Graph Editing to a Given Neighbourhood Degree List is Fixed-Parameter Tractable

by

Vijay Subramanya

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2016

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Graph editing problems have a long history and have been widely studied [35, 53] , with applications in biochemistry [23] and complex network analysis [5]. They generally ask whether an input graph can be modified by inserting and deleting vertices and edges to a graph with the desired property [5, 22]. We consider the problem Graph-Edit-to-NDL (GEN) where the goal is to modify to a graph with a given neighbourhood degree list (NDL). The NDL lists the degrees of the neighbours of vertices in a graph, and is a stronger invariant than the degree sequence, which lists the degrees of vertices.

We show Graph-Edit-to-NDL is NP-complete and study its parameterized complexity. In parameterized complexity, a problem is said to be fixed-parameter tractable with respect to a parameter if it has a solution whose running time is a function that is polynomial in the input size but possibly superpolynomial in the parameter.

Golovach and Mertzios [22] studied editing to a graph with a given degree sequence and showed the problem is fixed-parameter tractable when parameterized by $\Delta + \ell$, where Δ is the maximum degree of the input graph and ℓ is the number of edits. We prove Graph-Edit-to-NDL is fixed-parameter tractable when parameterized by $\Delta + \ell$.

Furthermore, we consider a harder problem Constrained-Graph-Edit-to-NDL (CGEN) that imposes constraints on the NDLs of intermediate graphs produced in the sequence. We adapt our FPT algorithm for Graph-Edit-to-NDL to solve Constrained-Graph-Edit-to-NDL, which proves Constrained-Graph-Edit-to-NDL is also fixed-parameter tractable when parameterized by $\Delta + \ell$.

Our results imply that, for graph properties that can be expressed as properties of NDLs, editing to a graph with such a property is fixed-parameter tractable when parameterized by $\Delta + \ell$. We show that this family of graph properties includes some well-known graph measures used in complex network analysis.

Acknowledgements

I am grateful to my supervisor, Professor Naomi Nishimura, for her constant support, guidance, and, above all, her patience. This work would not have been possible without her excellent feedback and suggestions.

I thank Professor Therese Biedl and Professor Anna Lubiw for agreeing to read the thesis and providing valuable comments.

I also thank my family for their love and my friends, both in Waterloo and back home in India, with whom I have had great fun over the years.

Finally, I'm grateful for cricket and the unparalleled joy it has given me.

Table of Contents

Abstract **iii**

Acknowledgements **iv**

1 Introduction **2**
1.1 Parameterized complexity . . . 3
1.2 Our contributions . . . 3
1.3 Our method . . . 4
1.4 Organization . . . 4

2 Literature Survey **6**
2.1 Graph editing . . . 6
2.2 Degree-based graph invariants . . . 7
2.3 Editing to a graph specified by a degree-based graph invariant . . . 8
2.4 Reconfiguration problems . . . 9

3 Basic Definitions **11**
3.1 Graph definitions . . . 11
3.2 List definitions . . . 12
3.3 Data structure . . . 12
3.4 Neighbourhood Degree List . . . 12
3.4.1 Graph properties expressed as Young properties of NDLs . . . 13
3.5 Graph edit . . . 14
3.6 NDL graph edit . . . 16
3.6.1 List-attributed edit pair . . . 17
3.6.2 NDL graph edit . . . 18
3.6.3 Complete LA edit pairs . . . 25
3.7 Time complexity of checking list-isomorphism and degree conditions . . . 28
3.8 Sequences of edit pairs and LA edit pairs . . . 30

3.9 Problem statement 31

4 Complexity Results **32**

5 Preliminary Results **36**
5.1 Restatement of GEN and CGEN in terms of NDL graph edit 36
5.2 NDL of $(G, \mathcal{E})$-produced graphs 39

6 Provisional Completeness of an LA Edit Pair Sequence **42**
6.1 CHECK-PROVISIONAL-COMPLETENESS procedure 42

7 Determining the completeness of an LA edit pair sequence in a graph **48**
7.1 Merging LA graphs 49
7.2 Antecedent of a merge graph 52
7.3 Origin graph 58
7.3.1 Antecedent and merge sequences 59
7.3.2 Origin graph 62

8 FPT Algorithms for GEN and CGEN **71**
8.1 FPT algorithm for GEN 71
8.1.1 Bounding the search space 71
8.1.2 SOLVE-GEN procedure 72
8.2 FPT algorithm for CGEN 73
8.3 Implications 74

9 Conclusion **76**

Notation **78**

References **82**

Chapter 1

Introduction

Graph editing problems ask whether an input graph can be modified to a graph with a given property using graph edit operations [35, 53, 40]. The permitted edit operations are usually chosen from among vertex and edge insertions and deletions [5, 22, 39], but more complex operations such as edge contraction (removal of an edge and identifying the vertices it connected) [3] and edge flipping (deletion of an edge and insertion of a different edge in a graph such that the resulting graph is in the same graph class) [7] have been considered. Graph editing problems have numerous applications, which are mainly determined by the graph property: editing to an interval graph is used to correct errors in DNA sequence fragmentation [21], editing to a planar graph has found application in graph drawing [51], and editing to satisfy anonymity constraints is useful in complex network analysis [9].

In this thesis, we consider insertions and deletions of vertices and edges as the permitted graph edit operations and a graph property that is defined on the degrees of vertices. More specifically, our graph property is a graph invariant called the *neighbourhood degree list* (NDL) [4], which is a list that contains lists of degrees of neighbours of the vertices of a graph. Our problem, Graph-Edit-to-NDL, asks whether an input graph can be modified to a graph with a given NDL by applying at most ℓ graph edits. Furthermore, we consider a harder problem Constrained-Graph-Edit-to-NDL that imposes constraints on the NDLs of intermediate graphs produced in the sequence.

Of degree-based graph invariants, the degree sequence, which lists the degrees of the vertices of a graph, is one of the simplest and is well-studied [15]. The NDL, on the other hand, was introduced recently [4] and is a stronger invariant than the degree sequence because the degree sequence of a graph is implicitly given by its NDL. However, the NDL is a weaker invariant than the *deck* of a graph [26], which is the set of all induced

subgraphs obtained by deleting exactly one vertex from the graph. Barrus and Donovan [4] gave a counting argument to show that the NDL of a graph is determined by its deck. Therefore, the Graph Reconstruction Conjecture due to Kelly [33] and Ulam [52], which claims that every graph is uniquely determined, up to isomorphism, by its deck, is true for the class of graphs that are uniquely determined, up to isomorphism, by their NDLs.

To the best of our knowledge, the problem of editing to a graph with a given deck has not been considered. However, recently, Golovach and Mertzios [22] studied editing to a graph with a given degree sequence and showed that it is NP-complete. They further analyzed the problem through the lens of parameterized complexity, and gave an FPT solution for the combined parameter maximum degree and the number of edits.

1.1 Parameterized complexity

Although no strictly polynomial-time solutions exist for NP-hard problems (unless P=NP), restricting the super-polynomial complexity to parameters that are small compared to the input size is of practical concern. *Parameterized complexity* [13] analyzes a problem in two dimensions: the size of the instance $|I|$ and a fixed parameter k. A problem is *fixed-parameter tractable* if it is solvable in $f(k) \cdot |I|^p$ time, where f is a computable function that depends only on k and p is a constant. The class FPT contains fixed-parameter tractable problems.

Kernelization is a preprocessing technique that reduces an instance I with parameter k to an instance I' with parameter k' such that $|I'| + k'$ is bounded by a function ϕ of k. The pair (I', k') is called a *kernel* of (I, k), and is a *polynomial kernel* if the function ϕ is polynomial.

Just as NP-hardness characterizes the intractability of a problem in the classical setting, W-hardness characterizes its fixed-parameter intractability with respect to a given parameter. There exists a hierarchy of classes called the *W-hierarchy* given by FPT $\subseteq W[1] \subseteq W[2] \subseteq \ldots$ XP, where XP contains the problems solvable in $\mathcal{O}(n^{f(k)})$ time for a function f of k. It is believed that FPT $\neq W[1]$.

1.2 Our contributions

We give an algorithm to show Graph-Edit-to-NDL is fixed-parameter tractable with respect to the parameter $\Delta+\ell$, where Δ is the maximum degree of the input graph and ℓ is the maximum number of graph edits. It follows from our algorithm that the harder problem Constrained-Graph-Edit-to-NDL, which constrains the NDLs

of the intermediate graphs obtained by applying the graph edit sequence to satisfy a certain property π, is in FPT for the parameter $\Delta + \ell$. Moreover, we show that both the problems are NP-complete.

Since the NDL is a stronger invariant than the degree sequence, there exist graph properties which can be expressed as properties of NDLs, but not as properties of degree sequences. Our solution to Graph-Edit-to-NDL can be adapted to solve editing to graphs with such properties. We mention three such graph properties in the domain of complex networks: neighbourhood degree anonymity, assortativity, and average nearest neighbour degree (see Examples 5, 6, and 7, respectively, in Chapter 3).

1.3 Our method

Our strategy is similar to the FPT solution for editing to a k-degree anonymous graph by Bazgan and Nichterlein [5], where a graph is said to be *k-degree anonymous* [37] if, for each vertex of the graph, there are at least $k-1$ other vertices with the same degree. Their algorithm constructs all possibilities of a "solution structure", each of which is checked to determine whether it leads to a k-degree-anonymous graph. The algorithm returns yes if and only if there exists such a solution structure which is also an induced subgraph of the input graph.

We view a graph edit as "replacing" a subgraph of a graph with a different graph. We extend this notion to NDL graph edits, which specify not only the modifications to a graph but also the changes to its NDL. In our FPT algorithm, we consider all possible sequences of NDL graph edits of length at most ℓ which lead to a graph with the desired NDL. For each such sequence, we check whether the modifications it specifies can be performed in the input graph by tracing back each modification to the input graph. This reduces the task to determining whether a certain graph is isomorphic to a subgraph of the input graph such that certain degree constraints are satisfied. We output yes if and only if there exists such a sequence of NDL graph edits.

Our algorithm also determines the NDLs of the intermediate graphs obtained by performing each sequence of NDL graph edits in the search space. This allows us to check whether the NDLs of intermediate graphs have a given property π and hence solve Constrained-Graph-Edit-to-NDL.

1.4 Organization

The rest of the thesis is organized as follows.

In Chapter 2, we provide background on our results. We briefly discuss the previous work in graph editing and the various graph invariants defined on the degrees of vertices. Then, we look at the graph editing problems where the desired graph property is defined using such a degree-based graph invariant, and how our problem differs from them.

In Chapter 3, we characterize a graph edit using pairs of graphs, which specify the "replaced" and the "replacing" subgraphs of a graph. Then, we specify an NDL graph edit by a pair of graphs with lists attributed to vertices to describe the changes to the NDL of the graph. We call such a pair of graphs a *list-attributed edit pair* (LA edit pair), and a sequence of LA edit pairs an *LA edit pair sequence*. We end the chapter with the problem statements of Graph-Edit-to-NDL and Constrained-Graph-Edit-to-NDL.

In Chapter 4, we prove the NP-completeness of Graph-Edit-to-NDL and Constrained-Graph-Edit-to-NDL by showing a reduction from Vertex-Cover.

In Chapter 5, we restate Graph-Edit-to-NDL in terms of NDL graph edits, and give a procedure to determine the NDL of the graph produced by a sequence of NDL graph edits using only the LA edit pair sequence.

In Chapter 6, we give a procedure to determine whether a given LA edit pair sequence accurately specifies the changes to the NDL of the input graph. Furthermore, we prove this is a feature of the LA edit pair sequence and is independent of the input graph.

