(\tau\)) as well as degree sequence constraints (II) requested. This can be handled by either setting \(\tau\) to the trivial degree list function with \(\tau(v) = \{0, \ldots, n-1\}\) for all \(v \in V\) or setting II to allow all possible degree sequences.

In this section, we show how to derive (under certain conditions) fixed-parameter tractability with respect to the maximum possible in- or outdegree \(\Delta^*\) of the output digraph for DD-ConSeqC. Note that \(\Delta^*\) in general is not known in advance. In practice, we might therefore instead consider upper bounds for \(\Delta^*\) which depend on the given input. For example, we always have \(\Delta_D \leq \Delta^* \leq r \leq \Delta_D + s\) since we are only adding at most \(s\) arcs to \(D\). Clearly, \(\Delta^*\) might also be upper-bounded depending on II (or on the combination of \(r\) and II) in some cases. Our generic framework consists of two main steps: First, we prove fixed-parameter tractability with respect to the combined parameter \((s, \Delta^*)\) in Section 3.1. This step generalizes ideas for the undirected case \cite{10}. Second, we show in Section 3.2 how to upper bound the number \(s\) of arc additions polynomially in \(\Delta^*\) by solving a certain problem specific numerical problem. For this step, we develop a new key argument based on a maximum flow (the undirected case was based on \(f\)-factor arguments).

### 3.1 Fixed-parameter tractability with respect to \((s, \Delta^*)\).

To start with, we show that DD-ConSeqC is fixed-parameter tractable with respect to the combination of the maximum number \(s\) of arcs to insert and the maximum possible in- or outdegree \(\Delta^*\). The basic idea underlying this result is that two vertices \(v\) and \(w\) of the same indegree and outdegree with \(\tau(v) = \tau(w)\) are interchangeable. Accordingly, we will show that it suffices to consider only a bounded number of vertices with the same “degree properties”. In particular, if there is a solution, then there is also a solution that only inserts arcs between this bounded number of vertices. To formalize this idea, we introduce the notion of an \(\alpha\)-block-type set for some positive integer \(\alpha\). An \(\alpha\)-block-type set is a vertex subset that contains \(\alpha\) vertices for each occurring combination of degree and type. The type of a vertex is defined via the number of arcs \(\alpha\) that allows to add to this vertex.

Let \((D, s, \tau, \Pi)\) be a DD-ConSeqC instance. A vertex \(v\) is of type \(t = \{0, \ldots, \Delta^*\} \times \{0, \ldots, \Delta^*\}\) if \(\deg_D(v) + t \in \tau(v)\). The subset of \(V(D)\) containing all vertices of type \(t\) is denoted by \(T_{D,\tau}(t)\). A vertex \(v\) of type \((0,0)\) (that is, \(\deg_D(v) \in \tau(v)\)) is called satisfied. A vertex which is not satisfied is called unsatisfied. The formal definition is as follows.

**Definition 1.** A vertex subset \(C \subseteq V(D)\) containing all unsatisfied vertices is called

- **\(\alpha\)-type set** if for each type \(t \neq (0,0)\) we have \(|C \cap T_{D,\tau}(t)| = \min\{|T_{D,\tau}(t)|, \alpha\}\);

- **\(\alpha\)-block set** if for each degree \(t \in \sigma(D)\) we have \(|C \cap B_D(t)| = \min\{|B_D(t)|, \alpha\}|;

- **\(\alpha\)-block-type set** if for each degree \(t \in \sigma(D)\) and each type \(t' \neq (0,0)\) we have \(|C \cap B_D(t) \cap T_{D,\tau}(t')| = \min\{|B_D(t) \cap T_{D,\tau}(t')|, \alpha\}|.

As a first step, we prove that these sets defined above can be computed efficiently.
Lemma 2. An \( \alpha \)-type / \( \alpha \)-block / \( \alpha \)-block-type set \( C \) as described in Definition 2 can be computed in \( O(m + |\tau| + r^2) / O(m + n + \Delta_D^2) / O(m + |\tau| + \Delta_D^2 + r^2) \) time.

Proof. To compute the \( \alpha \)-block-type set \( C \), we iterate once over the vertex set \( V(D) \). For each \( t \in (\tau(v)) \), we add \( v \) into the set \( X(\deg_D(v), t) \) if \( |X(\deg_D(v), t)| < \alpha \). If \( v \) is unsatisfied, then we also add \( v \) directly to \( C \). In a final step we set \( C := \bigcup_{t \in \tau} X(t', t) \). This can be done in \( O(m + |\tau| + \Delta_D^2 + r^2) \) time. The other two cases of computing an \( \alpha \)-block set and \( \alpha \)-type set can be done in a similar fashion. \( \square \)

We move on to the crucial lemma stating that a solution (if existing) can always be found in between vertices of an \( \alpha \)-block-type set \( C \) given that \( C \) contains “enough” vertices of each degree and type. For DDConSeqC, \( \alpha := 2s(\Delta_D + 1) \) will be enough.

Lemma 3. Let \( (D, s, \tau, \Pi) \) be a DDConSeqC instance and let \( C \) be a \( 2s(\Delta_D + 1) \)-block-type set. If \( (D, s, \tau, \Pi) \) is a yes-instance, then there exists a solution \( A^* \subseteq C^2 \) for \( (D, s, \tau, \Pi) \), that is, \( \{A^*\} \subseteq s \), \( \sigma(D + A^*) \) fulfills \( \Pi \), and \( \deg_{D + A^*}(v) \in \tau(v) \) for all \( v \in V \).

Proof. Let \( A' \subseteq V(D)^2 \setminus A(D) \) be a solution for \( (D, s, \tau, \Pi) \) that minimizes the number of vertices not in \( C \), that is, \( |V(A') \setminus C| \) is minimum. The solution \( A' \) exists since \( (D, s, \tau, \Pi) \) is a yes-instance. If \( V(A') \subseteq C \), then we are done. Hence, we assume that there exists a vertex \( v \) in \( V(D) \setminus C \) which is incident to at least one arc in \( A' \). Let \( V_v^- := \{u \mid (u, v) \in A'\} \) and let \( V_v^+ := \{w \mid (v, w) \in A'\} \) be the set of in- respectively outneighbors of \( v \) in \( A' \). Furthermore, let \( t := \deg_D(v) \) and \( t' := (V_v^+, |V_v^-|) \). Thus, \( v \) has degree \( t \) and is of type \( t' \). By construction of \( C \), it follows that \( |B_D(t) \cap T_{D,\tau}(t')| > 2s(\Delta_D + 1) \).

Now, we claim that there is a vertex \( v^* \in B_D(t) \cap T_{D,\tau}(t') \cap C \setminus V(A') \) such that we can replace \( v \) with \( v^* \) in the solution. More precisely, in all arcs of \( A' \) we want to replace \( v \) by \( v^* \), that is, we obtain a new arc set \( A^* := \{(u, w) \in A' \mid u \neq v \wedge w \neq v\} \cup \{(u, v^*) \mid u \in V_v^- \cup V_v^+ \} \cup \{(v^*, w) \mid w \in V_v^+ \} \). Since we cannot add arcs that are already in the input digraph \( D \), we need that \( N_D(v^*) \cap V_v^- = \emptyset \) and \( N_D(v^*) \cap V_v^+ = \emptyset \). Observe that such a vertex \( v^* \) exists: Since each of the at most \( s \) vertices in \( V_v^+ \cup V_v^- \) has at most \( \Delta_D \) ingoing and \( \Delta_D \) outgoing arcs, it follows that at most \( s \cdot 2\Delta_D \) vertices in \( B_D(t) \cap T_{D,\tau}(t') \cap C \) can have an arc from or to a vertex in \( V_v^+ \cup V_v^- \). Furthermore, since \( |A'| \leq s \), it follows that at most \( 2s - 1 \) vertices in \( B_D(t) \cap T_{D,\tau}(t') \cap C \) are incident to an arc in \( A' \) (the minus one comes from the fact that \( v \) is incident to at least one arc in \( A' \)). By construction of \( C \) it follows that \( |B_D(t) \cap T_{D,\tau}(t') \cap C| = 2s(\Delta_D + 1) > s \cdot 2\Delta_D + 2s - 1 \). Hence, there is at least one vertex \( v^* \in B_D(t) \cap T_{D,\tau}(t') \cap C \) that is not adjacent to any vertex in \( V_v^+ \cup V_v^- \) and not incident to any arc in \( A' \). Thus, we can replace \( v \) by \( v^* \).

We now show that \( A^* \) is still a solution: First, observe that \( \sigma(D + A') = \sigma(D + A^*) \) and, hence, \( \sigma(D + A^*) \) fulfills \( \Pi \). Second, observe that \( \deg_{D + A^*}(v) \in \tau(v) \) since \( v \notin C \) which implies that \( v \) was not unsatisfied. Furthermore \( \deg_{D + A^*}(v) \in \tau(v) \) since \( v^* \) is of type \( t' \). Hence, \( A^* \) is a solution and \( |V(A') \setminus C| > |V(A^*) \setminus C| \), a contradiction to the assumption that \( A' \) was the solution minimizing this value. \( \square \)

We remark that if there are no restrictions on the resulting degree sequence, then we can replace the \( 2s(\Delta_D + 1) \)-block-type set in Lemma 3 by a \( 2s(\Delta_D + 1) \)-type set.

Lemma 4. Let \( (D, s, \tau) \) be a DDConC instance and let \( C \) be a \( 2s(\Delta_D + 1) \)-type set. If \( (D, s, \tau) \) is a yes-instance, then there exists a solution \( A^* \subseteq C^2 \) for \( (D, s, \tau) \), that is, \( \{A^*\} \subseteq s \) and \( \deg_{D + A^*}(v) \in \tau(v) \) for all \( v \in V \).
Similarly, if there are no restrictions on the possible vertex degrees, that is, \( \tau \) is the degree list function \( \tau(v) = \{0, \ldots, n-1\}^2 \) for all \( v \in V(D) \), then we can replace the \( 2s(\Delta_D + 1) \)-block-type set by a \( 2s(\Delta_D + 1) \)-block set.

**Lemma 5.** Let \((D, s, \tau, \Pi)\) be a DDConSeqC instance where \( \tau(v) = \{0, \ldots, n-1\}^2 \) for all \( v \in V(D) \) and let \( C \) be a \( 2s(\Delta_D + 1) \)-block set. If \((D, s, \tau, \Pi)\) is a yes-instance, then there exists a solution \( A^* \subseteq C^2 \) for \((D, s, \tau, \Pi)\), that is, \(|A^*| \leq s \) and \( \sigma(D + A^*) \) fulfills \( \Pi \).

Lemma 3 implies a fixed-parameter algorithm by providing a reduced search space for possible solutions, namely any \( 2s(\Delta_D + 1) \)-block-type set \( C \): Simply try out all possibilities to insert at most \( s \) arcs with endpoints in \( C \) and check whether in one of the cases the degrees and the degree sequence of the resulting graph satisfy the requirements \( \tau \) and \( \Pi \). As \(|C| \leq 2s(\Delta_D + 1) \cdot (\Delta_D + 1)^2(\Delta^*)^2 \) and \( \Delta^* \leq \Delta_D + s \), there are at most \( O(2^{s(\Delta_D+1)}(\Delta_D+s)^2) \) possible subsets of arcs to insert. Altogether, this leads to the following theorem.

**Theorem 6.** If deciding \( \Pi \) is fixed-parameter tractable with respect to the maximum integer in the sequence, then DDConSeqC is fixed-parameter tractable with respect to \((s, \Delta_D)\).

### 3.2 Bounding the solution size \( s \) in \( \Delta^* \)

This subsection constitutes the major part of our framework. The rough overall scheme is analogous to the undirected case as described in Froese et al. [10]. By dropping the graph structure and solving a simpler problem specific number problem on the degree sequence of the input graph, we show how to solve DDConSeqC instances with “large” solutions provided that we can solve the associated number problem efficiently. The number problem is defined so as to simulate the insertion of arcs to a digraph on an integer tuple sequence. Note that adding an arc increases the indegree of a vertex by one and increases the outdegree of another vertex by one. Adding \( s \) arcs can thus be represented by increasing the tuple entries in the degree sequence by an overall value of \( s \) in each component. Formally, the corresponding number problem (abbreviated as \#DDConSeqC) is defined as follows.

**Numbers Only Digraph Degree Constraint Sequence Completion**

**Input:** A sequence \((c_1, d_1), \ldots, (c_n, d_n)\) of \( n \) nonnegative integer tuples, two positive integers \( s \) and \( \xi \), a “tuple list function” \( \tau : \{1, \ldots, n\} \rightarrow 2^{\{0, \ldots, r\} \times \{0, \ldots, r\}} \), and a sequence property \( \Pi \).

**Question:** Is there a sequence \( \sigma' = (c_1', d_1'), \ldots, (c_n', d_n') \) such that \( \sum_{i=1}^n c_i' - c_i = \sum_{i=1}^n d_i' - d_i = s \), \( c_i \leq c_i' \leq \xi \), \( d_i \leq d_i' \leq \xi \), and \((c_i', d_i') \in \tau(i)\) for all \( 1 \leq i \leq n \), and \( \sigma' \) fulfills \( \Pi \)?

Note that if we plug in the degree sequence of a digraph into \#DDConSeqC, then an integer tuple \((x_i, y_i)\) with \( x_i := c_i' - c_i \) and \( y_i := d_i' - d_i \) of a solution tells us to add \( x_i \) ingoing arcs and \( y_i \) outgoing arcs to a vertex \( v_i \). Hence, we call these tuples demands. Having computed the demands, we can then try to solve our original DDConSeqC instance by searching for a set of arcs to add that exactly fulfills the demands. Such a set, however, might not always exist. Hence, the remaining problem is to decide whether it is possible to realize the demands inside the given digraph. The following lemma shows (using flow computations) that this is in fact always possible if the number \( s \) of arcs to add is large compared to \( \Delta^* \).

**Lemma 7.** Let \( D = (V = \{v_1, \ldots, v_n\}, A) \) be a digraph and let \( x_1, \ldots, x_n, y_1, \ldots, y_n \) and \( \Delta^* \) be nonnegative integers such that
We build a flow network $N = (V_N, A_N)$ according to the following steps:

- Add a source vertex $v_s$ and a sink vertex $v_t$ to $N$;
- for each vertex $v_i \in V$, add two vertices $v_i^+$, $v_i^-$ to $N$;
- for each $i \in \{1, \ldots, n\}$, insert the arc $(v_s, v_i^+)$ with capacity $y_i$;
- for each $i \in \{1, \ldots, n\}$, insert the arc $(v_i^-, v_t)$ with capacity $x_i$;
- for each $(v_i, v_j) \in (V \times V) \setminus A$ with $i \neq j$, insert the arc $(v_i^+, v_j^-)$ with capacity one.

Then, there exists an arc set $A' \subseteq (V \times V) \setminus A$ of size $s$ such that for the digraph $D' := (V, A \cup A')$ it holds $\deg_D^+(v_i) = (\deg_D(v_i) + x_i, \deg_D(v_i) + y_i)$ for all $v_i \in V$. Moreover, the set $A'$ can be computed in $O(n(n^2 - m))$ time.

Proof. The proof is based on a flow network which we construct such that the corresponding maximum flow yields the set of arcs to be added to $D$ in order to obtain our target digraph $D'$.

Construction 8. We build a flow network $N = (V_N, A_N)$ according to the following steps.

- Add a source vertex $v_s$ and a sink vertex $v_t$ to $N$;
- for each vertex $v_i \in V$, add two vertices $v_i^+$, $v_i^-$ to $N$;
- for each $i \in \{1, \ldots, n\}$, insert the arc $(v_s, v_i^+)$ with capacity $y_i$;
- for each $i \in \{1, \ldots, n\}$, insert the arc $(v_i^-, v_t)$ with capacity $x_i$;
- for each $(v_i, v_j) \in (V \times V) \setminus A$ with $i \neq j$, insert the arc $(v_i^+, v_j^-)$ with capacity one.

Note that the network $N$ contains $|V_N| \in O(n)$ vertices and $|A_N| \in O(n^2 - m)$ arcs (since $m \leq n^2 - n$, we also have $|A_N| \in \Omega(n)$) and can be constructed in $O(n^2)$ time. See Figure 1 for an illustration of Construction 8. The idea is that adding an arc $(v_i, v_j)$ to $D$ corresponds to sending flow from $v_i^+$ to $v_j^-$. Since, by definition, each vertex $v_i^+$ will only receive at most $y_i$ flow from $v_s$ and each vertex $v_j^-$ will send at most $x_j$ flow to $v_t$, we cannot add more than $s$ arcs (due to Condition (IV)). Moreover, we claim that for $s > 2(\Delta^*)^2 + \Delta^*$ (Condition (V)), the maximum flow in the network is indeed $s$.

To see this, let $V_N^+ := \{v_i^+ \in V_N \mid i \in \{1, \ldots, n\}\}$ and let $V_N^- := \{v_i^- \in V_N \mid i \in \{1, \ldots, n\}\}$. In the following, a vertex $v_i^+ \in V_N^+$ ($v_j^- \in V_N^-$) is called saturated with respect to a flow $f : A_N \to \mathbb{R}^+$, if $f(v_s, v_i^+) = y_i$ ($f(v_j^-, v_t) = x_j$). Suppose that the maximum flow $f$ has value less than $s$. Then, there exist non-saturated vertices $v_i^+ \in V_N^+$ and $v_j^- \in V_N^-$. Let $X \subseteq V_N^+$ be the vertices to which $v_i^+$ has an outgoing arc and let $Y \subseteq V_N^-$ be the vertices which have an outgoing arc to $v_j^-$ in the residual graph. Observe that $\deg_D^+(v_i^+) = n - 1 - \deg_D(v_i)$ and $\deg_D^-(v_j^-) = n - 1 - \deg_D(v_j^-)$. Consequently, due to Condition (III).
|X| ≥ n − 1 − \deg_D^+(v_i) − y_i ≥ n − 1 − \Delta^*. Since \(v_i^+\) is not saturated, we know that |X| > n − 1 − \Delta^* ≥ 0 (due to Condition [II]). By the same reasoning (using Conditions [I] and [II]) it follows that |V| > n − 1 − \Delta^* ≥ 0.

Remember that \(f\) is a flow of maximum value. Hence, we know that each vertex in \(X\) and each vertex in \(Y\) is saturated. Otherwise, there would be an augmenting path in the residual graph, contradicting our assumption of \(f\) being maximal. Now, if a vertex \(x \in X\) would receive flow from a vertex \(y \in Y\), then this implies a backward arc in the residual graph resulting in an augmenting path \(v_s \rightarrow v_i^+ \rightarrow x \rightarrow y \rightarrow v_j^− \rightarrow v_t\), again contradicting our maximality assumption for \(f\). As a consequence, we can conclude that all the flow that goes into \(X\) has to come from the at most \(\Delta^*\) remaining vertices in \(V_N^+ \setminus (Y \cup \{v_i^+\})\). But since \(y_\ell \leq \Delta^*\) for all \(\ell \in \{1, \ldots, n\}\) (by Condition [III]), those \(\Delta^*\) vertices can cover at most a flow of value \((\Delta^*)^2\) and, hence,

\[\sum_{v_i^+ \in X} x_i \leq \sum_{v_i^+ \in V_N^+ \setminus (Y \cup \{v_i^+\})} y_i \leq (\Delta^*)^2.\]  

Since \(X\) is saturated, and since also \(x_\ell \leq \Delta^*\) for all \(\ell \in \{1, \ldots, n\}\) (Condition [II]), we obtain from Condition [IV]

\[s = \sum_{i=1}^n x_i = \sum_{v_i^+ \in X} x_i + \sum_{v_i^− \in V_N^− \setminus X} x_i \leq (\Delta^*)^2 + \sum_{v_i^− \in V_N^− \setminus X} \Delta^* = (\Delta^*)^2 + (\Delta^* + 1) \cdot \Delta^*.\]

This contradicts \(s > 2(\Delta^*)^2 + \Delta^*\) and hence proves the claim.

Now, let \(f\) be a maximum flow in \(N\) (computable in \(O(|V_N||E_N|) = O(n(n^2 – m))\) time [31]) and let \(A' := \{(v_i, v_j) \in V \times V \mid f((v_i^+, v_j^-)) = 1\}\) and note that |\(A'| = s\).

Clearly, adding all arcs in \(A'\) to \(D\) results in a digraph \(D' = (V, A \cup A')\) such that \(\deg_D(v_i) = (\deg_D^+ + x_i, \deg_D^+ + y_i)\) for all \(v_i \in V\).

We remark that a similar flow-construction as given in the above proof was already used by Gale [12] to prove the Gale-Ryser Theorem which characterizes the pairs of integer sequences that can be realized as a bipartite graph. However, since we consider another problem and the two results are incomparable, our proof requires different arguments.

With Lemma 7 we have the key which allows us to transfer solutions of #DDConSeqC to solutions of DDConSeqC. The following lemma is immediate.

**Lemma 9.** Let \(I := (D = (V,A), s, \tau, \Pi)\) with \(V = \{v_1, \ldots, v_n\}\) be an instance of DDConSeqC with \(s > 2(\Delta^*)^2 + \Delta^*\). If there exists an \(s' \in \{2(\Delta^*)^2 + \Delta^* + 1, \ldots, s\}\) such that \(I' := (\deg_D(v_1), \ldots, \deg_D(v_n), s', \Delta^*, \tau', \Pi)\) with \(\tau'(i) := \tau(v_i)\) for all \(v_i \in V\) is a yes-instance of #DDConSeqC, then also \(I\) is a yes-instance of DDConSeqC.

We now have all ingredients for our first main result, namely transferring fixed-parameter tractability with respect to the combined parameter \((s, \Delta^*)\) to fixed-parameter tractability with respect to the single parameter \(\Delta^*\), provided that #DDConSeqC is fixed-parameter tractable with respect to the largest possible integer \(\xi\) in the output sequence. The idea is to search for large solutions based on Lemma 9 using #DDConSeqC. If there are no large solutions (that is, \(s \in O((\Delta^*)^2))\), then we run an FPT-algorithm with respect to \((s, \Delta^*)\).

**Theorem 10.** If DDConSeqC is fixed-parameter tractable for \((s, \Delta^*)\) and #DDConSeqC is fixed-parameter tractable for \(\xi\), then DDConSeqC is fixed-parameter tractable for \(\Delta^*\).
Proof. In the following, let A be the fixed-parameter algorithm solving DDConSeqC in \(h(s, \Delta^*) \cdot n^{O(1)}\) time and let \(A'\) be the fixed-parameter algorithm solving \#DDConSeqC in \(h' \xi \cdot n^{O(1)}\) time. Let \(I := (D = (V, A), s, \tau, \Pi)\) be a DDConSeqC instance.

If \(s \leq 2(\Delta^*)^2 + \Delta^*\), then we can run algorithm A on I in time \(h(s, \Delta^*) \cdot n^{O(1)} \leq g(\Delta^*) \cdot n^{O(1)}\) for some function g.