In Chapter 7, we show that checking whether the sequence of NDL graph edits specified by an LA edit pair sequence of length at most ℓ can be performed in a given input graph is in FPT when parameterized by $\Delta + \ell$, where Δ is the maximum degree of the input graph. We reduce checking whether an NDL graph edit can be performed in an intermediate graph in the sequence to a condition on the input graph. Then, we prove the time taken for this reduction and checking the condition on the input graph is bounded by a function of $\Delta + \ell$.

In Chapter 8, we put it all together in our FPT algorithm, which checks the above conditions for each candidate LA edit pair sequence. We show that the search space of such sequences is bound by a function of $\Delta + \ell$, and so the algorithm runs in FPT time for the parameter $\Delta + \ell$.

Finally, we conclude in Chapter 9 by noting ways to strengthen our results, which include proving W-hardness of Graph-Edit-to-NDL when parameterized by Δ or ℓ individually, and exploring graph properties which can be expressed as properties of NDLs.

Chapter 2

Literature Survey

Our work broadly relates to two areas of research: (a) graph editing, and (b) degree-based graph invariants and their graph realizations. We begin the chapter with background on graph editing problems in Section 2.1. In Section 2.2, we discuss the various degree-based graph invariants that have been studied, which include degree sequences, integer-pair sequences, and neighbourhood degree lists. Later, in Section 2.3, we look at previous work in the intersection of these two areas: in short, we describe the graph editing problems stated using degree-based invariants. Finally, in Section 2.4, we discuss reconfiguration problems, which are similar in nature to graph editing problems.

2.1 Graph editing

The earliest studied graph editing problems were vertex deletion problems, which ask whether a subset of vertices can be deleted from a graph to obtain a graph with a desired property. Note that this family of problems contains well-known NP-complete problems – Vertex-Cover, for instance, asks whether a bounded number of vertices can be deleted to obtain a graph with no edges [32]. Lewis and Yannakakis [35] generalized this observation and proved that a vertex deletion problem is NP-complete if the desired graph property is nontrivial and hereditary on induced subgraphs. More recently, Hüffner et al. [30] showed that vertex deletion to a cluster graph is in FPT when parameterized by the number of vertices deleted.

Another class of graph editing problems is edge modification problems, which have received more attention than vertex deletion problems. The standard edge modification operations are edge insertion and edge deletion, although edge contraction, which identifies the end-vertices of an edge, is also considered. Watanabe et al. [53] showed that the edge deletion and edge contraction problems are NP-hard for any nontrivial

graph property characterized by a forbidden set consisting of 3-connected graphs. In fact, edge deletion problems in general tend to be NP-complete. For example, edge deletion to a cluster graph [42], bipartite graph [19], threshold graph [38], and interval graph [21] are all NP-complete. A well-studied edge insertion problem is the *sandwich* problem, which has applications in DNA mapping and parallel processing [23]. The sandwich problem asks whether inserting a bounded number of "admissible" edges in the input graph produces a graph with a given property. Golumbic et al. [23] showed that the sandwich problem is polynomial-time solvable for split graphs, cographs, and threshold graphs, but is NP-complete for Eulerian graphs and permutation graphs. Recently, Heggernes et al. [28] proved the problem for chordal graphs is in FPT when parameterized by the size of minimum vertex cover of the input graph.

The edit operations we allow are insertions and deletions of both vertices and edges. Furthermore, we allow an inserted vertex or edge to be deleted at a later time.

2.2 Degree-based graph invariants

A graph property that is invariant under isomorphism is called a *graph invariant*. A graph invariant defined on the degrees of vertices of a graph, such as the maximum (or minimum) degree, is a *degree-based graph invariant*. A value of a degree-based graph invariant is *realizable* if there exists a graph that satisfies the graph invariant, in which case we call the graph a *realization*.

A well-studied degree-based graph invariant is the *degree sequence*, which lists the vertex-degrees of the graph (see Definition 1 in Chapter 3). Much of the research on degree sequences has focused on their realizability. While both the Havel-Hakimi algorithm [27, 24] and the Erdős–Gallai theorem [8] determine the realizability of a degree sequence, there exist harder questions about the number of realizations and their properties. These problems have found applications in areas such as structural organic chemistry, where the structural isomers of a compound are given by the different realizations of a degree sequence [49], and so have attracted much attention. One such problem is to determine whether a degree sequence has a unique realization up to isomorphism, and if so, to find the realization. In early work, Li [36] gave a few characterizations of degree sequences that have unique realizations and, later, Hammer et al. [25] proved that the degree sequences of threshold graphs have unique realizations.

Patrinos and Hakimi [46] introduced the *integer-pair sequence* of a graph, where each pair of an integer-pair sequence corresponds to an edge of the graph and contains the degrees of its end-vertices. They also gave necessary and sufficient conditions for the realizability of an integer-pair sequence. Later, Das [11] characterized integer-pair sequences that have unique realizations and Achuthan [1] characterized those that have

connected realizations.

Barrus and Donovan [4] recently introduced the *neighbourhood degree list* (NDL) (see Definition 3 in Chapter 3), which provides more information about a graph than either its degree sequence or its integer-pair sequence. They gave necessary and sufficient conditions for the realizability of an NDL using a criterion given by Gale [18] and Ryser [48] for a degree sequence to have a bipartite graph as a realization. They also characterized non-isomorphic realizations of an NDL by defining an operation which on repeated application transforms one realization to another.

In addition to graph invariants, various graph measures that are relevant to particular domains are defined in terms of degrees of vertices. An example of such a graph measure is degree anonymity [37], which is useful in complex network analysis. In fact, k-degree-anonymity is one of many k-anonymity criteria for privacy preservation in social networks − others include k-neighbourhood-isomorphism [56], where the neighbourhood of any vertex is "similar" to the neighbourhoods of at least $k-1$ other vertices, and k-symmetry [54], where each vertex can be mapped to at least $k-1$ other vertices by automorphism.

While the degree anonymity of a graph is determined by its degree sequence, there exist other graph measures that can be determined by the NDL but not the degree sequence. *Assortativity* [43], which indicates the tendency of the vertices of a graph to be adjacent to vertices of similar degree, and *average nearest neighbours degree* [45], which gives the average of the mean degree of neighbours of vertices, are two examples.

2.3 Editing to a graph specified by a degree-based graph invariant

Editing problems where the desired property of the graph is specified using the degrees of its vertices have attracted attention. These problems are descended from the problem of finding an r-regular subgraph in a given graph [50, 47, 6]. Moser and Thilikos [40] were the first to formulate it as a graph editing problem, which meant they could use the number of graph edits, ℓ, as a parameter and show that the problem is in FPT when parameterized by $\ell + r$.

Froese et al. [17] considered the harder problem of editing to a graph whose degree sequence satisfies a desired property. When only edge insertions are allowed, they showed the problem is in FPT when parameterized by $\Delta + \ell$, where Δ is the maximum degree of the input graph. More recently, Golovach and Mertzios [22] studied the even harder problem of editing to a graph with a given degree sequence. When vertex deletion, edge insertion, and edge deletion are allowed, they proved that this problem

is in FPT when parameterized by $\Delta + \ell$. They also gave a polynomial kernel for the parameter $\Delta + \ell$, but if only edge insertions are allowed.

To the best of our knowledge, editing to a graph with a given integer-pair list has not been studied and we are the first to study editing to a graph with a given NDL.

Prior to Froese et al. [17], Mathieson and Szeider [39] had studied the more general problem of editing to a graph where the degree of each vertex is constrained to lie in a "degree set" of numbers that is assigned to the vertex by an input function. They called it Degree-Constraint-Editing and showed that if only vertex and edge deletions and edge insertions are permitted, the problem is in FPT for the parameter $s + \ell$, where s is the maximum number in the degree set of any vertex. Dabrowski et al. [10] considered a variant of Degree-Constraint-Editing which asks whether a planar, vertex- and edge-weighted input graph can be modified into a connected graph using only vertex and edge deletions, subject to additional constraints on the weights of vertices and edges deleted as well as on the total cost of the deletions. They gave a polynomial kernel for the problem when parameterized by the sum of the bounds on weights of deleted vertices and edges.

Although Graph-Edit-to-NDL looks similar to Degree-Constraint-Editing, they differ in the following way: Degree-Constraint-Editing specifies "local" constraints on the vertex-degrees by specifying a "degree set" for each vertex. On the other hand, Graph-Edit-to-NDL specifies a "global" constraint on the neighbourhoods of vertices. In other words, the vertex-degrees are constrained only by the requirement that the graph obtained has the given NDL.

In complex network analysis, editing to a k-degree-anonymous graph is well-studied on various combinations of edit operations [9]. Bazgan and Nichterlein [5] proved this problem is in FPT for the parameter $\Delta+\ell$ when vertex and edge insertions and deletions are allowed. Zhou and Pei [56] showed that editing to a k-neighbourhood-isomorphic graph using only edge insertions is NP-complete and gave a method for solving the problem, which they showed by empirical evaluation runs in time that depends only on Δ and ℓ.

Our results imply, for any graph measure determined by the NDL alone, editing to a graph with a given value of the graph measure is in FPT when parameterized by $\Delta + \ell$.

2.4 Reconfiguration problems

Reconfiguration problems are defined on the solution spaces of combinatorial problems. A modification operation is defined, which transforms one solution to another and defines an adjacency relation between the solutions. Now, the questions asked

include "can one solution be modified to another via a sequence of solutions?", "can any two solutions be transformed into each other?", and "what is the smallest number of modifications needed to transform one solution to another?" (see the survey by van den Heuvel [29]). The allowed modification operations in a reconfiguration problem generally depend on the nature of the combinatorial problem. For example, Vertex-Cover-Reconfiguration [41] allows only insertion and deletion of vertices, and Matching-Reconfiguration [31], which asks whether a matching of a graph can be transformed into a different matching, allows only insertion and deletion of edges. But there also exist problems where different modification operations lead to multiple reconfiguration problems. For example, three models of reconfiguration of independent sets have been studied − token jumping, token sliding, and token addition and removal −, which differ in the allowed modification operations [34].

The similarity between reconfiguration and graph editing is noticed if we view a graph editing problem as transforming a graph in one graph class to a graph in another. Graph-Edit-to-NDL, then, asks whether a given realization of one NDL can be transformed into a realization of a different NDL.

Chapter 3

Basic Definitions

3.1 Graph definitions

We refer to the text by Diestel [12] for basic graph definitions that we do not give in this section.

We consider simple, undirected, and unweighted graphs. The vertex set and the edge set of a graph G are denoted $V(G)$ and $E(G)$, respectively. Two vertices $u, v \in V(G)$ are *neighbours* in G if and only if $(u, v) \in E(G)$.

Two graphs G_1 and G_2 are *isomorphic* if and only if $|V(G_1)| = |V(G_2)|$ and there exists a bijection $f : V(G_1) \to V(G_2)$ such that $(u, v) \in E(G_1)$ if and only if $(f(u), f(v)) \in E(G_2)$ for any $u, v \in V(G_1)$.

A *path* of length t between vertices $u, v \in V(G)$ is a sequence of vertices $u, w_1, \ldots, w_{t-1}, v$, where $w_i \in V(G)$ for $1 \leq i \leq t-1$, $(u, w_1), (w_{t-1}, v) \in E(G)$, and $(w_j, w_{j+1}) \in E(G)$ for $1 \leq j \leq t-2$. For any two vertices $u, v \in V(G)$, the *distance* between u and v, denoted $dist(u, v)$, is the length of the shortest path, if a path exists, between u and v, and is infinity otherwise.

The *neighbourhood* of a vertex $v \in V(G)$ is defined as $N_G(v) = \{u \in V(G) \mid (u, v) \in E(G)\}$. The *k-neighbourhood* of v is the set of all vertices in G that lie at a distance k from v, i.e., $N_G^k(v) = \{u \in V(G) \mid dist(u, v) = k\}$.