Otherwise, we check for each \(s' \in \{2(\Delta^*)^2 + \Delta^* + 1, \ldots, s\}\), whether the instance \(I_{s'} := (\deg_D(v_1), \ldots, \deg_D(v_n), s', \tau', \Pi)\) with \(\tau'(i) := \tau(v_i)\) for all \(v_i \in V\) is a yes-instance of \#DDConSeqC using algorithm \(A'\). Note that the running time is at most \(s' \cdot h'(\Delta^*) \cdot n^{O(1)}\). If we find a yes-instance \(I_{s'}\) for some \(s'\), then we know by Lemma 9 that I is also a yes-instance.

If \(I_{s'}\) is a no-instance for all \(s' \in \{2(\Delta^*)^2 + \Delta^* + 1, \ldots, s\}\), then we also know that there cannot exist a solution for I of size larger than \(2(\Delta^*)^2 + \Delta^*\) since the existence of a solution for a DDConSeqC instance clearly implies a solution for the corresponding \#DDConSeqC instance. Therefore, I is a yes-instance if and only if \(I' := (D, 2(\Delta^*)^2 + \Delta^*, \tau, \Pi)\) is a yes-instance. We can thus run algorithm A on I' in \(h(2(\Delta^*)^2 + \Delta^* \cdot n^{O(1)}\) time.

Our second main result allows to transfer a polynomial-size problem kernel with respect to \((s, \Delta^*)\) to a polynomial-size problem with respect to \(\Delta^*\) if \#DDConSeqC is polynomial-time solvable. The proof is analogous to the proof of Theorem 10.

Theorem 11. If DDConSeqC admits a problem kernel with g(s, Δ*) vertices computable in p(n) time and \#DDConSeqC is solvable in q(n) time for polynomials p and q, then DDConSeqC admits a problem kernel with g(2(Δ*)^2 + Δ*, Δ*) vertices computable in O(sq(n) + p(n)) time.

Proof. Let \(I := (D = (V, A), s, \tau, \Pi)\) be a DDConSeqC instance. If \(s \leq 2(\Delta^*)^2 + \Delta^*\), then we simply run the kernelization algorithm on I obtaining an equivalent instance of size at most \(g(2(\Delta^*)^2 + \Delta^*, \Delta^*)\) in \(O(p(n))\) time. Otherwise, we check in \(O(q(n))\) time for each \(s' \in \{2(\Delta^*)^2 + \Delta^* + 1, \ldots, s\}\), whether the instance

\[I_{s'} := (\deg_D(v_1), \ldots, \deg_D(v_n), s', \tau', \Pi)\]

with \(\tau'(i) := \tau(v_i)\) for all \(v_i \in V\) is a yes-instance of \#DDConSeqC. Note that the running time is thus \(s' \cdot q(n)\). If we find a yes-instance \(I_{s'}\) for some \(s'\), then we know by Lemma 9 that also I is a yes-instance, and thus we return a trivial DDConSeqC yes-instance. If \(I_{s'}\) is a no-instance for all \(s' \in \{2(\Delta^*)^2 + \Delta^* + 1, \ldots, s\}\), then we also know that there cannot exist a solution for I of size larger than \(2(\Delta^*)^2 + \Delta^*\) since the existence of a solution for a DDConSeqC instance clearly implies a solution for the corresponding \#DDConSeqC instance. Therefore, I is a yes-instance if and only if \(I' := (D, 2(\Delta^*)^2 + \Delta^*, \tau, \Pi)\) is a yes-instance. Again, we run the kernelization algorithm on I' and return an equivalent instance with at most \(g(2(\Delta^*)^2 + \Delta^*, \Delta^*)\) vertices in \(O(p(n))\) time. The overall running time is thus in \(O(sq(n) + p(n))\) and we obtain a problem kernel with respect to \(\Delta^*\).

4 Applications

In the following, we show how the framework from Section 3 can be applied to three special cases of DDConSeqC. These special cases naturally extend known problems on undirected graphs to the digraph setting.
4.1 Digraph Degree Constraint Completion

In this section, we investigate the NP-hard special case of DDConSeqC where the property \( \Pi \) allows all degree sequences [23], see Figure 2 for two illustrating examples.

**Digraph Degree Constraint Completion (DDConC)**

**Input:** A digraph \( D = (V, A) \), a positive integer \( s \), and a “degree list function” \( \tau: V \to 2^{{\{0,\ldots,r\}}} \times 2^{{\{0,\ldots,r\}}} \).

**Question:** Is it possible to obtain a digraph \( D' \) by adding at most \( s \) arcs to \( D \) such that \( \deg_{D'}(v) \in \tau(v) \) for all \( v \in V \)?

DDConC is the directed (completion) version of the well-studied undirected Degree Constraint Editing problem [13, 28] for which an \( O(r^5) \) problem kernel is known [10]. We subsequently transfer the polynomial-size kernel for the undirected case to a polynomial-size kernel for DDConC with respect to \( \Delta^* \). Note that the parameter \( \Delta^* \) is clearly at most \( r \). Since it is trivial to decide \( \Pi \), we obtain fixed-parameter tractability of DDConC with respect to \( (s, \Delta_D) \) due to Theorem 6. Note that Theorem 6 is based on a bounded search space, namely a \( 2s(\Delta_D + 1) \)-type set (see Definition 1). For DDConC, we further strengthen this result by removing all vertices not in the \( 2s(\Delta_D + 1) \)-type set and adjusting the degree list function \( \tau \) properly. Lemma 4 then yields the correctness of this approach resulting in a polynomial-size problem kernel.

We start with the following simple reduction rule.

**Reduction Rule 4.1.** Let \( (D = (V, A), s, \tau) \) be a DDConC instance. If there are more than \( 2s \) unsatisfied vertices, then return a trivial no-instance. Moreover, if there exists a vertex \( v \in V \) with \( \deg_{D}^-(v) > \Delta^* \) or \( \deg_{D}^+(v) > \Delta^* \), then also return a trivial no-instance.

**Lemma 12.** Reduction Rule 4.1 is correct and can be computed in \( O(m + |\tau|) \) time.

**Proof.** If there are more than \( 2s \) unsatisfied vertices, then we can return a trivial no-instance since adding an arc can satisfy at most two vertices. Also, by adding arcs we can only increase in- and outdegrees of vertices. Hence, we can return a no-instance if the in- or outdegree of a vertex is larger than \( \Delta^* \). This proves the correctness.

The reduction rule is applicable in \( O(m + |\tau|) \) time by computing the degree \( \deg_{D}(v) \) of each vertex \( v \in V \) in \( O(n + m) \) time and subsequently iterating through the list \( \tau(v) \). □

Based on Reduction Rule 4.1 we obtain a polynomial-size problem kernel with respect to the combined parameter \( (s, \Delta^*) \) as follows.

**Theorem 13.** DDConC admits a problem kernel containing \( O(s(\Delta^*)^3) \) vertices. It is computable in \( O(m + m^2) \) time.
Proof. Let $I = (D = (V,A),s,\tau)$ be an instance of DDConC. First, we apply Reduction Rule 1 in $O(m + |\tau|)$ time. If a no-instance is returned, then we are done. Otherwise, we know that there are at most $2s$ unsatisfied vertices. Also, we know that $\deg_D^-(v) \leq \Delta^*$ and $\deg_D^+(v) \leq \Delta^*$ for all $v \in V$. We compute an $\alpha$-type set $C$ (see Definition 11) for $\alpha := 2s(\Delta_D + 1)$ in $O(m + |\tau| + r^2)$ time (Lemma 3) and return the instance $I' = (D' := D[C], s, \tau_C)$, where the adjusted degree list $\tau_C(v)$, for each $v \in V$, is defined as follows:

$$\tau_C(v) := \{(i,j) \in \{0, \ldots, \Delta^*\} \times \{0, \ldots, \Delta^*\} \mid (i,j) \in C, |N_D^-(v) \setminus C|, |N_D^+(v) \setminus C| \in \tau(v)\}.$$  

The instance $I'$ can be computed in $O(m + |\tau| + r^2)$ time. We now show that $I'$ is an equivalent instance of DDConC.

Assume that $I'$ is yes-instance, that is, there exists a set $A \subseteq C^2$ of at most $s$ such that $\deg_{D' + A}(v) \in \tau_C(v)$ for each $v \in C$. Then, the set $A$ is also a solution for $I$ since for each vertex $v \in C$, it holds

$$\deg_{D' + A}(v) = \deg_{D' + A}(v) + (|N_D^-(v) \setminus C|, |N_D^+(v) \setminus C|) \in \tau(v),$$

by definition of $\tau_C(v)$. Moreover, for each vertex $v \in V \setminus C$, we have $\deg_{D' + A}(v) = \deg_D(v) \in \tau(v)$ since $V \setminus C$ contains only satisfied vertices. Hence, $I$ is a yes-instance.

Conversely, let $I$ be a yes-instance. Then, by Lemma 4, we know that there exists an arc set $A^* \subseteq C^2$ of size at most $s$ such that $\deg_{D' + A^*}(v) \in \tau(v)$ for all $v \in V$. Then, for each vertex $v \in C$, it holds

$$\deg_{D' + A^*}(v) = \deg_{D' + A^*}(v) - (|N_D^-(v) \setminus C|, |N_D^+(v) \setminus C|) \in \tau_C(v),$$

by definition of $\tau_C$. Hence, also $I'$ is a yes-instance.

Concerning the size of $D'$, observe that $C$ contains at most $2s$ unsatisfied vertices and at most $\alpha$ vertices for each of the $(\Delta^* + 1)^2$ possible types. Therefore,

$$|C| \leq 2s + (\Delta^* + 1)^2 \cdot \alpha \leq 2s + (\Delta^* + 1)^2 \cdot 2s(\Delta_D + 1).$$

Since $\Delta_D \leq \Delta^*$, we obtain a problem kernel with $O(s(\Delta^*)^3)$ vertices. The overall running time is in $O(m + |\tau| + r^2)$. \qed

Theorem 14. DDConC admits a problem kernel containing $O(s(\Delta^*)^3) \subseteq O(sr^3)$ vertices. It is computable in $O(m + nr^2)$ time.

The goal now is to use our framework (Theorem 11) to transfer the polynomial-size kernel with respect to $(s, \Delta^*)$ to a polynomial-size kernel with respect to $\Delta^*$. To this end, we show that the corresponding number problem is polynomial-time solvable. Let us start by defining #DDConC:

**Numbers Only Digraph Degree Constraint Completion (#DDConC)**

**Input:** A sequence $(c_1,d_1), \ldots, (c_n,d_n)$ of $n$ nonnegative integer tuples, two positive integers $s$ and $\xi$, and a “tuple list function” $\tau: \{1, \ldots, n\} \to 2^{0^{(\xi - r)} \times 0^{(\xi - r)}}$.

**Question:** Is there a sequence $(c'_1,d'_1), \ldots, (c'_n,d'_n)$ such that $\sum_{i=1}^n c'_i - c_i = \sum_{i=1}^n d'_i - d_i = s$, and $c_i \leq c'_i \leq \xi$, $d_i \leq d'_i \leq \xi$, and $(c'_i, d'_i) \in \tau(i)$ for all $1 \leq i \leq n$?

#DDConC can be solved in polynomial time by a dynamic programming algorithm.

**Lemma 15.** #DDConC is solvable in $O(n(sr)^2)$ time.
Figure 3: Example instance of DDSeqC. The input digraph (solid arcs) has the degree sequence \{((0, 1), (0, 2), (2, 0), (2, 1))\}. Adding the dashed arc yields a digraph with the given target sequence \(\sigma\).