The *degree* of a vertex $v \in V(G)$, denoted $d_G(v)$, is the size of its open neighbourhood in G, $|N_G(v)|$. Given a graph G, by $deg_i(G)$, we refer to the number of vertices of degree i in G. The maximum integer i for which $deg_i(G) > 0$ is the *maximum degree*, denoted $\Delta(G)$. Where G is obvious, we refer to its maximum degree as Δ.

A graph H is a *subgraph* of a graph G if and only if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. Furthermore, a graph H is an *induced subgraph* of G if and only if H is a subgraph of G and for every pair of vertices $u, v \in V(H)$, $(u, v) \in E(H)$ if and only if $(u, v) \in E(G)$.

Given a graph G, the *subgraph induced by a subset of vertices* $W \subseteq V(G)$ is defined as the induced subgraph H of G with $V(H) = W$, which we denote as $G[W]$.

3.2 List definitions

We define a *nonincreasing list* $\mathcal{L}$ as a sequence of integers in nonincreasing order and its *size,* $|\mathcal{L}|$, as the length of the sequence. The ith element of $\mathcal{L}$ is denoted $\mathcal{L}[i]$ for $1 \leq i \leq |\mathcal{L}|$.

The only operations that modify a nonincreasing list $\mathcal{L}$ are: (i) *list extension*, denoted $\mathcal{L} \oplus a$, which we use to add an element a to $\mathcal{L}$ such that the resulting list is a nonincreasing list, and (ii) *list contraction*, denoted $\mathcal{L} \ominus a$, which we use to remove an element a from $\mathcal{L}$.

A *Young tableau* [55] is a list of nonincreasing lists in a nonincreasing order of their sizes. Suppose two distinct nonincreasing lists $\mathcal{L}_1$ and $\mathcal{L}_2$ in a Young tableau $\mathcal{T}$ have the same size and i is the smallest index for which $\mathcal{L}_1[i] \neq \mathcal{L}_2[i]$. Then $\mathcal{L}_1$ precedes $\mathcal{L}_2$ in $\mathcal{T}$ if $\mathcal{L}_1[i] > \mathcal{L}_2[i]$; otherwise, $\mathcal{L}_2$ precedes $\mathcal{L}_1$ in $\mathcal{T}$.

The *size* of a Young tableau $\mathcal{T}$, denoted $|\mathcal{T}|$, is the number of nonincreasing lists in $\mathcal{T}$. We denote the i^{th} nonincreasing list of $\mathcal{T}$ by $\mathcal{T}[i]$ for $1 \leq i \leq |\mathcal{T}|$. Two Young tableaux $\mathcal{T}_1$ and $\mathcal{T}_2$ are equal if and only if $|\mathcal{T}_1| = |\mathcal{T}_2|$ and $\mathcal{T}_1[i] = \mathcal{T}_2[i]$ for each $1 \leq i \leq |\mathcal{T}_1|$.

We define a *Young property* as a property defined on Young tableaux. A Young property π is *verifiable in polynomial time* if determining whether a Young tableau $\mathcal{T}$ satisfies π is verifiable in time polynomial in $|\mathcal{T}|$.

3.3 Data structure

The computational model used in this work is the RAM model [2]. We store nonincreasing lists as dynamic arrays and Young tableaux as dynamic arrays of dynamic arrays. This implies that the list extension and list contraction operations take time linear in the size of the list. Furthermore, adding and removing a list from a Young tableau take time linear in the size of the Young tableau.

3.4 Neighbourhood Degree List

Two measures of a vertex defined using its neighbourhood are its degree and the list of the degrees of its neighbours, which we call the neighbourhood degree sequence

(Definition 2). Analogously, two invariants defined using the neighbourhoods of the vertices of a graph are (i) a list of the degrees of its vertices, which is called the degree sequence (Definition 1), and (ii) a list of the neighbourhood degree sequences of its vertices, which is called the neighbourhood degree list (Definition 3).

Definition 1 ([12])**.** The *degree sequence* of a graph G on n vertices, $(d_1, d_2, \ldots, d_n)$, is the nonincreasing list of the degrees of vertices of G.

Definition 2. The *neighbourhood degree sequence* (NDS) of a vertex $v \in V(G)$ of degree p, $NDS(G, v) = (d_1, d_2, \ldots, d_p)$, is a sequence of the degrees of vertices in $N_G(v)$ in nonincreasing order.

Definition 3 ([4])**.** The *neighbourhood degree list* (NDL) of a graph G is a Young tableau given by

$$NDL(G) = (NDS(G, v_1), NDS(G, v_2), \ldots, NDS(G, v_n))$$

where $V(G) = \{v_1, v_2, \ldots, v_n\}$ and $d_G(v_1) \geq d_G(v_2) \geq \cdots \geq d_G(v_n)$.

Observe that the degree sequence of a graph is given implicitly by its NDL, since the degree of a vertex is the size of its corresponding NDS in the NDL.

3.4.1 Graph properties expressed as Young properties of NDLs

A Young property π defines a class of graphs whose NDLs satisfy π. In other words, certain graph properties can be expressed as Young properties of the NDLs of graphs. Here, we give a few examples of graph measures that define such graph properties. Observe that while degree anonymity is determined by the degree sequence, the other three are defined using the degrees of neighbours of the vertices in a graph, and so are not determined by the degree sequence.

Example 4 ([37])**.** A graph is *k-degree anonymous* if, for each $v \in V(G)$, there are $k - 1$ other vertices of G whose degrees are $d_G(v)$.

Observe that the degree anonymity of a graph is determined by its degree sequence. Degree anonymity is a privacy measure in social networks, and a high degree anonymity protects against re-identification of vertices of a social network using their degrees [37].

Example 5. *Neighbourhood Degree Anonymity.* We introduce neighbourhood degree anonymity as an extension of degree anonymity. A graph G is *k-neighbourhood degree*

anonymous if, for each $v \in V(G)$, there are at least $k-1$ other vertices of G whose NDSs equal $NDS(G, v)$.

Observe that whether a graph G is k-neighbourhood degree anonymous can be determined by checking whether for each $NDS(G, v)$ in $NDL(G)$, there are at least $k-1$ other NDSs in $NDL(G)$ that are identical to $NDS(G, v)$. We note that the neighbourhood degree anonymity is a stronger privacy measure than k-degree anonymity.

Example 6. *Assortativity* [43]. Assortativity is a measure of the tendency of vertices in a graph to be adjacent to vertices of similar degree. The *assortativity* of a graph G is given by

$$A(G) = \frac{\sum_{v \in V(G)} \left((d_G(v) - \bar{d}) \cdot \sum_{u \in N_G(v)} (d_G(u) - \bar{d}) \right)}{\sum_{v \in V(G)} \left(d_G(v) \cdot (d_G(v) - \bar{d})^2 \right)}$$

where $\bar{d}$ is the average degree of the vertices of G. Given $NDL(G)$, we are given the average degree $\bar{d}$ and also the degree of each neighbour u of v, and so we can determine $A(G)$. Taking assortativity into account substantially increases the accuracy of social network models since many real-world networks show preference of vertices to be attached to vertices of similar degree [43]. For example, the assortativity of a social network helps predict the spread of a disease in the community [43].

Example 7. *Average nearest neighbours degree* [45]. The *average nearest neighbours degree* of a graph is the average of the average degrees of neighbours across all vertices.

$$\bar{k_{nn}}(G) = \frac{1}{|V(G)|} \cdot \sum_{v \in V(G)} \frac{\sum_{u \in N_G(v)} d_G(u)}{d_G(v)}$$

Since $d_G(u)$ is given by $NDL(G)$ for each neighbour u of v, we can determine $\bar{k_{nn}}(G)$ from $NDL(G)$. The average nearest neighbours degree has been used to describe the topological and dynamical properties of the Internet [45].

3.5 Graph edit

We consider four edit operations: vertex insertion (of degree zero), vertex deletion, edge insertion, and edge deletion.

Intuitively, a graph edit in a graph G can be viewed as "replacing" a (possibly empty) induced subgraph of G with a different graph. A graph edit, then, is characterized by a pair of graphs – the "replaced" induced subgraph X and the "replacing" graph Y – which specify the modifications to G.

Note that a vertex deletion also deletes the edges incident with the deleted vertex. Hence, these deleted edges must be present in X and absent from Y. This implies that the neighbours of the deleted vertex, whose degrees are modified by the vertex deletion, are in both X and Y.

Definition 8. An *edit pair* is an ordered pair of graphs (X, Y) such that one of the following is true:

1. *Vertex insertion:* $V(X) = \emptyset$ and $V(Y) = \{y\}$;
2. *Vertex deletion:* $V(X) = \{z\} \cup V(Y)$ such that
   (a) $N_X(z) = V(Y)$ and
   (b) $E(Y) = E(X) \setminus \{(z, p) \mid p \in V(Y)\}$;
3. *Edge insertion:* $V(X) = V(Y) = \{u, v\}$, $E(X) = \emptyset$, and $E(Y) = \{(u, v)\}$; and
4. *Edge deletion:* $V(X) = V(Y) = \{u, v\}$, $E(X) = \{(u, v)\}$, and $E(Y) = \emptyset$.

Since we view a graph edit as "replacing" an induced subgraph of the given graph, we require that X be an induced subgraph of G. This also prevents inserting an edge already present in G. Although this requirement forces the edges between the neighbours of a deleted vertex to be included in X, these edges are retained in the resulting graph because we delete only the edges incident with the deleted vertex.

Next, to avoid duplicate vertex-labels, the vertex inserted in G must not already be present in G.

Finally, for an edit pair (X, Y) to fully specify the modifications in G, X must include all vertices of G whose degrees, and hence neighbourhoods, are modified. In the case of vertex deletion, this means all neighbours of the deleted vertex in G must be its neighbours in X.

Definition 9. We define *graph edit* by an operation *gedit*. Given a graph G and an edit pair (X, Y), we say $\mathit{gedit}(G, (X, Y)) = G'$ if and only if (X, Y) satisfies the following preconditions with respect to G:

1. X is an induced subgraph of G,
2. $(V(Y) \setminus V(X)) \cap V(G) = \emptyset$, and
3. $d_X(z) = d_G(z)$ for $z \in V(X) \setminus V(Y)$, if any,

and the graph G' is given by

$$\begin{aligned} V(G') &= (V(G) \setminus V(X)) \cup V(Y) \\ E(G') &= (E(G) \setminus E(X)) \cup E(Y) \end{aligned}$$

We say (X, Y) *fits* G if and only if (X, Y) satisfies the preconditions for the *gedit* operation with respect to G.

Note that the operation *gedit* is well-defined.

Fact 10. *If $gedit(G, (X, Y)) = G'$ for a graph G and an edit pair (X, Y), then Y is an induced subgraph of G'.*

Proof. Suppose $gedit(G, (X, Y)) = G'$. By Definition 9, $V(Y) \subseteq V(G')$ and $E(Y) \subseteq E(G')$, and so Y is a subgraph of G'. We claim Y is also an induced subgraph of G'.

If (X, Y) is a vertex insertion, then, by Definition 8, $V(X) = E(X) = E(Y) = \emptyset$ and $V(Y) = \{y\}$. Now, $y \in V(G')$ by Definition 9, which means the only vertex of Y is also in G' and so Y is an induced subgraph of G'.

For other graph edit operations, we claim $(y, z) \in E(Y)$ if and only if $(y, z) \in E(G')$ for any pair of vertices $y, z \in V(Y)$. If $(y, z) \in E(Y)$, then we know $(y, z) \in E(G')$ because Y is a subgraph of G'. To prove the other direction, we show if $(y, z) \notin E(Y)$, then $(y, z) \notin E(G')$.

Suppose $(y, z) \notin E(Y)$. Now, if (X, Y) is *not* a vertex insertion, then by Definition 8, $V(Y) \subseteq V(X)$. Therefore, it holds that $y, z \in V(X)$, and so we have two cases: either $(y, z) \in E(X)$ or $(y, z) \notin E(X)$. We show $(y, z) \notin E(G')$ in both the cases.

If $(y, z) \in E(X)$, then, $(y, z) \notin E(G) \setminus E(X)$, and hence, $(y, z) \notin E(G')$ by Definition 9. On the other hand, suppose $(y, z) \notin E(X)$. Since (X, Y) fits G, we know X is an induced subgraph of G by Definition 9, and so $(y, z) \notin E(G)$. Therefore, $(y, z) \notin E(G')$ by Definition 9.

Hence, for every $y, z \in V(Y)$, $(y, z) \in E(Y)$ if and only if $(y, z) \in E(G')$, which means Y is an induced subgraph of G'. □

3.6 NDL graph edit

We extend the notion of graph edit to specify the changes to the NDSs of the vertices, and hence the changes to the NDL. These changes are given by the nonincreasing lists we assign to vertices, which state their NDSs before and after the edit.

3.6.1 List-attributed edit pair

Definition 11. A *list attribution* Λ of a graph G is an assignment of a nonincreasing list $\Lambda(v)$ to each vertex $v \in V(G)$. A *list-attributed graph (LA graph)* $(G;\Lambda)$ is a graph G with a list attribution Λ.

We call a pair of LA graphs which specifies the modifications to a graph and to its NDL an LA edit pair (formally defined in Definition 12).

Given an LA edit pair $\mathcal{E} = ((X;\Lambda_X),(Y;\Lambda_Y))$, we call a vertex or edge *retained* if it is in both X and Y. We denote the sets of retained vertices and edges in $\mathcal{E}$ as $V_R(\mathcal{E})$ and $E_R(\mathcal{E})$, respectively.

Note that if the degree of a vertex in a graph changes, then the NDSs of its neighbours change too. Hence, an LA edit pair $\mathcal{E} = ((X;\Lambda_X),(Y;\Lambda_Y))$ contains *degree-modified vertices*, whose degrees in X and Y differ, as well as the retained neighbours of such vertices, called *list-modified vertices*, whose nonincreasing lists in $(X;\Lambda_X)$ and $(Y;\Lambda_Y)$ alone may differ. We denote the sets of degree-modified and list-modified vertices of $\mathcal{E}$ as $M_{deg}(\mathcal{E})$ and $M_{list}(\mathcal{E})$, respectively.

In Definition 12, when $\mathcal{E}$ is an edge insertion/deletion, the only degree-modified vertices are the end-vertices of the edge inserted/deleted, and they are explicitly stated.

Note that neither the degree nor the NDS of any vertex is changed by a vertex insertion in a graph.

Definition 12. A *list-attributed edit pair (LA edit pair)* $\mathcal{E} = ((X;\Lambda_X),(Y;\Lambda_Y))$ is an ordered pair of LA graphs $(X;\Lambda_X)$ and $(Y;\Lambda_Y)$ such that one of the following is true:

1. *Vertex insertion:* $V(X) = \emptyset$ and $V(Y) = \{y\}$;
2. *Vertex deletion:* $V(X) = \{z\} \cup M_{deg}(\mathcal{E}) \cup M_{list}(\mathcal{E})$ for some mutually disjoint vertex sets $\{z\}$, $M_{deg}(\mathcal{E})$, and $M_{list}(\mathcal{E})$, where $M_{deg}(\mathcal{E}) = N_X(z)$, $M_{list}(\mathcal{E}) = N_X^2(z)$, and Y is given by
$$\begin{aligned} V(Y) &= V(X) \setminus \{z\} \\ E(Y) &= E(X) \setminus \{(z,p) \mid p \in M_{deg}(\mathcal{E})\} \end{aligned}$$
3. *Edge insertion:* $V(X) = V(Y) = \{u,v\} \cup M_{list}(\mathcal{E})$ for some disjoint vertex sets $\{u,v\}$ and $M_{list}(\mathcal{E})$, where
   (a) $(u,v) \notin E(X)$,
   (b) $M_{list}(\mathcal{E}) = (N_X(u) \cup N_X(v)) \setminus \{u,v\}$, and
   (c) $E(Y) = E(X) \cup \{(u,v)\}$; and

4. *Edge deletion:* $V(X) = V(Y) = \{u, v\} \cup M_{list}(\mathcal{E})$ for some disjoint vertex sets $\{u, v\}$ and $M_{list}(\mathcal{E})$, where

   (a) $(u, v) \in E(X)$,
   (b) $M_{list}(\mathcal{E}) = (N_X(u) \cup N_X(v)) \backslash \{u, v\}$, and
   (c) $E(Y) = E(X) \setminus \{(u, v)\}$.

Fact 13 follows from Definition 12.

Fact 13. *$V_R(\mathcal{E}) = M_{deg}(\mathcal{E}) \cup M_{list}(\mathcal{E})$, or, every retained vertex is either degree-modified or list-modified.*

Definition 14. Given an LA edit pair $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$, we define the following sets:

1. $V_+(\mathcal{E}) = V(Y) \setminus V(X)$ contains the inserted vertex, if any,
2. $V_-(\mathcal{E}) = V(X) \setminus V(Y)$ contains the deleted vertex, if any,
3. $E_+(\mathcal{E}) = E(Y) \setminus E(X)$ contains the inserted edge, if any, and
4. $E_-(\mathcal{E}) = E(X) \setminus E(Y)$ contains the deleted edges, if any.

The fact that an LA edit pair inserts at most one edge at a time places a bound on the maximum degree of Y.

3.6.2 NDL graph edit

While an edit pair that fits a graph specifies the exact modifications to the graph, an LA edit pair that fits a graph gives only the "form" of the modifications. More precisely, unlike an edit pair, an LA edit pair $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$ specifies not a definite induced subgraph to be "replaced" in a graph G, but rather a family of induced subgraphs, each of which is "replaceable". These induced subgraphs are isomorphic to X, and hence are pairwise-isomorphic. The NDL graph edit operation implements the modifications to a graph specified by an LA edit pair with respect to a specific "replaceable" induced subgraph.

Since the *gedit* operation is well-defined, the graph obtained by a graph edit is determined by the edit pair. An LA edit pair, on the other hand, determines a family of graphs obtained by applying it to a given graph G, since G contains a family of "replaceable" induced subgraphs. However, as we shall prove in Lemma 57, each such

graph obtained G' has the same NDL, which means the LA edit pair determines the NDL of G'.

Also, we use the list-attributions of X and Y to specify the changes to the NDL of G, i.e., the changes to the NDSs of the vertices of a "replaceable" induced subgraph H. Therefore, X must be isomorphic to H such that the nonincreasing lists of $(X;\Lambda_X)$ match the NDSs of the corresponding vertices of H, or, $(X;\Lambda_X)$ must be list-isomorphic to H. Then, the nonincreasing lists of $(Y;\Lambda_Y)$ may be used to give the modified NDSs of the vertices of H.

Definition 15. Given a graph G, an LA graph $(A;\Lambda)$ is *list-isomorphic* to a subgraph H of G, which we denote by $(A;\Lambda) \simeq H$, if and only if A is isomorphic to H with respect to a bijection $f : V(A) \to V(H)$ such that $\Lambda(a) = NDS(G, f(a))$ for each vertex $a \in V(A)$.

For any vertex deleted from a graph, the degrees of its neighbours are changed. Therefore, to fully specify the modifications to G, a deleted vertex of X must have the same number of degree-modified neighbours as the deleted vertex of G. Similarly, to fully specify the changes to the NDL of G, the number of list-modified neighbours of a degree-modified vertex in X must equal the number of neighbours of its corresponding vertex in G.

We say $\mathcal{E}$ fits G if and only if $\mathcal{E}$ satisfies these conditions in addition to $(X;\Lambda_X)$ being list-isomorphic to an induced subgraph of G.

Definition 16. An LA edit pair $\mathcal{E} = ((X;\Lambda_X),(Y;\Lambda_Y))$ *fits* a graph G if and only if $(X;\Lambda_X) \simeq H$ for some induced subgraph H of G with respect to a bijection $f : V(X) \to V(H)$ such that $d_X(x) = d_G(f(x))$ for each vertex $x \in V_-(\mathcal{E}) \cup M_{deg}(\mathcal{E})$. We call $(G, \mathcal{E}, H, f)$ an *NDL-tuple*.

Given an NDL-tuple $(G, \mathcal{E}, H, f)$, the modifications to G are completely specified, or, in other words, the sets of vertices and edges that are inserted in and deleted from G are determined. We define these sets in Definition 17 in a way that "mirrors" in G the modifications specified by $\mathcal{E}$. The bijection f defines a mapping between the deleted vertices of X and G as well as between the edges inserted in or deleted from X and G.

If $\mathcal{E}$ is a vertex insertion, since G may contain a vertex with the same label as $y \in V_+(\mathcal{E})$, we insert a vertex $s \notin V(G)$.

Definition 17. Let $\Phi = (G, \mathcal{E}, H, f)$ be an NDL-tuple. We define the following sets with respect to Φ:

1. $V_+(\Phi) = \begin{cases} \{s\} \text{ for some } s \notin V(G) & \text{if } V_+(\mathcal{E}) \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases}$ contains the inserted vertex, if any,

2. $V_-(\Phi) = \{f(z) \mid z \in V_-(\mathcal{E})\}$ contains the deleted vertex, if any,

3. $E_+(\Phi) = \{(f(u), f(v)) \mid (u, v) \in E_+(\mathcal{E})\}$ contains the inserted edge, if any, and

4. $E_-(\Phi) = \{(f(w), f(z)) \mid (w, z) \in E_-(\mathcal{E})\}$ contains the deleted edges, if any.

Note that the above sets are well-defined. We use these sets to define *NDL graph edit* as an operation *ndledit*, which applies an LA edit pair to a given graph with respect to an induced subgraph and a bijection.

Definition 18. We define *NDL graph edit* as an operation *ndledit*. Given an NDL-tuple $\Phi = (G, \mathcal{E}, H, f)$, we say $\mathit{ndledit}(\Phi) = G'$ if and only if the graph G' is given by

$$V(G') = (V(G) \setminus V_-(\Phi)) \cup V_+(\Phi) \tag{3.6.1}$$
$$E(G') = (E(G) \setminus E_-(\Phi)) \cup E_+(\Phi) \tag{3.6.2}$$

Note that the *ndledit* operation is well-defined.

Fact 19. *Suppose an LA edit pair $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$ fits a graph G. Then, $|V(X)| \leq \Delta(G)^2 + 1$ and $|V(Y)| \leq \Delta(G)^2 + 1$.*

Proof. Since $\mathcal{E}$ fits G, $(X; \Lambda_X)$ is list-isomorphic to an induced subgraph of G (Definition 16), and so X is isomorphic to a subgraph of G. Hence, $\Delta(X) \leq \Delta(G)$.

By Definition 12, the following hold:

1. $|V(X)| = 0$ and $|V(Y)| = 1$ when $\mathcal{E}$ is a vertex insertion,

2. $|V(X)| \leq 1 + \Delta(X) + \Delta(X)(\Delta(X) - 1) \leq \Delta(G)^2 + 1$ and $|V(Y)| \leq \Delta(G)^2$ when $\mathcal{E}$ is a vertex deletion,

3. $|V(X)| = |V(Y)| \leq 2\Delta(X) + 2$ when $\mathcal{E}$ is an edge insertion because u and v can each have $\Delta(G)$ neighbours, and

4. $|V(X)| = |V(Y)| \leq 2\Delta(X)$ when $\mathcal{E}$ is an edge deletion because (u, v) is an edge, which means u and v can each have at most $\Delta(G) - 1$ neighbours not in $\{u, v\}$.