Proof. Let \(I := (\tau, (c_1, d_1), \ldots, (c_n, d_n), s)\) be an instance of \#DDCONC. We solve \(I\) using a modified version of the dynamic programming algorithm for NCE due to Froese et al. \[10\]. To this end, we define the Boolean table \(M[i, j, l]\) for \(i \in \{1, \ldots, n\}, j, l \in \{0, \ldots, s\}\), where \(M[i, j, l] = \text{true}\) if and only if there exist tuples \((c'_{i_1}, d'_{i_1}), \ldots, (c'_{i_j}, d'_{i_j})\) with \(c'_p \geq c_p, d'_p \geq d_p\) and \((c'_p, d'_p) \in \tau(p)\) for all \(p \in \{1, \ldots, i\}\) such that \(\sum_{p=1}^{i} (c'_p - c_p) = j\) and \(\sum_{p=1}^{i} (d'_p - d_p) = l\). Thus, \(I\) is a yes-instance if \(M[n, s, s] = \text{true}\). We compute \(M\) based on the recurrence

\[
M[i, j, l] = \begin{cases} 
\text{true,} & \text{if } (c_1 + j, d_1 + l) \in \tau(1), \\
\text{false,} & \text{otherwise.}
\end{cases}
\]

The size of \(M\) is in \(O(ns^2)\) and a single entry can be computed in \(O(r^2)\).

Combining Theorem \[14\] and Lemma \[15\] yields the following corollary of Theorem \[11\].

**Corollary 16.** DDConC admits a problem kernel containing \(O((\Delta^*)^5) \subseteq O(r^5)\) vertices. It is computable in \(O(m + ns^3r^2)\) time.

### 4.2 Digraph Degree Sequence Completion

In this section, we investigate the NP-hard special case of DDConSeqC where \(\tau\) does not restrict the allowed degree of any vertex and \(\Pi\) is fulfilled by exactly one specific degree sequence \(\sigma\) \[29\]. The undirected problem variant is due to Golovach and Mertzios \[14\].

**Digraph Degree Sequence Completion (DDSeqC)**

**Input:** A digraph \(D = (V, A)\), a digraph degree sequence \(\sigma\) with \(|V|\) elements and an integer \(s\).

**Question:** Is it possible to obtain a digraph \(D'\) by adding at most \(s\) arcs to \(D\) such that \(\sigma(D') = \sigma\)?

See Figure 3 for an example. For DDSeqC, the parameter \(\Delta^*\) is by definition equal to \(\Delta_\sigma\). Since deciding whether \(\Pi\) holds can again be done efficiently, we immediately obtain fixed-parameter tractability of DDConC with respect to \((s, \Delta_\sigma)\) due to Theorem \[6\]. We further strengthen this result by developing a polynomial-size problem kernel for DDConC with respect to \((s, \Delta_\sigma)\). The idea for the kernelization is inspired by the problem kernel for the undirected problem \[14\]. The main idea is to only keep the vertices of a \(2s(\Delta_D + 1)\)-block set (see Definition \[1\]) together with some additional “dummy” vertices and to adjust the degree sequence \(\sigma\) properly. The following problem kernel is based on ideas used for the kernel results for undirected graphs by Golovach and Mertzios \[14\].

\[2\] Although not stated explicitly, the NP-hardness follows from the prove of Theorem 3.2 of Millani \[29\] as the construction allows for only one feasible target degree sequence.
Reduction Rule 4.2. Let \((D, \sigma, s)\) be a DDSEQC instance. If \(\Delta^-_D > \Delta^-_\sigma\) or \(\Delta^+_D > \Delta^+_\sigma\), then return a trivial no-instance. Further, if there exists a vertex \(v\) in \(D\) such that \(\lambda_{\sigma(D)}(\deg_D(v)) > \lambda_{\sigma}(\deg_D(v)) + 2s\), then also return a trivial no-instance.

Lemma 17. Reduction Rule 4.2 is correct and can be applied in \(O(n+m)\) time.

Proof. Clearly, since we are only allowed to add arcs to the digraph \(D\), we can never decrease the in- or outdegree of any vertex. Hence, if \(\Delta^-_D > \Delta^-_\sigma\) or \(\Delta^+_D > \Delta^+_\sigma\), then we have a no-instance. Furthermore, by adding an arc, we can change the degrees of at most two vertices. Hence, if a vertex degree \(\deg_D(v)\) appears more than \(\lambda_{\sigma}(\deg_D(v)) + 2s\) times in \(\sigma(D)\), then we need to change the degrees of more than \(2s\) vertices, which is not possible.

To apply the rule, we first compute \(\Delta^-_D\) and \(\Delta^+_D\) in \(O(n)\) time. Then, in \(O(n+m)\) time, we compute the digraph degree sequence \(\sigma(D)\) checking for each vertex \(v\) whether \(\deg^-_D(v) > \Delta^-_\sigma\) or \(\deg^+_D(v) > \Delta^+_\sigma\). We then sort the sequences \(\sigma(D)\) and \(\sigma\) lexicographically in \(O(n)\) time (using radix sort). We can then check in \(O(n)\) time whether \(\lambda_{\sigma(D)}(t) > \lambda_{\sigma}(t) + 2s\) for some tuple \(t \in \sigma(D)\) by iterating once over both sorted sequences \(\sigma(D)\) and \(\sigma\). The overall running time is in \(O(n+m)\).

Theorem 18. DDSEQC admits a problem kernel containing \(O(s\Delta^3_\sigma)\) vertices computable in \(O(m+n\Delta_\sigma)\) time.

Proof. Let \((D, \sigma, s)\) be a DDSEQC instance. First, we apply Reduction Rule 4.2 in \(O(n+m)\) time. If a no-instance is returned, then we are done. Otherwise, we know that \(\Delta_D \leq \Delta_\sigma\) and that \(\lambda_{\sigma(D)}(\deg_D(v)) \leq \lambda_{\sigma}(\deg_D(v)) + 2s\) for each \(v \in V(D)\). We now compute a \((\Delta_D + 1)\)-block set \(C\) (see Definition 1) in \(O(n+m)\) time (Lemma 2).

We return the instance \((D', \sigma', s)\) which is defined as follows. The digraph \(D'\) is constructed from \(D\) by the following steps:

- Delete all vertices of \(V(D) \setminus C\).
- Add \(h := \Delta_\sigma + 2\) new vertices \(W := \{w_1, \ldots, w_h\}\) and insert all arcs \(W \times W\).
- For each \(v \in C\) such that the number \(r^-_v := |\{(u \in V(D) \setminus C) | (u, v) \in A(D)\}|\) of in-neighbors in \(V(D) \setminus C\) is at least one, insert the arcs \(\{(w_i, v) | 1 \leq i \leq r^-_v\}\).
- For each \(v \in C\) such that the number \(r^+_v := |\{(u \in V(D) \setminus C) | (v, u) \in A(D)\}|\) of out-neighbors in \(V(D) \setminus C\) is at least one, insert the arcs \(\{(v, w_i) | 1 \leq i \leq r^+_v\}\).

The digraph \(D'\) can be constructed in \(O(n\Delta_\sigma)\) time. Observe that \(\deg^-_{D'}(w_i) \geq \Delta_\sigma + 1\) and \(\deg^+_{D'}(w_i) \geq \Delta_\sigma + 1\) holds for all \(i \in \{1, \ldots, h\}\), and that \(\deg_{D'}(v) = \deg_{D}(v) \leq \Delta_\sigma\) holds for all \(v \in C\). The number of vertices in \(D'\) equals \(|C| + h\). Note that \(C\) contains at most \(2s(\Delta_D + 1)\) vertices of each of the \((\Delta_\sigma + 1)^2\) possible vertex degrees in \(D\). Thus, \(D'\) contains \(O(s\Delta^3_\sigma)\) vertices.

The digraph degree sequence \(\sigma'\) is constructed from \(\sigma\) as follows:

- For each vertex \(v \in V(D) \setminus C\) that was removed from \(D\), remove a copy of the tuple \(\deg_D(v)\) from \(\sigma\).
- For each \(i \in \{1, \ldots, h\}\), add the tuple \(\deg_{D'}(w_i)\).
Note that this construction is well-defined, that is, we can always apply the first step and remove a copy of $\deg_D(v)$ from $\sigma$ since we remove at most
\[
|T_{\deg_D(v)}| - 2s(\Delta_D + 1) < \lambda_{\sigma}(\deg_D(v)) - 2s \leq \lambda_{\sigma}(\deg_D(v))
\]
copies. The construction of $\sigma'$ can be done in $O(n)$ time. Hence, the overall running time of computing the problem kernel is in $O(m + n\Delta_{\sigma})$.

It remains to show that $(D', \sigma', s)$ is a yes-instance if and only if $(D, \sigma, s)$ is a yes-instance. Assume first that $(D, \sigma, s)$ is a yes-instance. We know from Lemma 8 that there exists a solution $A^* \subseteq \binom{\mathcal{C}}{2}$ with $\sigma(D + A^*) = \sigma$. Using
\[
\forall v \in V(D) \setminus C : \deg_{D+A^*}(v) = \deg_D(v),
\forall v \in C : \deg_{D'+A'}(v) = \deg_{D+A^*}(v), \text{ and}
\forall w_i \in W : \deg_{D'+A'}(w_i) = \deg_D(w_i),
\]
it is then easy to verify that $\sigma(D' + A^*) = \sigma'$, and thus, $(D', \sigma', s)$ is a yes-instance.

Conversely, let $A' \subseteq \binom{\mathcal{V}(D')}{2}$ be a solution for $(D', \sigma', s)$ with $\sigma(D' + A') = \sigma'$. We claim that $A' \subseteq \binom{\mathcal{C}}{2}$, that is, $A'$ does not contain an arc incident to a vertex in $W$. To see this, recall that by construction
\[
\deg_D'(w_1) = \Delta_{\sigma}^\downarrow \geq \ldots \geq \deg_D'(w_h) \geq \Delta_{\sigma} + 1 > \deg_D'(v) \text{ and}
\deg_D'(w_1) = \Delta_{\sigma}^\uparrow \geq \ldots \geq \deg_D'(w_h) \geq \Delta_{\sigma} + 1 > \deg_D'(v)
\]
hold for all $v \in C$. That is, $\deg_D'(w_1) = (\Delta_{\sigma}^\downarrow, \Delta_{\sigma}^\uparrow)$, and thus a solution must not add arcs incident to $w_1$. It follows that $\deg_{D'+A'}(w_1) = \deg_D'(w_1)$. This recursively also holds for $w_2, \ldots, w_h$. Hence, $A'$ does not contain any arcs incident to vertices in $W$, that is, $A' \subseteq \binom{\mathcal{C}}{2}$. Thus, we can derive
\[
\forall w_i \in W : \deg_{D'+A'}(w_i) = \deg_D(w_i),
\forall v \in C : \deg_{D'+A'}(v) = \deg_{D+A^*}(v), \text{ and}
\forall v \in V(D) \setminus C : \deg_{D'+A'}(v) = \deg_D(v).
\]
It is now straightforward to check that $\sigma(D + A') = \sigma$. \hfill $\square$

**Theorem 19.** DDSEQC admits a problem kernel containing $O(s\Delta_{\sigma}^3)$ vertices. It is computable in $O(m + n\Delta_{\sigma})$ time.

The corresponding number problem is defined as follows.

**Numbers Only Digraph Degree Sequence Completion (#DDSEQC)**

**Input:** A sequence $(c_1, d_1), \ldots, (c_n, d_n)$ of $n$ nonnegative integer tuples, a multiset $\phi$ containing $n$ nonnegative integer tuples, and two positive integers $s$ and $\xi$.

**Question:** Is there an ordering $(c'_1, d'_1), \ldots, (c'_n, d'_n)$ of the tuples in $\phi$ such that $\sum_{i=1}^n c'_i - c_i = \sum_{i=1}^n d'_i - d_i = s$, and $c_i \leq c'_i \leq \xi$, and $d_i \leq d'_i \leq \xi$, for all $1 \leq i \leq n$?