Therefore, $|V(X)| \leq \Delta(G)^2 + 1$ and $|V(Y)| \leq \Delta(G)^2 + 1$. □

We now show that the induced subgraph H contains the vertices and edges deleted in G, and so the vertices and edges of G not in H are "untouched".

Fact 20. *Let $\Phi = (G, \mathcal{E}, H, f)$ be an NDL-tuple. Then, $V_-(\Phi) \subseteq V(H)$ and $E_-(\Phi) \subseteq E(H)$.*

Proof. First, we claim that the vertex deleted from G, if any, is a vertex of H. Since $f : V(X) \to V(H)$ is a bijection and $V_-(\mathcal{E}) \subseteq V(X)$ by Definition 14, $V_-(\Phi) \subseteq V(H)$.

Next, we claim that the edge deleted from G, if any, is an edge of H. We know that, for each pair of vertices $w, z \in V(X)$, the vertices $f(w), f(z) \in V(H)$ because $f : V(X) \to V(H)$ is a bijection. Since $(X; \Lambda_X) \simeq H$ with respect to f, it follows from Definition 15 that X is isomorphic to H with respect to f. Therefore, $(w, z) \in E(X)$ if and only if $(f(w), f(z)) \in E(H)$. Now, for each edge $(f(w), f(z)) \in E_-(\Phi)$, $(w, z) \in E_-(\mathcal{E})$ by Definition 17. We know that $E_-(\mathcal{E}) \subseteq E(X)$ by Definition 14, and so $(w, z) \in E(X)$, which implies $(f(w), f(z)) \in E(H)$. Hence, $E_-(\Phi) \subseteq E(H)$. □

A graph G and an LA edit pair $\mathcal{E}$ define a set of NDL graph edits, and a family of graphs obtained by performing these NDL graph edits, which we call $(G, \mathcal{E})$-produced graphs. In Fact 22, we bound the maximum degree of a $(G, \mathcal{E})$-produced graph in terms of the maximum degree of G.

Definition 21. Given a graph G and an NDL graph edit $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$, a graph G' is *$(G, \mathcal{E})$-produced* if and only if there exists an induced subgraph H of G and a bijection $f : V(X) \to V(H)$ such that $(G, \mathcal{E}, H, f)$ is an NDL-tuple and $ndledit((G, \mathcal{E}, H, f)) = G'$.

Fact 22. *If G' is a $(G, \mathcal{E})$-produced graph, then $\Delta(G') \leq \Delta(G) + 1$.*

Proof. Suppose $G' = ndledit((G, \mathcal{E}, H, f))$ for some H and f. Clearly, $\Delta(G') > \Delta(G)$ only if $\mathcal{E}$ is an edge insertion. We claim $\Delta(G') \leq \Delta(G) + 1$.

Inserting an edge (u, v) in G raises the degrees of its end-vertices by one each and so $\Delta(G') \leq \Delta(G) + 1$ with equality holding if $d_G(u) = \Delta(G)$ or $d_G(v) = \Delta(G)$. □

The vertices of G' can be divided into three kinds: the inserted vertex, "untouched" vertices, and those whose corresponding vertices in $\mathcal{E}$ are retained vertices.

Lemma 23. *Suppose $G' = ndledit((G, \mathcal{E}, H, f))$. Then, for each vertex $v \in V(G')$, exactly one of the following holds:*

1. *$v \in V(G') \setminus V(G)$,*
2. *$v \in V(G) \setminus V(H)$, or*
3. *there exists a retained vertex $w \in V_R(\mathcal{E})$ such that $v = f(w)$.*

Proof. We show that $V(G')$ is a union of three sets – $V(G') \setminus V(G)$, $V(G) \setminus V(H)$, and $\{f(w) \mid w \in V_R(\mathcal{E})\}$ – and that these sets are mutually disjoint.

Because $V(G)$ can also be written as $(V(G) \setminus V(H)) \cup V(H)$, we have

$$V(G) \setminus V_{-}(\Phi) \;=\; \Big((V(G) \setminus V(H)) \cup V(H)\Big) \setminus V_{-}(\Phi)$$

Now, $V_{-}(\Phi) \subseteq V(H)$ by Fact 20, and so

$$V(G) \setminus V_{-}(\Phi) \;=\; \Big(V(G) \setminus V(H)\Big) \cup \Big(V(H) \setminus V_{-}(\Phi)\Big)$$

Since $\mathcal{E}$ fits G, we know X is isomorphic to H with respect to f (Definition 16), which means $V(H) = \{f(x) \mid x \in V(X)\}$. Furthermore, $V_{-}(\Phi) = \{f(z) \mid z \in V_{-}(\mathcal{E})\} = \{f(z) \mid z \in V(X) \setminus V(Y)\}$ by Definitions 14 and 17. Therefore,

$$\begin{aligned} V(H) \setminus V_{-}(\Phi) &= \{f(w) \mid w \in V(X) \cap V(Y)\} \\ &= \{f(w) \mid w \in V_R(\mathcal{E})\} \end{aligned}$$

We also know by Definition 18 that $V(G') = (V(G) \setminus V_{-}(\Phi)) \cup V_{+}(\Phi)$, where $V_{+}(\Phi) = V(G') \setminus V(G)$. Hence, it follows from the above equations that

$$\begin{aligned} V(G') &= (V(G) \setminus V_{-}(\Phi)) \cup V_{+}(\Phi) \\ &= \Big(V(G) \setminus V(H)\Big) \cup \{f(w) \mid w \in V_R(\mathcal{E})\} \cup \Big(V(G') \setminus V(G)\Big) \end{aligned} \tag{3.6.3}$$

We show that these subsets of $V(G')$ are mutually disjoint. Clearly, $V(G) \setminus V(H)$ and $V(G') \setminus V(G)$ are disjoint. Since $V(H) = \{f(x) \mid x \in V(X)\}$ and $V_R(\mathcal{E}) \subseteq V(X)$, it follows that $\{f(w) \mid w \in V_R(\mathcal{E})\} \subseteq V(H)$. Hence, $\{f(w) \mid w \in V_R(\mathcal{E})\}$ has no vertices in common with either $V(G) \setminus V(H)$ or $V(G') \setminus V(G)$ because H is a subgraph of G. □

Since an NDL graph edit "replaces" an induced subgraph isomorphic to X in G, we might expect that Y is isomorphic to an induced subgraph of G'.

Lemma 24. *If $G' = ndledit(\Phi)$, where $\Phi = (G, \mathcal{E}, H, f)$ and $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$, then Y is isomorphic to an induced subgraph of G' with respect to a bijection f' given by*

$$f'(y) = \begin{cases} f(y) & \text{if } y \in V_R(\mathcal{E}) \\ s \in V(G') \setminus V(G) & \text{if } y \in V_{+}(\mathcal{E}) \end{cases}$$

Proof. Suppose $ndledit(\Phi) = G'$, where $\Phi = (G, \mathcal{E}, H, f)$ and $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$. By Definition 16, $(X, \Lambda_X) \simeq H$ with respect to f and hence X is isomorphic to H with respect to f. We will construct H' by modifying H as specified by $\mathcal{E}$ and the bijection

f. We will then show that H' is an induced subgraph of G' and that Y is isomorphic to H'.

Let H' be given by

$$V(H') = (V(H) \setminus V_-(\Phi)) \cup (V(G') \setminus V(G)) \tag{3.6.4}$$

$$E(H') = (E(H) \setminus E_-(\Phi)) \cup E_+(\Phi) \tag{3.6.5}$$

Note that the graph H' is legally defined since, by Fact 20, $V_-(\Phi) \subseteq V(H)$ and $E_-(\Phi) \subseteq E(H)$.

Observe that Equations 3.6.4 and 3.6.5 are quite similar to Equations 3.6.1 and 3.6.2 in Definition 18, respectively. The only difference is that, if $\mathcal{E}$ is a vertex insertion, then H' contains the actual vertex inserted in G rather than the arbitrary vertex in $V_+(\Phi)$ to ensure the vertex-labels of the inserted vertices in H' and G' match.

First, we show H' is an induced subgraph of G'. We know H is an induced subgraph of G by Definition 16. Comparing Equations 3.6.4 and 3.6.5 with Equations 3.6.1 and 3.6.2, respectively, we observe that a vertex v is inserted in (deleted from) H if and only if v is inserted in (respectively, deleted from) G. Similarly, an edge (u, v) is inserted in (deleted from) H if and only if (u, v) is inserted in (respectively, deleted from) G. Hence, every vertex of H' is also in G', and for each pair of vertices $u, v \in V(H')$, $(u, v) \in E(H')$ if and only if $(u, v) \in E(G')$, which implies H' is an induced subgraph of G'.

Next, we claim $|V(H')| = |V(Y)|$. We know $|V(H)| = |V(X)|$ because H and X are isomorphic. Now, by Definition 17, for $z \in V_-(\mathcal{E})$, if any, there exists a vertex $f(z) \in V_-(\Phi) = V(H) \setminus V(H')$. Furthermore, by Definition 17 and Equation 3.6.1, for $y \in V_+(\mathcal{E})$, if any, there exists a vertex $s \in V(G') \setminus V(G) = V(H') \setminus V(H)$. Therefore, $|V(H')| = |V(Y)|$.

Further, we claim that f' is well-defined, i.e., $f'(y) \in V(H')$ for each $y \in V(Y)$. If $y \in V_R(\mathcal{E})$, then y is neither in $V_-(\Phi)$ nor in $V(G') \setminus V(G)$ by Definition 17 and Equation 3.6.1. Hence, by Equation 3.6.4, $f'(y) = f(y) \in V(H) \cap V(H')$. On the other hand, if $y \in V_+(\mathcal{E})$, then $V(H') \setminus V(H) = V(G') \setminus V(G) = \{s\}$ for some s as given by Definitions 17 and 18, and so $f'(y) = s \in V(H')$.

Now, we show that Y is isomorphic to H' with respect to f', or, $(f'(u), f'(v)) \in E(H')$ if and only if $(u, v) \in E(Y)$ for each pair of vertices $u, v \in V(Y)$. The following cases cover all pairs of vertices of Y.

Case 1. (u, v) is a retained edge in $\mathcal{E}$.

Then, the vertices $u, v \in V_R(\mathcal{E})$, and so $f'(u) = f(u)$ and $f'(v) = f(v)$. Since X is isomorphic to H with respect to f, $(u, v) \in E(X)$ implies $(f(u), f(v)) \in$

$E(H)$. Now, since $(u, v) \in E_R(\mathcal{E})$, it follows that $((f(u), f(v)) \notin E_-(\Phi)$ by Definition 17, or, $(f(u), f(v))$ is present in G'. Hence, by Equation 3.6.5, $(f(u), f(v))$ is also present in H', and so $(f'(u), f'(v)) = (f(u), f(v)) \in E(H')$.

Case 2. (u, v) is the inserted edge in $\mathcal{E}$.

Then, the end-vertices $u, v \in V_R(\mathcal{E})$, and so $f'(u) = f(u)$ and $f'(v) = f(v)$. Furthermore, by Definition 17, $(f(u), f(v)) \in E_+(\Phi)$, i.e., the edge $(f(u), f(v))$ is inserted in G. Therefore, by Equation 3.6.5, $(f(u), f(v))$ is also inserted in H and so $(f'(u), f'(v)) = (f(u), f(v)) \in E(H')$.

Case 3. (u, v) is a deleted edge in $\mathcal{E}$.

Then, the end-vertices $u, v \in V_R(\mathcal{E})$, and so $f'(u) = f(u)$ and $f'(v) = f(v)$. Moreover, by Definition 17, $(f(u), f(v)) \in E_-(\Phi)$, i.e., the edge $(f(u), f(v))$ is deleted from G. Hence, by Equation 3.6.5, $(f(u), f(v))$ is also deleted from H, which implies $(f'(u), f'(v)) = (f(u), f(v)) \notin E(H')$.

Case 4. (u, v) is not an edge in $\mathcal{E}$, i.e., $(u, v) \notin E(X) \cup E(Y)$.

Case i. $u, v \in V(X)$.

Since we assume $u, v \in V(Y)$, it follows that $u, v \in V_R(\mathcal{E})$, and so $f'(u) = f(u)$ and $f'(v) = f(v)$. Also, $(f(u), f(v)) \notin E(H)$ because X is isomorphic to H with respect to f. Moreover, by Definition 17, $(f(u), f(v)) \notin E_+(\Phi)$, i.e., $(f(u), f(v))$ is absent from G'. Therefore, by Equation 3.6.5, $(f(u), f(v))$ is also absent from H', which implies $(f'(u), f'(v)) = (f(u), f(v)) \notin E(H')$.

Case ii. $u \notin V(X)$.

Then, $\mathcal{E}$ is a vertex insertion and so $f'(u) = s \in V(G') \setminus V(G)$ and $f'(v) = f(v)$. By Definition 17, then, $(f'(u), f'(v)) \notin E_+(\Phi)$, i.e., $(f'(u), f'(v))$ is absent from G'. Therefore, $(f'(u), f'(v))$ is absent from H', and so $(f'(u), f'(v)) \notin E(H')$ by Equation 3.6.5.

Case iii. $v \notin V(X)$.

The argument proceeds similar to that in Case (ii), with u replacing v and vice-versa.

Thus, Y is isomorphic to H with respect to the bijection f'. □

The LA edit pair $\mathcal{E}$ specifies the degree in G' of each vertex whose degrees in G and G' differ.

Fact 25. *Suppose* $ndledit(\Phi) = G'$*, where* $\Phi = (G, \mathcal{E}, H, f)$ *and* $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$*. Then,* $d_Y(p) = d_{G'}(f(p))$ *for each* $p \in M_{deg}(\mathcal{E})$*.*

Proof. For each degree-modified vertex $p \in M_{deg}(\mathcal{E})$, it holds that $d_X(p) = d_G(p)$ by Definition 16. We show that, for each edge incident with p that is inserted in (deleted from) X, a corresponding edge incident with $f(p)$ is inserted in (respectively, deleted from) G, and so the degrees of p in Y and $f(p)$ in G' are equal.

Suppose an edge (p, q) is inserted in X. Then, $d_Y(p) = d_X(p) + 1$ (Definition 12). Since $(p, q) \in E_+(\mathcal{E})$, by Definition 17, $(f(p), f(q)) \in E_+(\Phi)$, i.e., the edge $(f(p), f(q))$ is inserted in G. Hence, by Equation 3.6.2, $(f(p), f(q)) \in E(G')\backslash E(G)$, and so $d_{G'}(f(p)) = d_G(f(p)) + 1$.

Suppose an edge (p, r) is deleted from X either by an edge deletion or a vertex deletion. Then, $d_Y(p) = d_X(p) - 1$. Since $(p, r) \in E_-(\mathcal{E})$, by Definition 17, $(f(p), f(r)) \in E_-(\Phi)$, i.e., the edge $(f(p), f(r))$ is deleted in G. Hence, by Equation 3.6.2, $(f(p), f(r)) \in E(G) \setminus E(G')$, and so $d_{G'}(f(p)) = d_G(f(p)) - 1$.

Thus, $d_Y(p) = d_{G'}(f(p))$. □

3.6.3 Complete LA edit pairs

We introduce the notion of completeness of $\mathcal{E}$ in G to specify that the NDSs of the "NDS-modified" vertices in G' are given by Λ_Y, where $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$ and G' is a $(G, \mathcal{E})$-produced graph. This suggests $(Y; \Lambda_Y)$ is list-isomorphic to an induced subgraph of G' (as we will prove in Fact 29). It also implies the list-modified vertices in $\mathcal{E}$ correspond to the "NDS-modified" vertices of G (as we will prove in Fact 27). Since X is isomorphic to a subgraph H of G (Definition 16), the NDSs of the "untouched" vertices of G, i.e. the vertices not in H, are unchanged (as we will prove in Fact 28).

Note that the NDS of an inserted vertex is an empty list since we insert only isolated vertices.

Definition 26. An LA edit pair $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$ that fits a graph G is *complete* in G if and only if, for any graph $G' = ndledit((G, \mathcal{E}, H, f))$, the following hold:

1. $\Lambda_Y(y)$ is an empty list for $y \in V_+(\mathcal{E})$ and
2. $NDS(G', f(w)) = \Lambda_Y(w)$ for each $w \in V_R(\mathcal{E})$.

Fact 27. *Suppose* $ndledit((G, \mathcal{E}, H, f)) = G'$ *and* $\mathcal{E}$ *is complete in* G*. Then, for any vertex* $p \in V_R(\mathcal{E})$*,* $p \in M_{deg}(\mathcal{E})$ *if and only if* $d_G(f(p)) \neq d_{G'}(f(p))$*. Also, for any vertex* $q \in V_R(\mathcal{E})$*,* $q \in M_{list}(\mathcal{E})$ *if and only if* $NDS(G, f(q)) \neq NDS(G', f(q))$*.*

Proof. Suppose $ndledit(\Phi) = G'$, where $\Phi = (G, \mathcal{E}, H, f)$ and $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$ is complete in G. First, we show that $p \in M_{deg}(\mathcal{E})$ if and only if $d_G(f(p)) \neq d_{G'}(f(p))$ for any $p \in V_R(\mathcal{E})$.

Consider a vertex $p \in M_{deg}(\mathcal{E})$, which means $d_X(p) \neq d_Y(p)$. Since $\mathcal{E}$ fits G, we know that $d_X(p) = d_G(f(p))$ by Definition 16. We also know that $d_Y(p) = d_{G'}(f(p))$ by Fact 25. Hence, $d_G(f(p)) \neq d_{G'}(f(p))$.

Suppose $p \notin M_{deg}(\mathcal{E})$, which means $d_X(p) = d_Y(p)$. In other words, no edge incident with p is either inserted in or deleted from X, i.e., neither $E_+(\mathcal{E})$ nor $E_-(\mathcal{E})$ contains an edge incident with p. Therefore, neither $E_+(\Phi)$ nor $E_-(\Phi)$ contains an edge incident with $f(p)$ by Definition 17. Hence, no edge incident with $f(p)$ is either inserted in or deleted from G, and so $d_G(f(p)) = d_{G'}(f(p))$.

Next, we show that $q \in M_{list}(\mathcal{E})$ if and only if $NDS(G, f(q)) \neq NDS(G', f(q))$ for any $q \in V_R(\mathcal{E})$.

Suppose $q \in M_{list}(\mathcal{E})$, which means $\Lambda_X(q) \neq \Lambda_Y(q)$. Since $\mathcal{E}$ fits G, $(X; \Lambda_X) \simeq H$ with respect to f by Definition 16, and since $q \in V(X)$, $\Lambda_X(q) = NDS(G, f(q))$. Now, $\mathcal{E}$ is also complete in G, and since $q \in V_R(\mathcal{E})$, $\Lambda_Y(q) = NDS(G', f(q))$ by Definition 26. Therefore, $NDS(G, f(q)) \neq NDS(G', f(q))$.

Suppose $q \notin M_{list}(\mathcal{E})$, which means $\Lambda_X(q) = \Lambda_Y(q)$. Arguing similarly as in the case $q \in M_{list}(\mathcal{E})$, we obtain $\Lambda_X(q) = NDS(G, f(q))$ and $\Lambda_Y(q) = NDS(G', f(q))$, which implies $NDS(G, f(q)) = NDS(G', f(q))$. □

Fact 28. *Suppose* $ndledit(\Phi) = G'$, *where* $\Phi = (G, \mathcal{E}, H, f)$. *Then* $NDS(G', u) = NDS(G, u)$ *for each* $u \in V(G) \setminus V(H)$.

Proof. We show that, for each vertex $u \in V(G) \setminus V(H)$, neither the degree nor the NDS of u is changed by the NDL graph edit.

We know $(X; \Lambda_X) \simeq H$ with respect to f by Definition 16, which implies X is isomorphic to H with respect to f.

First, we claim that H contains every vertex of G whose degree is changed. Suppose for contradiction that $d_G(u) \neq d_{G'}(u)$ for some $u \in V(G) \setminus V(H)$. Then, either an edge (u, v) is inserted/deleted by the NDL graph edit or a neighbour w of u is deleted. In the former case, $(u, v) \in E_+(\Phi)$ or $(u, v) \in E_-(\Phi)$, which means there exists a vertex $x \in V(X)$ such that $f(x) = u$ by Definition 17. This implies $u \in V(H)$, which is a contradiction. On the other hand, if a neighbour w of u is deleted, then $w \in V_-(\Phi)$, and so there exists a vertex $z \in V_-(\mathcal{E})$ such that $f(z) = w$. Now, since $\mathcal{E}$ fits G, $d_X(z) = d_G(w)$ by Definition 16, and so there exists a vertex $x \in N_X(z)$ such that $f(x) = u$ because X is isomorphic to H with respect to f. This implies $u \in V(H)$, which is again a contradiction.

Next, we claim that H contains every vertex of G whose NDS alone is modified. Suppose for contradiction that $d_G(u) = d_{G'}(u)$ but $NDS(G, u) \neq NDS(G', u)$ for some $u \in V(G) \setminus V(H)$. Then, the degree of a neighbour v of u is changed by the NDL graph edit. It follows from our previous claim that H contains v, which implies there exists a vertex $p \in V(X)$ such that $f(p) = v$. Moreover, by Fact 27, $p \in M_{deg}(\mathcal{E})$ because $d_G(v) \neq d_{G'}(v)$. Now, since $\mathcal{E}$ fits G, $d_X(p) = d_G(v)$ by Definition 16, and so there exists a vertex $q \in N_X(p)$ such that $f(q) = u$ because X is isomorphic to H with respect to f. This implies $u \in V(H)$, which is a contradiction.

Thus, for each vertex $u \in V(G) \setminus V(H)$, neither its degree nor its NDS is changed by the NDL graph edit, and so $NDS(G', u) = NDS(G, u)$. □

Fact 29. *Suppose* $ndledit(\Phi) = G'$, *where* $\Phi = (G, \mathcal{E}, H, f)$ *and* $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$ *such that* $\mathcal{E}$ *is complete in* G. *Then,* $(Y; \Lambda_Y)$ *is list-isomorphic to an induced subgraph of* G' *with respect to a bijection* f' *given by*

$$f'(y) = \begin{cases} f(y) & \text{if } y \in V_R(\mathcal{E}) \\ s \in V(G') \setminus V(G) & \text{if } y \in V_+(\mathcal{E}) \end{cases}$$

Proof. Suppose $ndledit(\Phi) = G'$. We know that Y is isomorphic to an induced subgraph H' of G' with respect to f' by Fact 24. We claim that $(Y; \Lambda_Y) \simeq H'$, i.e., $NDS(G', f'(y)) = \Lambda_Y(y)$ for each vertex $y \in V(Y)$.

Observe that each vertex of Y is either a retained vertex or an inserted vertex, i.e., $V(Y) = V_R(\mathcal{E}) \cup V_+(\mathcal{E})$.

By Definition 26, $NDS(G', f(w)) = \Lambda_Y(w)$ for each $w \in V_R(\mathcal{E})$. If there exists a vertex $r \in V_+(\mathcal{E})$, then $\Lambda_Y(r)$ is an empty list by Definition 26, and so is $NDS(G', f'(r))$ because $f'(r)$ is an isolated vertex of G'.