#DDSEQC can be solved in polynomial time by finding perfect matchings in an auxiliary graph.

**Lemma 20.** #DDSEQC is solvable in $O(n^{2.5})$ time.
Figure 4: Example instance of DDA. The input digraph with three components (solid arcs) is 1-anonymous since there is only one vertex with degree (0, 1). By adding the dashed arc, the digraph becomes 7-anonymous since all vertices have degree (1, 1).

Proof. We show how to solve the problem by computing a perfect matching in a bipartite graph. Let \((c_1, d_1), \ldots, (c_n, d_n), \{(c'_1, d'_1), \ldots, (c'_n, d'_n)\}, s\) be a TSC instance. To start with, note that the second condition in the problem definition is independent of the respective ordering since, for any ordering \(\pi_1, \ldots, \pi_n\) of \(1, \ldots, n\), it holds

\[
\sum_{i=1}^{n} (c'_{\pi_i} - c_i) = \sum_{i=1}^{n} c'_{\pi_i} - \sum_{i=1}^{n} c_i
\]

The same holds analogously for the sum over the \(d_i\)'s. Thus, we can first check whether

\[
\sum_{i=1}^{n} c'_i - \sum_{i=1}^{n} c_i = \sum_{i=1}^{n} d'_i - \sum_{i=1}^{n} d_i = s
\]

holds and otherwise reject the instance. If the above condition is met, then we try to find an ordering of the tuples in \(\phi\) as follows: We construct an undirected bipartite graph \(G := (V \cup W, E)\). For each \(i \in \{1, \ldots, n\}\), there is a vertex \(v_i \in V\) corresponding to the tuple \((c_i, d_i)\), and a vertex \(w_i \in W\) corresponding to \((c'_i, d'_i)\). For each \(i, j \in \{1, \ldots, n\}\), \(i \neq j\), the edge \(\{v_i, w_j\}\) is in \(E\) if and only if \(\xi \geq c'_j \geq c_i\) and \(\xi \geq d'_j \geq d_i\) hold. The graph \(G\) can be computed in \(O(n^2)\) time. Note that any perfect matching in \(G\) defines an ordering of the tuples in \(\phi\) that satisfies the first condition in the problem definition. Hence, we can solve a TSC instance by computing a perfect matching in a bipartite graph, which can be done in \(O(|E|\sqrt{|V \cup W|}) = O(n^{2.5})\) time [20].

Combining Theorem 19 and Lemma 20 yields the following corollary of Theorem 11.

Corollary 21. DDSeqC admits a problem kernel containing \(O(\Delta^5\sigma)\) vertices. It is computable in \(O(sn^{2.5})\)-time.

4.3 Degree Anonymity

We extend the definition of Degree Anonymity due to Liu and Terzi [26] to digraphs and obtain the following NP-hard problem [29] (Figure 4 presents an example):

DIGRAPH DEGREE ANONYMITY (DDA)

Input: A digraph \(D = (V, A)\) and two positive integers \(k\) and \(s\).

Question: Is it possible to obtain a digraph \(D'\) by adding at most \(s\) arcs to \(D\) such that \(D'\) is \(k\)-anonymous, that is, for every vertex \(v \in V\) there are at least \(k - 1\) other vertices in \(D'\) with the same in- and outdegree?

The computational and parameterized complexity as well as the (in)approximability of Degree Anonymity are well-studied [3, 6, 19]. There also exist many heuristic approaches to solve the problem [4, 18, 27]. Notably, our generic approach shown in Section 3.2 originates
from a heuristic of Liu and Terzi [26] for Degree Anonymity. Later, Hartung et al. [19] used this heuristic to prove that “large” solutions of Degree Anonymity can be found in polynomial time and Froese et al. [10] extended this approach for a more general class of problems. Surprisingly, while we can apply our generic approach on digraphs for the problems shown in the previous two subsections, DDA seems to be more intricate. As we will subsequently indicate, the source of the difficulties seems to be associated with the number problem.

We first provide a problem kernel based on Lemma 5 in a similar fashion as in the proof of Theorem 19. We keep a $2s(\Delta_D + 1)$-block set $C$ in the kernel and remove all other vertices. In order to not change the degrees of the vertices we kept, we introduce “dummy” vertices that will have a very high degree so that there is no interference with the vertices we kept. The approach is inspired by Hartung et al. [19] with their polynomial kernel for Degree Anonymity with respect to $(s, \Delta_G)$ (\[\Delta_G\] denotes the maximum degree of the given undirected graph $G$). We provide a problem kernel based on Lemma 5 in a similar fashion as in the proof of Theorem 19. More precisely, by Lemma 5 we know that we only need to keep a $2s(\Delta_D + 1)$-block set $C$, that is, $2s(\Delta_D + 1)$ arbitrary vertices of each block. Since deleting all vertices that are not in $C$ changes the degrees of the vertices in $C$, we repair this in a similar way as in the kernel stated in Theorem 19. After deleting the vertices that are not in $C$, we add vertices incident to the vertices in $C$ in such a way that the vertices in $C$ keep their degree. Denoting the set of newly added vertices by $P$, we also need to separate the vertices in $P$ from the vertices in $C$ so that they do not interfere in the target degree sequence. We do this, as in the kernel stated in Theorem 19, by increasing the degrees of all vertices in $P$ to at least $\Delta_D + s + 1$. Furthermore, we need to ensure that a solution in the new instance does not insert arcs between vertices in $C$ and vertices in $P$ since we cannot map such solutions back to solutions for the original instance. Solving this issue is not as simple as in the kernel stated in Theorem 19 and requires some adjustment of the actual number of vertices we keep. As a result, we will prove that if there is a solution inserting arcs between $C$ and $P$, then there is also a solution not inserting such arcs (Lemma 22).

Another difference to the kernel presented in Theorem 19 is the adjustment of the anonymity level $k$: If $k$ is large, then we need to shrink it since otherwise we would always create no-solutions. The general idea is to keep the “distance to size $k$”, meaning that if in the original instance some block contains $k + x$ vertices for some $x \in \{-2s, \ldots, 2s\}$, then in the new instance this block should contain $k' + x$ vertices where $k'$ is the new anonymity level. The reason for the specific range of values for $x$ between $-2s$ and $2s$ is that if some block has size larger than $k + 2s$ for example, then after adding $s$ arcs, this block will still be of size larger than $k$. Similarly, if a block contains less than $k - 2s$ vertices, then after adding $s$ arcs it will contain less than $k$ vertices and it will violate the $k$-anonymity constraint unless it is empty. Hence, the interesting cases for $x$ are between $-2s$ and $2s$. (In order to ensure that there is a solution not inserting arcs between $C$ and $P$, we need to increase this range from $-2s$ to $4s$, see the proof of Lemma 22 for further details.)

In the following, we describe the details of our kernelization algorithm, see Algorithm 1 for the pseudocode. Observe that our general approach is an adaption of the polynomial kernel for the undirected Degree Anonymity problem provided by Hartung et al. [19].

Lemma 22. Let $(D, k, s)$ be an instance of DDA and let $(D', k', s)$ be the instance computed by Algorithm 1, where $P := P_{\text{in}} \cup P_{\text{out}}$ is the set of newly added vertices. If there is a solution $S \subseteq V(D')^2$ with $|S| \leq s$, then there is also a solution $S' \subseteq V(D')^2$ with $|S'| \leq |S|$
Algorithm 1: The pseudocode of the algorithm computing a polynomial-size kernel with respect to \( (\Delta_D, s) \) for DDA.

**Input:** A digraph \( D = (V, A) \) and integers \( k, s \in \mathbb{N} \).

**Output:** A digraph \( D' \) and integers \( k', s \in \mathbb{N} \).

1. \( |V| \leq (\Delta_D + 1)^2(\beta + 2s) \) then // \( \beta \) is defined as \( \beta := (\Delta_D + 2)2s \)
   2. **return** \( (D, k, s) \)
   3. \( k' \leftarrow \min\{k, \beta\} \)
   4. \( C \leftarrow \emptyset \)
   5. **foreach** distinct tuple \( t \) occurring in \( \sigma(D) \) do
      6. **if** \( 2s < |B_D(t)| < k - 2s \) then // insufficient budget for \( B_D(t) \)
         7. **return** trivial no-instance
      8. **if** \( k < \beta \) then // determine number of retained vertices
         9. \( x \leftarrow \min\{|B_D(t)|, \beta + 2s\} // keep at most \( \beta + 2s \) vertices
      10. **else if** \( |B_D(t)| \leq 2s \) then // “small” block
          11. \( x \leftarrow |B_D(t)| // keep all vertices (“distance to size zero”)
      12. **else** // “large” block and \( k' = \beta \)
          13. \( x \leftarrow k' + \min(2s, (|B_D(t)| - k)) // keep “distance to size \( k \)
      14. add \( x \) arbitrary vertices from \( B_D(t) \) to \( C \)
   15. \( D' \leftarrow D[C] \)
   16. **foreach** \( v \in C \) do // insert new vertices to preserve degrees of vertices in \( C \)
      17. add \( \deg_D^+(v) - \deg_D^+(v) \) many vertices with an ingoing arc from \( v \) to \( D' \)
      18. add \( \deg_D^+(v) - \deg_D^+(v) \) many vertices with an outgoing arc to \( v \) to \( D' \)
   19. denote by \( P_{in} \) the set of vertices inserted in Line 17
   20. denote by \( P_{out} \) the set of vertices inserted in Line 18
   21. while \( \min(|P_{in}|, |P_{out}|) < \max(\Delta_D + s + 1, k') \) do
      22. add a new vertex \( v \) to \( D' \) and add \( v \) to \( P_{in} \)
      23. add a new vertex \( u \) to \( D' \) and add \( u \) to \( P_{out} \)
      24. add the arc \((u, v)\) to \( D' \)
      25. add all arcs \( P_{in}^2 \) and \( P_{out}^2 \) to \( D' \)
      26. add all arcs from \( P_{in} \times P_{out} \) to \( D' \)
   27. **return** \( (D', k', s) \) // ensure high degree difference from vertices in \( C \)
   // separate \( P_{in} \) from \( P_{out} \)

such that \( V(S') \cap P = \emptyset \).

**Proof.** Let \( S \subseteq V(D') \) 2 be a solution for \((D', k', s)\) such that \( V(S') \cap P \neq \emptyset \). We construct a new solution \( S' \subseteq V(D') \) 2 such that \( |S'| \leq \min\{|S|, |V(S') \cap P = \emptyset \} \). The idea is to replace the endpoints of arcs that are in \( P \) by new endpoints from one “large” block (of size at least \( \beta + 2s \)) in \( C \). To this end, observe that if \( \beta \) \( V(D) \leq (\Delta_D + 1)^2(\beta + 2s) \), then Algorithm 1 returns the original instance (see Line 1) and we are done. Hence, there is at least one block \( B_D(t) \) for some \( t \in \sigma(D') \) of size at least \( \beta + 2s \) since there are at most \( (\Delta_D + 1)^2 \) blocks. We will use vertices in \( B_D(t) \) as a replacement for the vertices in \( P \) within the arcs of \( S \).

We now construct \( S' \). To this end, initialize \( S' := S \cap C^2 \) and insert further arcs in the following way. First, consider those arcs in \( S \) that have exactly one endpoint in \( P \). For each arc \((u, v) \) in \( S \) with \( u \in C \) and \( v \in P \), add the arc \((u, w)\) to \( S' \) where \( w \in B_D(t) \) such that \( w \) is not incident to any arc in \( S' \) and not an outneighbor of \( u \). Since \( |B_D(t)| \geq \beta + 2s = (\Delta_D + 3)2s \) and \( |S'| \leq s \), it follows that \( B_D(t) \) contains such a vertex \( w \). Similarly, for each arc \((v, u) \) in \( S \) with \( u \in C \) and \( v \in P \), add the arc \((w, u)\) to \( S' \) where \( w \in B_D(t) \) is a vertex not incident
to any arc in \(S'\) and not an inneighbor of \(u\). Again, due to the size of \(B_D(t)\), such a vertex exists.