Therefore, $NDS(G', f'(y)) = \Lambda_Y(y)$ for each vertex $y \in V(Y)$, which implies $(Y; \Lambda_Y) \simeq H'$ with respect to f'. □

We introduce the notion of provisional completeness, which, unlike completeness, is a property of the LA edit pair alone.

Definition 30. An LA edit pair $\mathcal{E}$ is *provisionally complete* if, for any graph G, $\mathcal{E}$ fits G implies $\mathcal{E}$ is complete in G.

Later, we shall prove in Chapter 6 that $\mathcal{E}$ is complete in a graph G only if $\mathcal{E}$ is provisionally complete (Lemma 64). In other words, if $\mathcal{E}$ is complete in G, then $\mathcal{E}$ is also complete in every graph it fits.

3.7 Time complexity of checking list-isomorphism and degree conditions

Here, we show we can check whether an LA graph $(A;\Lambda)$ is list-isomorphic to an induced subgraph of a fixed graph G such that a degree condition holds on a subset of vertices of H in FPT time when parameterized by $\Delta(G)$ and $|V(A)|$. By Definition 16, this would imply we can check whether an LA edit pair $\mathcal{E} = ((X;\Lambda_X),(Y;\Lambda_Y))$ fits G in FPT time when parameterized by $\Delta(G)$ and $|V(X)|$. We use a result by Frick and Grohe [16], which states model checking with a formula of first-order logic on graphs with bounded local treewidth is in FPT.

A *tree decomposition* [12] of a graph G is a tree T which has "bags" associated with its nodes such that (i) for each edge $(u,v) \in E(G)$, at least one bag of T contains u and v, and (ii) for each $v \in V(G)$, the nodes of T that contain v form a non-empty subtree in T. The *size* of a tree decomposition is one less than the maximum number of vertices in a bag. The *treewidth* of G is the minimum size of a tree decomposition of G.

A class of graphs has *bounded local treewidth* [14] if and only if there exists a function $\rho : \mathbb{N} \to \mathbb{N}$ such that, for each graph in the class, the treewidth of the subgraph induced by the r-neighbourhood of any vertex is bounded by $\rho(r)$ for each $1 \leq r < n$.

Frick and Grohe [16, Theorem 1.1.] showed that a property defined by a formula φ of first-order logic can be checked in a graph G with bounded local treewidth in time $\mathcal{O}(g(|\varphi|) \cdot |V(G)|)$, where $|\varphi|$ is the size of the formula, for some function g. They also showed that graphs with bounded degree have bounded local treewidth [16, Example 5.3.]. Therefore, for an arbitrary G, there exists a function h such that φ can be checked in G in time $\mathcal{O}(h(|\varphi| + \Delta(G)) \cdot |V(G)|)$, which implies Fact 31.

Fact 31. *Given a graph G and a property specified by a formula φ of first-order logic, checking whether G satisfies the property is in FPT when parameterized by $|\varphi| + \Delta(G)$.*

Now, we define a first-order formula φ_L to determine whether $(A;\Lambda)$ is list-isomorphic to an induced subgraph H of a fixed G with respect to a bijection $f : V(A) \to V(H)$ such that for a given $D \subseteq V(A)$, $d_A(z) = d_G(f(z))$ for each $z \in D$.

Definition 32. Given a graph G, an LA graph $(A;\Lambda)$ where $V(A) = \{u_1, u_2, \ldots, u_a\}$, and a subset $D \subseteq V(A)$, we define $\varphi_L = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$, where

1. $\varphi_1 : \exists v_1, v_2, \ldots, v_a \in V(G) : \bigwedge_{1 \leq i,j \leq a} (v_i \neq v_j)$, or, there exist a distinct vertices of G,

2. $\varphi_2 : \bigwedge\limits_{1\leq i,j\leq a} \Big((u_i, u_j) \in E(A) \Leftrightarrow (v_i, v_j) \in E(G)\Big)$, or, an edge exists between a pair of vertices in A if and only if an edge exists between their corresponding vertices in G,

3. $\varphi_3 : \bigwedge\limits_{1\leq i\leq a} \Big(\Lambda(u_i) = NDS(G, v_i)\Big)$, or, the list attribution Λ gives the NDSs of the vertices of H in G, and

4. φ_4: $\bigwedge\limits_{1\leq i\leq a} \Big((u_i \in D) \Rightarrow (d_A(u_i) = d_G(v_i))\Big)$, or, for each vertex in D, its degree in A matches the degree of its corresponding vertex in G.

If we do not require that H be induced in G, we replace the biconditional in φ_2 with a conditional so that $E(A) \subseteq E(H)$.

Definition 33. We define $\varphi'_L = \varphi_1 \wedge \varphi'_2 \wedge \varphi_3 \wedge \varphi_4$, where $\varphi'_2 : \bigwedge_{1\leq i,j\leq a} \Big((u_i, u_j) \in E(A) \Rightarrow (v_i, v_j) \in E(G)\Big)$ and φ_1, φ_3, and φ_4 are as stated in Definition 32.

Fact 34. *Given a graph G and and LA graph $(A; \Lambda)$, $|\varphi_L|$ and $|\varphi'_L|$ are polynomial in $|V(A)| + \Delta(G)$.*

Proof. We show that the sizes of φ_1, φ_2, φ_3, and φ_4 are each polynomial in $|V(A)| + \Delta(G)$. First, we bound the size of $\Lambda(u)$ for each $u \in V(A)$. Note that $|NDS(G, v)| \leq \Delta(G)$ for each $v \in V(G)$ because v has at most $\Delta(G)$ neighbours. Therefore, if $\Lambda(u) \geq \Delta(G)$ for some $u \in V(A)$, then we know $\Lambda(u) \neq NDS(G, v)$ for any $v \in V(G)$, which means $(A; \Lambda)$ is not list-isomorphic to any induced subgraph of G. Hence, we assume $\Lambda(u) \leq \Delta(G)$ for each $u \in V(A)$.

φ_1 contains an inequality condition for each pair of vertices of A, and there are $|V(A)|^2$ such pairs. Hence, $|\varphi_1|$ is in $\mathcal{O}(|V(A)|^2)$.

φ_2 contains a biconditional for each pair of vertices of A, which means $|\varphi_2|$ is in $\mathcal{O}(|V(A)|^2)$.

φ_3 contains the equality condition for each vertex of A, so there are $|V(A)|$ equality conditions. For a given ith condition, $\Lambda(u_i)$ and $NDS(G, v_i)$ each has a size of at most $\Delta(G)$. Therefore, $|\varphi_3|$ is in $\mathcal{O}(|V(A)| \cdot \Delta(G))$.

φ_4 contains a conditional for each vertex in $D \subseteq V(A)$. Hence, there are at most $|V(A)|$ conditionals, which means $|\varphi_4|$ is in $\mathcal{O}(|V(A)|)$.

Thus, $|\varphi_L|$ is in $\mathcal{O}((|V(A)| + \Delta(G))^2)$. Since φ'_L is obtained from φ_L by replacing the biconditional symbol by a conditional symbol in φ_2, $|\varphi'_L|$ is also in $\mathcal{O}((|V(A)| + \Delta(G))^2)$. □

Lemma 35 follows from Facts 31 and 34.

Lemma 35. *Given a graph G, an LA graph $(A;\Lambda)$, and a set of vertices $D \subseteq V(A)$, determining whether $(A;\Lambda)$ is list-isomorphic to an (induced) subgraph H of G with respect to a bijection $f : V(A) \to V(H)$ such that $d_A(z) = d_G(f(z))$ for each $z \in D$ is in FPT when parameterized by $\Delta(G) + |V(A)|$.*

3.8 Sequences of edit pairs and LA edit pairs

Definition 36. Let $\mathcal{X} = \langle \mathcal{X}[i] \rangle_0^t$ and $\mathcal{Y} = \langle \mathcal{Y}[i] \rangle_0^t$ be sequences of graphs for some $t \geq 0$ such that $(\mathcal{X}[i], \mathcal{Y}[i])$ is an edit pair for each $0 \leq i \leq t$. Then, we say $\langle (\mathcal{X}[i], \mathcal{Y}[i]) \rangle_0^t$ is an *edit pair sequence*.

Definition 37. Let $\mathcal{X} = \langle \mathcal{X}[i] \rangle_0^t$ and $\mathcal{Y} = \langle \mathcal{Y}[i] \rangle_0^t$ be sequences of graphs for some $t \geq 0$. Moreover, let $\Lambda_{\mathcal{X}[i]}$ and $\Lambda_{\mathcal{Y}[i]}$ be list-attributions of $\mathcal{X}[i]$ and $\mathcal{Y}[i]$, respectively, such that $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ is an LA edit pair for each $0 \leq i \leq t$. Then, we say $\langle \mathcal{E}_i \rangle_0^t$ is an *LA edit pair sequence*.

Definition 38. A sequence of edit pairs $\langle (\mathcal{X}[i], \mathcal{Y}[i]) \rangle_0^t$ *fits* a graph G_0 if and only if there exists a sequence of graphs $\langle G_j \rangle_0^{t+1}$ such that $(\mathcal{X}[i], \mathcal{Y}[i])$ fits G_i and $\mathit{gedit}(G_i, (\mathcal{X}[i], \mathcal{Y}[i])) = G_{i+1}$ for $0 \leq i \leq t$. We say $\mathit{gedit}^t(G_0, \langle (\mathcal{X}[i], \mathcal{Y}[i]) \rangle_0^t) = G_{t+1}$.

Definition 39. A sequence of LA edit pairs $\langle \mathcal{E}_i \rangle_0^t$ *fits* a graph G_0 if and only if there exists a sequence of graphs $\langle G_j \rangle_0^{t+1}$ such that (i) $\mathcal{E}_i$ fits G_i for $0 \leq i \leq t$, and (ii) G_{i+1} is a $(G_i, \mathcal{E}_i)$-produced graph for $0 \leq i \leq t$. We say G_{t+1} is a $(G_0, \langle \mathcal{E}_i \rangle_0^t)$*-produced graph*.

Definition 40. An LA edit pair sequence $\langle \mathcal{E}_i \rangle_0^t$ is *complete* in G_0 if and only if $\langle \mathcal{E}_i \rangle_0^t$ fits G_0 with respect to a sequence of graphs $\langle G_j \rangle_0^{t+1}$ and $\mathcal{E}_i$ is complete in G_i for $0 \leq i \leq t$.

Definition 41. An LA edit pair sequence is *provisionally complete* if it is a sequence of provisionally complete LA edit pairs.

Fact 42. *Suppose a sequence of LA edit pairs $\langle \mathcal{E}_i \rangle_0^t$ fits a graph G_0 with respect to a sequence of graphs $\langle G_j \rangle_0^{t+1}$. Then, $\Delta(G_j) \leq \Delta(G_0) + t + 1$ for each $0 \leq j \leq t + 1$.*

Proof. Since $\langle \mathcal{E}_i \rangle_0^t$ fits G_0, G_{i+1} is a $(G_i, \mathcal{E}_i)$-produced graph for $0 \leq i \leq t$ (Definition 39). Hence, $\Delta(G_{i+1}) \leq \Delta(G_i) + 1$ for each i (Fact 22), which implies $\Delta(G_j) \leq \Delta(G_0) + j \leq \Delta(G_0) + t + 1$ for each $0 \leq j \leq t + 1$. □

Fact 43. *Suppose a sequence of LA edit pairs $\langle \mathcal{E}_i \rangle_0^t$ fits a graph G_0, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$. Then, $|V(\mathcal{X}[i])|$ and $|V(\mathcal{Y}[i])|$ are at most $(\Delta(G_0)+t+1)^2+1$ for each $0 \leq i \leq t$.*

Proof. Since $\langle \mathcal{E}_i \rangle_0^t$ fits G_0, we know $\mathcal{E}_i$ fits G_i for each $0 \leq i \leq t$. By Fact 19, therefore, $|V(\mathcal{X}[i])|$ and $|V(\mathcal{Y}[i])|$ are at most $\Delta(G_i)^2 + 1$. But $\Delta(G_i) \leq \Delta(G_0) + t + 1$ by Fact 42. Hence, $|V(\mathcal{X}[i])|$ and $|V(\mathcal{Y}[i])|$ are at most $(\Delta(G_0) + t + 1)^2 + 1$. □

3.9 Problem statement

We now state our problems.