Second, consider those arcs in \(S\) having both endpoints in \(P\). For each arc \((u, v)\) in \(S\) with \(u, v \in P\), add the arc \((u', v')\) to \(S'\) where \(u', v' \in B_D(t)\) such that neither \(u'\) nor \(v'\) is incident to any arc in \(S'\) and \((u', v') \notin A(D')\). Since \(|B_D(t)| \geq \beta + 2s = (\Delta_D + 3)2s\) and \(|S'| \leq s\), it follows that these vertices \(u'\) and \(v'\) exist. Observe that after all these modifications, there are still at least \(\beta\) vertices left in \(B_D(t)\).

Clearly, we have \(|S'| \leq |S|\). It remains to prove that \(D' + S'\) is \(k'\)-anonymous. To this end, observe that since the outdegree of each vertex in \(P_{in}\) is at least \(|P_{out}| - 1 \geq \Delta_D + s + 1\) (see Line 26) larger than the outdegree of any vertex in \(P_{out}\), it follows that the vertices in \(P\) which are incident to an arc in \(S\) end up in blocks of \(D' + S\) that are empty in \(D'\). Thus, at least \(k'\) vertices in \(P\) are the head of an arc in \(S\) and at least \(k'\) vertices in \(P\) are the tail of an arc in \(S\). Hence, we used at least \(k'\) vertices from \(B_D(t)\) as an replacement in \(S'\) and thus the blocks \(B_{D'+S'}(t + (1, 0))\) and \(B_{D'+S'}(t + (0, 1))\) contain at least \(k'\) vertices. Furthermore, all other vertices in \(C\) have the same degree in \(D' + S\) and in \(D' + S'\) and the vertices in \(P\) are not incident to any arc in \(S'\). Since \(S\) was a solution, it follows that also \(D' + S'\) is \(k'\)-anonymous.

Note that the proof of Lemma 22 only uses parts of the proof of the corresponding lemma in the undirected case [19, Lemma 6].

**Theorem 23.** DDA admits a problem kernel containing \(O(\Delta_D^5 s)\) vertices. It is computable in \(O(\Delta_D^{10}s^2 + \Delta_D^5 sn)\) time.

**Proof.** We use Algorithm 1 to compute the problem kernel. The correctness of the kernelization follows from the following two lemmas. Their proofs are, however, a straightforward adaption of the corresponding undirected counterparts [19, Lemmas 7 and 8].

Let \(D = (V, A)\) be a digraph and \(k \in \mathbb{N}\). An arc set \(S \subseteq V \times V\) is called \(k\)-insertion set for \(D\), if \(D + S\) is \(k\)-anonymous.

**Lemma 24.** If the instance \((D', k', s)\) constructed by Algorithm 2 is a yes-instance, then \((D, k, s)\) is a yes-instance.

**Proof of Lemma 24.** First, observe that if \(k \leq \beta\), then \(k' = k\) and each \(k\)-insertion set for \(D'\) is a \(k\)-insertion set for \(D\) as all blocks with less than \(\beta + 2s\) vertices remain unchanged. Hence, it remains to consider the case that \(k > \beta\) and thus \(k' = \beta\).

Let \(S'\) be an arc set such that \(|S'| \leq s\) and \(D' + S'\) is \(k'\)-anonymous. By Lemma 22, we can assume that each arc in \(S'\) has both endpoints in \(C\). We show that \(D + S'\) is \(k\)-anonymous, that is, for each block \(B_{D+S'}(t)\) we have \(|B_{D+S'}(t)| \geq k\) or \(|B_{D+S'}(t)| = 0\). To this end, we distinguish two cases on whether the corresponding block in \(D' + S'\) is empty or contains at least \(k'\) vertices.

First, consider the case \(|B_{D+S'}(t)| = 0\). Since \(|S'| \leq s\), it follows that \(|B_D(t)| \leq 2s\). By Lines 11 and 13, it follows that \(D\) and \(D'\) contain the same vertices of degree \(t\), that is, \(B_D(t) = B_{D'}(t)\). Hence, we have \(|B_{D+S'}(t)| = 0\).

Second, consider the case \(|B_{D+S'}(t)| \geq k\). If \(|B_D(t)| \geq k + 2s\), then \(|B_{D+S'}(t)| \geq k\) and we are done. Otherwise, by Line 13 we have \(|B_D(t)| - k = |B_{D'}(t)| - k'\). Since \(S'\) only contains arcs with both endpoints in \(C\), it follows that by inserting \(S\), the same vertices will be added and removed from \(B_D(t)\) and \(B_{D'}(t)\), that is, \(|B_{D+S'}(t)| - k = |B_{D'+S'}(t)| - k'\.
Since \( |B_{D+S}(t)| \geq k' \) it follows that \( |B_{D+S}(t)| \geq k \). Thus, \( D + S' \) is \( k \)-anonymous and \((D, k, s)\) is a yes-instance.

Lemma 25. If \((D, k, s)\) is a yes-instance, then the instance \((D', k', s)\) constructed by Algorithm 1 is a yes-instance.

Proof of Lemma 25. Observe that in the instance \((D', k', s)\) constructed by Algorithm 1 for each degree \( t \in \mathbb{N}^2 \), we have that either \( B_D(t) = B_D(t) \) (in case that \( B_D(t) \) contains no vertices, see Lines 9 and 11) or \( B_D(t) \subseteq B_D(t) \) contains at least \( \beta - 2s \) vertices \((|B_D(t)| \geq \beta - 2s = (\Delta_D + 1)2s \), see Lines 7, 9, and 13). Thus, \( D' \) contains a \((\Delta_D + 1)2s\)-block set. Since \((D, k, s)\) is a yes-instance, it follows from Lemma 5 that there is a \( k \)-insertion set \( S \) of size at most \( s \) for \( D \) such that \( S \subseteq C^2 \).

We next show that \( D' + S \) is \( k' \)-anonymous, and hence, \((D', k', s)\) is a yes-instance. First, consider the case that \( k \leq \beta \) and thus \( k' = k \). Observe that every block \( B_D(t) \) containing at least \( k + 2s \) vertices contains at least \( k' \) vertices in \( D' + S \). For every block \( B_D(t) \) containing less than \( k + 2s \) vertices it holds that \( B_D(t) = B_D(t) \) (see Line 9). Thus, \( |B_{D+S}(t)| = |B_{D+S}(t)| \) and therefore \( B_{D+S}(t) \) fulfills the \( k \)-anonymity requirement.

Second, consider the case that \( k > \beta \) and thus \( k' = \beta \). Let \( B_D(t) \) be some block of \( D' \). We show that \( |B_{D+S}(t)| = 0 \) or \( |B_{D+S}(t)| \geq k' \). If \( |B_D(t)| \leq 2s \), then \( B_D(t) = B_D(t) \) (see Line 11). Hence, \( B_{D+S}(t) = B_{D+S}(t) = 0 \) since \( k > \beta > 4s \). If \( |B_D(t)| > 2s \), then \( |B_D(t)| > k - 2s \) (see Line 7) and thus \( |B_D(t)| = \beta + \min\{2s, (|B_D(t)| - k)\} \) (see Line 13). Observe that \( |B_{D+S}(t)| - |B_D(t)| = |B_{D+S}(t)| - |B_D(t)| \), and thus,

\[
|B_{D+S}(t)| = (|B_{D+S}(t)| - |B_D(t)|) + |B_D(t)|. \tag{2}
\]

Since \( |S| \leq s \), we have \( |B_{D+S}(t)| - |B_D(t)| \geq -2s \). We now distinguish the two cases \( |B_D(t)| - k \geq 2s \) and \( |B_D(t)| - k < 2s \). In the first case, it follows that \( |B_D(t)| = \beta + 2s \) and from eq. 2 it follows

\[
|B_{D+S}(t)| \geq -2s + \beta + 2s = \beta = k'.
\]

In the second case, it follows that \( |B_D(t)| = \beta + |B_D(t)| - k \) (see Line 13). Observe that \( |B_{D+S}(t)| \geq k \) since \( |B_D(t)| > k - 2s \). From eq. (2) we conclude that

\[
|B_{D+S}(t)| \geq k - |B_D(t)| + \beta + |B_D(t)| - k = \beta = k'.
\]

The size of the kernel can be seen as follows: For each of the at most \((\Delta_D + 1)^2\) different blocks in the input graph \( D \), the algorithm keeps at most \( \beta + 2s = (\Delta_D + 3)2s \) vertices in the set \( C \) (see Lines 6, 11, and 13). Thus, \(|C| \in O(\Delta_D^2 s)\). The number of newly added vertices in Lines 17 to 23 is at most \( \max\{\Delta_D^2 \cdot |C|, k', \Delta_D + s + 1\} \). Hence, \(|P| \in O(\Delta_D^3 s)\) and thus the instance produced by Algorithm 1 contains at most \( O(\Delta_D^3 s) \) vertices.

The running time can be seen as follows: Using bucket sort, one can lexicographically sort the \( n \) vertices by degree in \( O(n) \) time. Furthermore, in the same time one can create \((\Delta_D + 1)^2\) lists—each list containing the vertices of some degree \( t \in \mathbb{N}^2 \). Then, the selection of the \( O(\Delta_D^2 s) \) vertices of \( C \) can be done in \( O(\Delta_D^3 s n) \) time. Clearly, inserting the vertices in \( P \) can be done in \( O(\Delta_D^3 s) \) time. Finally, adding the arcs between the vertices in \( P \) (Lines 25 and 26) takes \( O(\Delta_D^{10} s^2) \) time.
In contrast to both number problems in Sections 4.1 and 4.2, we were unable to find a polynomial-time algorithm for the number problem for DDA, which is defined as follows:

**Numbers Only Digraph Degree Anonymity (#DDA)**

**Input:** A sequence \((c_1, d_1), \ldots, (c_n, d_n)\) of \(n\) nonnegative integer tuples, three positive integers \(s, k, \) and \(\xi\).

**Question:** Is there a sequence \(\sigma' = (c'_1, d'_1), \ldots, (c'_n, d'_n)\) such that (i) \(\sum_{i=1}^{n} c'_i - c_i = \sum_{i=1}^{n} d'_i - d_i = s\), (ii) \(c_i \leq c'_i \leq \xi\), and \(d_i \leq d'_i \leq \xi\) for all \(1 \leq i \leq n\), and for each tuple in \(\sigma'\), there are at least \(k - 1\) other tuples with the same values?

We can show that #DDA is weakly NP-hard by a polynomial-time many-one reduction from Partition.

**Theorem 26.** #DDA is (weakly) NP-hard even if \(k = 2\).

**Proof.** Given a multiset \(A = \{a_1, \ldots, a_n\}\) of positive integers that sum up to \(2B\), Partition asks whether there is a subset \(A' \subset A\) whose elements sum up to exactly \(B\). Observe that we can assume without loss of generality that each integer in \(A\) is less than \(B\) (otherwise we could solve the instance in polynomial time).

We create the following #DDA-instance with \(s := B, k := 2, \xi := 2B(n + 1)^3\) and the sequence \(\sigma\) containing the following tuples. For each \(a_i \in A\) create five tuples: one tuple \(x_i\) of type \((2B(i+1) - a_i, 0)\), one block \(X_i\) that contains two tuples of type \((2B(i+1), 0)\), and one block \(X'_i\) that contains two tuples of type \((2B(i+1) - a_i, a_i)\). This completes the construction.

We show that there is a subset \(A' \subset A\) whose elements sum up to exactly \(B\) if and only if there is a sequence \(\sigma' = (c'_1, d'_1), \ldots, (c'_n, d'_n)\) that fulfills Conditions (i)–(iii) of #DDA.

First, assume that there is some \(A' \subset A\) and \(\sum_{a \in A'} a = B\). Then, we obtain the desired sequence \(\sigma'\) by first copying \(\sigma\) and changing \(x_i\) as follows: For each \(a_i \in A'\) change the tuple \(x_i\) from type \((2B(i+1) - a_i, 0)\) to type \((2B(i+1), 0)\) and for each \(a_i \notin A'\) change the tuple \(x_i\) from type \((2B(i+1) - a_i, 0)\) to type \((2B(i+1) + 1) - a_i, a_i)\). It is not hard to verify that this \(\sigma'\) is indeed a solution: For Condition (i), observe that \(\sum_{i=1}^{n} (c'_i - c_i) = d'_i - d_i = B = \sum_{i=1}^{n} (d'_i - d_i)\) since the elements in \(A'\) as well as the elements in \(A \setminus A'\) sum up to \(B\). Condition (ii) is clearly ensured by construction of \(\sigma'\). For Condition (iii), note that in sequence \(\sigma'\) block \(X_i\) contains either \(k = 2\) tuples (if \(a_i \notin A'\)) or \(k + 1\) tuples (if \(a_i \in A'\)) and, analogously, note that block \(X'_i\) contains either \(k = 2\) tuples (if \(a_i \in A'\)) or \(k + 1\) tuples (if \(a_i \notin A'\)); \(\sigma'\) contains no further tuples.

Second, assume that there is a sequence \(\sigma' = (c'_1, d'_1), \ldots, (c'_n, d'_n)\) that is a solution for our constructed #DDA instance. First note that \(\sigma'\) does not differ from \(\sigma\) “a lot” in the following sense. Since \(s = B\) and \(k = 2\), in sequence \(\sigma'\) the first component and the second component of all tuples can in total be increased by at most \(B\), respectively. Next, observe that each tuple \(x_i\) must either be of type \((2B(i+1), 0)\) or of type \((2B(i+1) - a_i, a_i)\), since every other tuple is too far away (recall that \(a < B\) for all \(a \in A\)). This means that each tuple \(x_i\) contributes with \(a_i\) to the total sum over the differences in either the first component \((\sum_{i=1}^{n} c'_i - c_i)\), or the second component \((\sum_{i=1}^{n} d'_i - d_i)\). Since \(\sum_{a \in A} a = 2B\), it follows that the tuples \(x_i\) require at least a budget of \(B\) in either the first or the second component. Let \(A' := \{a_i \mid x_i\text{ is of type } (2B(i+1), 0)\text{ in } \sigma'\}\). We show that \(\sum_{a \in A'} a = B\). Assume towards a contradiction that \(\sum_{a \in A'} a \neq B\). Since \(\sum_{i=1}^{n} (c'_i - c_i) = \sum_{a \in A'} a\) and

\[\text{We do not need to restrict the numbers, hence, setting } \xi = \infty \text{ or any value } \geq 2B(n+1) \text{ works.}\]
\[ \sum_{i=1}^{n}(d'_i - d_i) = \sum_{a \in A'} a, \text{ either } \sum_{i=1}^{n}(c'_i - c_i) \text{ or } \sum_{i=1}^{n}(d'_i - d_i) \text{ would be greater than } B - \text{a contradiction to our budget.} \]

Note that the hardness from Theorem 26 does not translate to instances of \#DDA originating from digraph degree sequences because in such instances all numbers in the sequence are polynomially bounded in the number \( n \) of tuples. Since there are pseudo-polynomial-time algorithms for PARTITION, Theorem 26 leaves open whether \#DDA is strongly NP-hard or can be solved in polynomial time for instances coming from digraphs.

To again apply our framework, we show that \#DDA is fixed-parameter tractable with respect to \( \xi \). To this end, we develop an integer linear program that contains at most \( O(\xi^4) \) integer variables and apply the famous result due to Lenstra [24].

**Theorem 27.** \#DDA is fixed-parameter tractable with respect to \( \xi \).

**Proof.** Let \((\sigma, k, s)\) be an instance of \#DDA. The key idea is that knowing how many tuples of type \( t \) in \( \sigma \) are transformed into a tuples of type \( t' \) in \( \sigma' \) for each pair \( \{t, t'\} \) of tuples is sufficient to describe a solution of our \#DDA instance. To this end, observe that there are at most \((\xi + 1)^2 \) tuple blocks in \( \sigma \) and in \( \sigma' \), respectively.

We describe an integer linear problem and create one variable \( x_{t,t'} \) for each pair \( t, t' \in \{0,\ldots,\xi\}^2 \) which denotes the number of tuples of type \( t \) in sequence \( \sigma \) that become tuples of type \( t' \) in sequence \( \sigma' \). We further use the binary variables \( u_t \) for each \( t \in \{0,\ldots,\xi\}^2 \) being 1 if and only if some tuple of type \( t \) is used in the solution, that is, there is at least one tuple of type \( t \) in \( \sigma' \). We add a set of constraints ensuring that all tuples from \( \sigma \) appear in \( \sigma' \):

\[ \forall t \in \{0,\ldots,\xi\}^2 : \sum_{t' \in \{0,\ldots,\xi\}^2} x_{t,t'} = \lambda_\sigma(t). \]

Then, we ensure that (i) holds by:

\[ \forall (t_1, t_2), (t'_1, t'_2) \in \{0,\ldots,\xi\}^2 \text{ with } t'_1 < t_1 \text{ or } t'_2 < t_2 : x_{(t_1, t_2), (t'_1, t'_2)} = 0. \]

We ensure that (ii) holds by:

\[ \sum_{(t_1, t_2), (t'_1, t'_2) \in \{0,\ldots,\xi\}^2} (t'_1 - t_1) \cdot x_{(t_1, t_2), (t'_1, t'_2)} = s \]

and by:

\[ \sum_{(t_1, t_2), (t'_1, t'_2) \in \{0,\ldots,\xi\}^2} (t'_2 - t_2) \cdot x_{(t_1, t_2), (t'_1, t'_2)} = s. \]

We ensure that (iii) holds by:

\[ \forall t' \in \{0,\ldots,\xi\}^2 : \sum_{t \in \{0,\ldots,\xi\}^2} x_{t,t'} + k \cdot (1 - u_{t'}) \geq k. \]

Finally, we add the following constraint set to ensure consistency between the \( u_t \) and \( x_{t,t'} \) variables:

\[ \forall t' \in \{0,\ldots,\xi\}^2 : \sum_{t \in \{0,\ldots,\xi\}^2} x_{t,t'} \leq u_{t'} \cdot n. \]

Finally, fixed-parameter tractability follows by the famous result of Lenstra [24] (later improved by Frank and Tardos [9], Kannan [21]) that says that an ILP with \( \rho \) variables and \( \ell \) input bits can be solved in \( O(\rho^{3.5\rho + o(\rho)} \ell) \) time. \( \square \)
Combining Theorems 6, 10 and 27 yields fixed-parameter tractability for DDA with respect to $\Delta^*$. Hartung et al. [19] showed fixed-parameter tractability with respect to $\Delta_D$ in the undirected setting. This result was based on showing that $\Delta^* \leq \Delta_D^2 + 5\Delta_D^2 + 2$. In the directed setting, however, we can only show that $\Delta^* \leq 4k(\Delta_D + 2)^2$.

**Lemma 28.** Let $D$ be a digraph and let $S$ be a minimum size arc set such that $D + S$ is $k$-anonymous. Then the maximum degree in $D + S$ is at most $4k(\Delta_D + 2)^2 + \Delta_D$.

Proof. Let $D = (V, A)$ be a digraph with maximum degree $\Delta_D$ and let $k$ be a positive integer. An arc set $S \subseteq V \times V$ is called $k$-insertion set for $D$, if $D + S$ is $k$-anonymous. Further, let $S \subseteq V \times V$ be a minimum size $k$-insertion set. We will show that if $|V(S)| \geq 4k(\Delta_D + 2)^2$, then the maximum degree in $D + S$ is at most $\Delta_D + 2$. If $|V(S)| < 4k(\Delta_D + 2)^2$, then the degree in $D + S$ is clearly at most $4k(\Delta_D + 2)^2 + \Delta_D$.

Now suppose that $|V(S)| \geq 4k(\Delta_D + 2)^2$ and assume towards a contradiction that $D + S$ has a maximum degree $\Delta_{D+S} > \Delta_D + 2$. We next construct a smaller $k$-insertion set $S'$ in two steps. In the first step, we define for each vertex $v \in V$, a target degree $\tau(v) = (\tau^-(v), \tau^+(v))$ such that the following (and further conditions that are discussed later) holds:

(a) $\deg_D^-(v) \leq \tau^-(v) \leq \Delta_D + 2$,

(b) $\deg_D^+(v) \leq \tau^+(v) \leq \Delta_D + 2$, and

(c) the multiset $\sigma(\tau) := \{\tau(v) \mid v \in V\}$ is $k$-anonymous, that is $\lambda_{\sigma(\tau)}(\tau(v)) \geq k$ for each $v \in V$.

As a second step, we use Lemma 7 to provide an arc set $S'$ such that $\sigma(D + S') = \sigma(\tau)$. Since $\sigma(\tau)$ is $k$-anonymous, it follows that $S'$ is a $k$-insertion set and we will show that $|S'| < |S|$.

We now give a detailed description of the two steps and start with defining the target degree function $\tau$ as follows

$$
\tau(v) := (\min\{\Delta_D + 1, \deg_{D+S}^-(v)\}, \min\{\Delta_D + 1, \deg_{D+S}^+(v)\}).
$$

Observe that $\tau$ satisfies the above three Conditions (a) to (c). Furthermore, we have

$$
\sum_{v \in V} \deg_{D+S}^+(v) > \sum_{v \in V} \tau^+(v)
$$

since the maximum degree in $D + S$ is larger than $\Delta_D + 2$. If we can realize the target degrees $\tau$ with a $k$-insertion set $S'$, then it follows that $|S'| < |S|$.

To apply Lemma 7 with $\Delta_{D'} := \Delta_D + 2$, $x_i := \tau^+(v_i) - \deg_{D}^+(v_i)$, and $y_i := \tau^+(v_i) - \deg_{D}^-(v_i)$ we need to satisfy Conditions (I) to (IV) of Lemma 7. By assumption, it holds $\Delta_{D'} = \Delta_D + 2 < \Delta_{D+S} \leq |V| - 1$. Hence, Condition (I) is fulfilled. Moreover, $\tau^-(v) \leq \Delta_{D'}$ and $\tau^+(v) \leq \Delta_{D'}$ holds for all $v \in V$. Conditions (III) and (IV) are thus also satisfied. However, we also need to ensure $\sum_{i=1}^n x_i = \sum_{i=1}^n y_i$ (Condition (IV)), that is, we need to ensure that $\tau$ changes the inddegrees and outdegrees of the vertices in $D$ by the same overall amount. This might not be true as we changed the inddegrees and outdegrees independently.

To overcome this problem, we subsequently adjust $\tau$ again.

Assume without loss of generality that compared to $D + S$ the target degree function $\tau$ reduced more inddegrees than outdegrees, that is,

$$
\sum_{v \in V} (\tau^+(v) - \deg_{D}^+(v)) > \sum_{v \in V} (\tau^-(v) - \deg_{D}^-(v)).
$$
Denote by \( \text{diff}_\tau \) the difference between the two sums, that is,
\[
\text{diff}_\tau := \sum_{v \in V} \left( (\tau^+(v) - \deg_D^+(v)) - (\tau^-(v) - \deg_D^-(v)) \right) \\
= \sum_{v \in V} (\tau^+(v) - \tau^-(v) + \deg_D^-(v) - \deg_D^+(v)) \\
= \sum_{v \in V} (\tau^+(v) - \tau^-(v)) + \sum_{v \in V} (\deg_D^+(v) - \deg_D^-(v)) \\
= \sum_{v \in V} (\tau^+(v) - \tau^-(v)).
\]

Further, denote by \( B_\tau(\tau(v)) \) the block of \( v \), that is the set vertices having the same target degree as \( v \). In the final adjustment of \( \tau \) we need \( \text{diff}_\tau \) to be at least \( k \) and at most \( 3k \). Hence, if \( \text{diff}_\tau < k \), then we adjust \( \tau \) as follows: Pick an arbitrary vertex \( v \) such that the outdegree of \( v \) in \( D + S \) is larger than \( \Delta_D + 1 \). Observe that such a vertex must exist: We assumed to reduce the indegrees more than the outdegrees (thus \( 0 < \text{diff}_\tau \)), hence we reduced the indegrees of the vertices of at least one block, that is, of at least \( k \) vertices. Since \( \text{diff}_\tau < k \) it follows that we also reduced the outdegrees of at least one block and thus, such a vertex \( v \) exists. If the block of \( v \) contains at least \( 2k \) vertices, then increase the target outdegree of exactly \( k \) of these vertices by one. Otherwise, if the block contains less than \( 2k \) vertices, then increase the target outdegree of all these vertices by one. It follows that \( \text{diff}_\tau > k \).

Furthermore, observe that \( \sum_{v \in V} \tau^+(v) \leq \sum_{v \in V} \deg_D^+ S_{D+1}(v) \), that is, after realizing the target degrees \( \tau \), the corresponding \( k \)-insertion set \( S' \) is still smaller than \( S \).

In the following, we increase the indegrees in two rounds. Observe that if we do not increase outdegrees, then it still holds that \( |S'| < |S| \). In the first round, while \( \text{diff}_\tau \geq 3k \) do the following:

1. Pick an arbitrary vertex with \( \tau^-(v) \leq \Delta_D \).
2. If \( |B_\tau(\tau(v))| \leq 2k \), then increase the target indegree \( \tau^-(u) \) by one for each \( u \in B_\tau(\tau(v)) \).
3. Else, it follows that \( |B_\tau(\tau(v))| > 2k \). Let \( B' \subseteq B_\tau(\tau(v)) \) be an arbitrary subset of size exactly \( k \) and increase the target indegree \( \tau^-(u) \) by one for each \( u \in B' \).

Observe that in Point 2 as well as in Point 3 we increase the target indegree of at least \( k \) vertices that have the same target degree. Furthermore, in Point 3 we ensure that at least \( k \) vertices with the original target degree remain. Hence, the (changed) multiset \( \sigma(\tau) \) is still \( k \)-anonymous. Furthermore, it is easy to verify that the maximum target indegree is at most \( \Delta_D + 1 \). Finally, observe that we decrease \( \text{diff}_\tau \) in each iteration by at most \( 2k \) and, hence, we have \( \text{diff}_\tau \geq k \).

In the second round, we have that \( k \leq \text{diff}_\tau < 3k \). We simply pick a block \( B_\tau(\tau(v)) \) with at least \( 4k \) vertices and increase the target indegree of exactly \( \text{diff}_\tau \) vertices. Since \( |V(S)| \geq 4k(\Delta_D + 2)^2 \) and there are at most \( (\Delta_D + 2)^2 \) different degrees in \( \tau \) (in- and outdegrees between \( 0 \) and \( \Delta_D + 1 \)), it follows that there exists such a block of size at least \( 4k \). Furthermore, observe that after this change in the second round \( \sigma(\tau) \) is still \( k \)-anonymous and the maximum target indegree is at most \( \Delta_D + 2 \). Hence, the adjusted target degree function \( \tau \) fulfills Conditions [1] to [LV] of Lemma 7.

It remains to show the last condition in Lemma 7 that is, Condition [V] stating \( s = \sum_{i=1}^n x_i \geq 2\Delta_{D'}^2 + \Delta_{D'} \). Due to the definition of \( \tau \) (see eq. [3]), it follows that we only
decreased the degrees of vertices with in- or outdegree more than $\Delta D + 1$ in $D + S$. Since the target degrees of these vertices is at least $\Delta D + 1$ (the later changes to $\tau$ only increased some degrees), it follows that $V(S)$ is exactly the set of vertices whose target indegree (outdegree) is larger than their indegree (outdegree) in $D$. Hence,

$$\sum_{v \in V} \tau^+(v) - \deg^+_D(v) \geq |\{v \in V \mid \tau^+(v) > \deg^+_D(v)\}| = |V(S)| \geq 4k(\Delta D + 2)^2.$$  

Since $\Delta_D^* = \Delta_D + 2$ it follows that Condition $[\text{V}]$ is indeed fulfilled. Thus, the set $S' := A'$ realizing $\tau$ is a $k$-insertion set of size less than $|S|$; a contradiction to the fact that $S$ is a minimum size $k$-insertion set for $D$.

Consequently, combining Theorems 6, 10 and 27 and Lemma 28 we obtain the following.

**Corollary 29.** DDA is fixed-parameter tractable with respect to $\Delta^*$ and $(k, \Delta_D)$.

It remains open whether DDA is fixed-parameter tractable with respect to $\Delta_D$. We remark that the problems DDCONC and DDSEQC are both NP-hard for $\Delta_D = 3$. This follows from an adaption of the construction given by Millani [29, Theorem 3.2].

5 Conclusion

We proposed a general framework for digraph degree sequence completion problems and demonstrated its wider applicability in case studies. Surprisingly, the presumably more technical case of digraphs allowed for some elegant tricks (based on flow computations) that seems not to work for the presumably simpler undirected case. Once having established the framework (see Section 3), the challenges then associated with deriving fixed-parameter tractability and kernelizability results usually boil down to the question for the polynomial-time solvability of an associated, problem-specific number problem. While in most cases we could develop polynomial-time solutions for these number problems, in case of DIGRAPH DEGREE ANONYMITY the polynomial-time solvability of the associated number problem remains open. Another widely open field is to attack weighted versions of our problems. Finally, we believe that due to the fact that many real-world networks are inherently directed (compare relations such as “follower” or “like” or “cites”) more studies (e.g., exploiting special graph properties) of digraph sequence completion problems are desirable.

References

[1] J. Bang-Jensen, A. Frank, and B. Jackson. Preserving and increasing local edge-connectivity in mixed graphs. SIAM Journal on Discrete Mathematics, 8(2):155–178, 1995.
[2] J. Bang-Jensen, J. Huang, and X. Zhu. Completing orientations of partially oriented graphs. CoRR abs/1509.01301, 2015.
[3] C. Bazgan, R. Bredereck, S. Hartung, A. Nichterlein, and G. J. Woeginger. Finding large degree-anonymous subgraphs is hard. Theoretical Computer Science, 622:90–110, 2016.
[4] J. Casas-Roma, J. Herrera-Joancomartí, and V. Torra. An algorithm for $k$-degree anonymity on large networks. In Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM ’13), pages 671–675. ACM Press, 2013.
[5] W.-K. Chen. On the realization of a \((p, s)\)-digraph with prescribed degrees. *Journal of The Franklin Institute*, 281(5):406–422, 1966.

[6] S. Chester, B. Kapron, G. Srivastava, and S. Venkatesh. Complexity of social network anonymization. *Social Network Analysis and Mining*, 3(2):151–166, 2013.

[7] F. Dorn, H. Moser, R. Niedermeier, and M. Weller. Efficient algorithms for eulerian extension and rural postman. *SIAM Journal on Discrete Mathematics*, 27(1):75–94, 2013.

[8] P. Erdős and T. Gallai. Graphs with prescribed degrees of vertices (in Hungarian). *Matematikai Lapok*, 11:264–274, 1960.

[9] A. Frank and É. Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.

[10] V. Froese, A. Nichterlein, and R. Niedermeier. Win-win kernelization for degree sequence completion problems. *Journal of Computer and System Sciences*, 2016. Available online.

[11] D. Fulkerson. Zero-one matrices with zero trace. *Pacific Journal of Mathematics*, 10(3):831–836, 1960.

[12] D. Gale. A theorem on flows in networks. *Pacific Journal of Mathematics*, 7:1073–1082, 1957.

[13] P. A. Golovach. Editing to a graph of given degrees. *Theoretical Computer Science*, 591:72–84, 2015.

[14] P. A. Golovach and G. B. Mertzios. Graph editing to a given degree sequence. *CoRR* abs/1601.03174, 2016. Accepted at *CSR ’16*.

[15] G. Gutin and A. Yeo. Some parameterized problems on digraphs. *Comput. J.*, 51(3):363–371, 2008.

[16] S. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. I. *Journal of SIAM*, 10(3):496–506, 1962.

[17] S. Hartung and A. Nichterlein. NP-hardness and fixed-parameter tractability of realizing degree sequences with directed acyclic graphs. *SIAM Journal on Discrete Mathematics*, 29(4):1931–1960, 2015.

[18] S. Hartung, C. Hoffmann, and A. Nichterlein. Improved upper and lower bound heuristics for degree anonymization in social networks. In *Proceedings of the 13th International Symposium on Experimental Algorithms (SEA ’14)*, volume 8504 of *LNCS*, pages 376–387. Springer, 2014.

[19] S. Hartung, A. Nichterlein, R. Niedermeier, and O. Suchý. A refined complexity analysis of degree anonymization in graphs. *Information and Computation*, 243:249–262, 2015.

[20] J. E. Hopcroft and R. M. Karp. An \(n^{5/2}\) algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[21] R. Kannan. Minkowski’s convex body theorem and integer programming. *Mathematics of Operations Research*, 12:415–440, 1987.

[22] D. Kleitman and D. Wang. Algorithms for constructing graphs and digraphs with given valences and factors. *SIAM Journal on Discrete Mathematics*, 6(1):79–88, 1973.

[23] M. Koseler. Kernelization for degree-constraint editing on directed graphs, November 2015. URL [http://fpt.akt.tu-berlin.de/publications/theses/BA-marcel-koseler.pdf](http://fpt.akt.tu-berlin.de/publications/theses/BA-marcel-koseler.pdf). Bachelor thesis.

[24] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[25] J. Leskovec and E. Horvitz. Planetary-scale views on a large instant-messaging network. In *Proceedings of the 17th International Conference on World Wide Web (WWW ’08)*,
[26] K. Liu and E. Terzi. Towards identity anonymization on graphs. In Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD ’08, pages 93–106. ACM, 2008.

[27] X. Lu, Y. Song, and S. Bressan. Fast identity anonymization on graphs. In Proceedings of the 23rd International Conference on Database and Expert Systems Applications (DEXA ’12), Part I, volume 7446 of LNCS, pages 281–295. Springer, 2012.

[28] L. Mathieson and S. Szeider. Editing graphs to satisfy degree constraints: A parameterized approach. Journal of Computer and System Sciences, 78(1):179–191, 2012.

[29] M. G. Millani. Algorithms and complexity for degree anonymization in directed graphs, March 2015. URL http://fpt.akt.tu-berlin.de/publications/theses/BA-marcelo-millani.pdf. Bachelor thesis.

[30] H. Moser and D. M. Thilikos. Parameterized complexity of finding regular induced subgraphs. Journal of Discrete Algorithms, 7(2):181–190, 2009.

[31] J. B. Orlin. Max flows in $o(nm)$ time, or better. In Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC ’13), pages 765–774. ACM, 2013.

[32] M. Weller, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. On making directed graphs transitive. Journal of Computer and System Sciences, 78(2):559–574, 2012.