Graph-Edit-to-NDL (GEN)	
Instance:	A triple $(G_0, \mathcal{T}, \ell)$, where G_0 is a graph, $\mathcal{T}$ is a Young tableau, and ℓ is an integer.
Question:	Is there an edit pair sequence $\langle(\mathcal{X}[i], \mathcal{Y}[i])\rangle_0^t$, where $t \leq \ell$, such that the graph $G_{t+1} = \mathit{gedit}^t(G_0, \langle(\mathcal{X}[i], \mathcal{Y}[i])\rangle_0^t)$, has NDL $\mathcal{T}$?

Constrained-Graph-Edit-to-NDL (CGEN)	
Instance:	A quadruple $(G_0, \pi, \mathcal{T}, \ell)$, where G_0 is a graph, π is a Young property verifiable in polynomial time, $\mathcal{T}$ is a Young tableau that satisfies π, and ℓ is an integer.
Question:	Is there an edit pair sequence $\langle(\mathcal{X}[i], \mathcal{Y}[i])\rangle_0^t$, where $t \leq \ell$, such that the graph $G_{t+1} = \mathit{gedit}^t(G_0, \langle(\mathcal{X}[i], \mathcal{Y}[i])\rangle_0^t)$, has NDL $\mathcal{T}$ and the NDL of each intermediate graph G_i satisfies π for $0 \leq i \leq t$?

Chapter 4

Complexity Results

In this chapter, we show that GEN and CGEN are NP-complete, which justifies our attempt to find FPT solutions for these problems. First, we show that these problems are verifiable in polynomial time (Theorems 46 and 47). Then, we show GEN is NP-complete by showing a reduction from Vertex-Cover in Theorem 48. Finally, we reduce GEN to CGEN in Theorem 49, which proves CGEN is also NP-complete.

A vertex cover V_C of a graph G is a subset of $V(G)$ such that every edge in $E(G)$ has at least one end-vertex in V_C. Given an input graph G and an integer k, Vertex-Cover asks whether G has a vertex cover of size at most k. A Vertex-Cover instance is denoted (G, k).

Lemmas 44 and 45 are used to show GEN is verifiable in polynomial time.

Lemma 44. *We can check whether an edit pair (X, Y) fits a graph G in time polynomial in $|V(G)|$.*

Proof. To check whether an edit pair (X, Y) fits a graph G, we need to determine whether the conditions in Definition 9 hold.

First, we can check whether X is an induced subgraph of G in time polynomial in $|V(X)|$, and so polynomial in $|V(G)|$.

Second, checking whether the inserted vertex $y \in V(Y) \setminus V(X)$, if any, is not already a vertex of G can be done in constant time.

Finally, checking whether $d_X(z) = d_G(z)$ for the deleted vertex $z \in V(X) \setminus V(Y)$, if any, takes at most $|V(G)|$ steps.

Hence, we can check whether (X, Y) is applicable in G in time polynomial in $|V(G)|$. □

Lemma 45. *Given an edit pair (X, Y) that fits a graph G, the operation gedit$(G, (X, Y))$ can be performed in time polynomial in $|V(G)|$.*

Proof. By Definition 9, the *gedit* operation computes the sets $(V(G) \setminus V(X)) \cup V(Y)$ and $(E(G) \setminus E(X)) \cup E(Y)$, which can be done in time polynomial in $|V(G)|$. □

Theorem 46. GRAPH-EDIT-TO-NDL *is in NP.*

Proof. Let $(G_0, \mathcal{T}, \ell)$ be an instance of GEN and an edit pair sequence $\langle(\mathcal{X}[i], \mathcal{Y}[i])\rangle_0^t$ for $t < \ell$ be a certificate. To verify that the graph edit sequence $(\mathcal{X}[i], \mathcal{Y}[i])$ is a solution, the certifier must verify that $\langle(\mathcal{X}[i], \mathcal{Y}[i])\rangle_0^t$ fits G_0 and $NDL(G_{t+1}) = \mathcal{T}$, where $G_{t+1} = gedit^t(G_0, \langle(\mathcal{X}[i], \mathcal{Y}[i])\rangle_0^t)$. We show that both the tasks can be performed in time polynomial in the size of the instance.

Given a graph G_i for $0 \leq i \leq t$, we can check whether $(\mathcal{X}[i], \mathcal{Y}[i])$ fits G_i in time polynomial in $|V(G_i)|$ by Lemma 44. Furthermore, by Lemma 45, if $(\mathcal{X}[i], \mathcal{Y}[i])$ fits G_i, then the operation $gedit(G_i, (\mathcal{X}[i], \mathcal{Y}[i]))$ can be performed in time polynomial in $|V(G_i)|$ to obtain G_{i+1}.

Since $|V(G_{i+1})| > |V(G_i)|$ only if a vertex is inserted in G_i, in which case $|V(G_{i+1})| = |V(G_i)| + 1$, it follows that $|V(G_i)| \leq |V(G_0)| + t < |V(G_0)| + \ell$ for $0 \leq i \leq t$.

Therefore, we can check whether $(\mathcal{X}[i], \mathcal{Y}[i])$ fits G_i for each $0 \leq i \leq t$ and, if so, determine the graph sequence $\langle G_{i+1}\rangle_0^t$ in time polynomial in $(|V(G_0)| + \ell) \cdot t < (|V(G_0)| + \ell) \cdot \ell$, which is polynomial in $|V(G_0)| + \ell$.

Now it remains to check whether $NDL(G_{t+1}) = \mathcal{T}$, which can be done in time polynomial in $|V(G_{t+1})| \leq |V(G_0)| + \ell$.

Thus, we can check whether the certificate $\langle(\mathcal{X}[i], \mathcal{Y}[i])\rangle_0^t$ is a solution to $(G_0, \mathcal{T}, \ell)$ in time polynomial in $|V(G_0)| + \ell$.

Now, the length of the certificate sequence t is bounded by $\mathcal{O}(|V(G_0)|^2)$ because, otherwise, for at least one vertex or edge, there would be a pair of insertion and deletion of that vertex/edge in the sequence. We could then remove the pair of graph edits to form a shorter sequence that produces the same graph.

Hence, we can check whether the certificate $\langle(\mathcal{X}[i], \mathcal{Y}[i])\rangle_0^t$ is a solution to $(G_0, \mathcal{T}, \ell)$ in time polynomial in $|V(G_0)|$, which implies GEN is in NP. □

Theorem 47. CONSTRAINED-GRAPH-EDIT-TO-NDL *is in NP.*

Proof. Let $(G_0, \pi, \mathcal{T}, \ell)$ be an instance of CGEN and a graph edit sequence $\langle(\mathcal{X}[i], \mathcal{Y}[i])\rangle_0^t$ for $t < \ell$ be a certificate. By Theorem 46, we can verify whether $\langle(\mathcal{X}[i], \mathcal{Y}[i])\rangle_0^t$ fits G_0 and produces a sequence of graphs $\langle G_i\rangle_0^{t+1}$ such that $NDL(G_{t+1}) = \mathcal{T}$ in time polynomial in $|V(G_0)| + \ell$. Moreover, the sequence of graphs $\langle G_i\rangle_0^{t+1}$ is also determined.

Since π is verifiable in polynomial time, checking whether $NDL(G_i)$ satisfies π takes time polynomial in $|NDL(G_i)|$ for $0 \leq i \leq t+1$. This in turn is polynomial in $|V(G_i)|$, and therefore, polynomial in $|V(G_0)| + \ell$. Therefore, checking whether the NDL of each

graph in $\langle G_i \rangle_0^{t+1}$ satisfies π takes time polynomial in $(|V(G_0)| + \ell) \cdot t < (|V(G_0)| + \ell) \cdot \ell$, which is polynomial in $|V(G_0)| + \ell$.

Hence, checking whether $\langle (\mathcal{X}[i], \mathcal{Y}[i]) \rangle_0^t$ is a solution to $(G_0, \pi, \mathcal{T}, \ell)$ takes time polynomial in $|V(G_0)| + \ell$.

Now, the length of the certificate sequence t is bounded by $\mathcal{O}(|V(G_0)|^2)$ because, otherwise, for at least one vertex or edge, there would be a pair of insertion and deletion of that vertex/edge in the sequence. We could then remove the pair of graph edits to form a shorter sequence that produces the same graph.

Hence, we can check whether the certificate $\langle (\mathcal{X}[i], \mathcal{Y}[i]) \rangle_0^t$ is a solution to $(G_0, \pi, \mathcal{T}, \ell)$ in time polynomial in $|V(G_0)|$, which implies CGEN is in NP. □

Theorem 48. GRAPH-EDIT-TO-NDL *is NP-complete.*

Proof. First, to show that GEN is NP-hard, we reduce VERTEX-COVER, which is known to be NP-hard [19], to GEN. Let (G, k) be an instance of VERTEX-COVER. We claim that (G, k) is a YES-instance if and only if $(G, \mathcal{T}_E, k)$ is a YES-instance of GEN, where $\mathcal{T}_E$ is the empty Young tableau.

Suppose (G, k) is a YES-instance. Then there exists a vertex cover V_C of G of size at most k. Since each edge of G has at least one end-vertex in V_C, deleting the vertices of V_C in G yields a graph G' with an empty edge set and, hence, an empty NDL. Moreover, $|V_C| \leq k$ and so we need to perform at most k vertex deletions to obtain G' from G. Therefore, $(G, \mathcal{T}_E, k)$ is a YES-instance.

Now, suppose $(G, \mathcal{T}_E, k)$ is a YES-instance. There exists an edit pair sequence $\mathcal{S}$ of length at most k which produces a graph G' with an empty NDL or, in other words, with no edges. We construct a vertex cover V_C of G as follows. The set V_C is initially set to empty and first we add the vertices deleted in S to V_C. Then, for each edge (u, v) deleted by an edit pair in $\mathcal{S}$, if $u, v \in V(G)$ and $u, v \notin V_C$, we arbitrarily choose one of u and v and add it to V_C.

We claim that every edge of G has at least one end-vertex in C. Consider an edge $e \in E(G)$. Because $E(G')$ is empty, e is deleted by an edit pair in $\mathcal{S}$. If e is deleted by the deletion of one of its end-vertices, then, by our construction, the deleted end-vertex is in V_C. If, on the other hand, e is deleted by an edge deletion, then again, by our construction, one of its end-vertices is in V_C. Note that e cannot be deleted by either edge insertion or vertex insertion. Therefore, V_C is a vertex cover of G. Furthermore, since $\mathcal{S}$ contains at most k edit pairs and we add at most one vertex per edit pair in our construction of V_C, $|V_C| \leq k$. Hence, (G, k) is a YES-instance.

Since GEN is in NP by Theorem 46, GEN is NP-complete. □

Theorem 49. CONSTRAINED-GRAPH-EDIT-TO-NDL *is NP-complete.*

Proof. Suppose $(G, \mathcal{T}, \ell)$ is an instance of GEN. Consider the CGEN instance $(G, \mathcal{T}, \ell, \pi_0)$, where the property π_0 imposes no constraints on the NDLs of graphs obtained in the sequence. It is obvious that $(G, \mathcal{T}, \ell)$ is a YES-instance if and only if $(G, \mathcal{T}, \ell, \pi_0)$ is a YES-instance. Therefore, since GEN is NP-hard by Theorem 48, CGEN is NP-hard. Furthermore, since CGEN is in NP by Theorem 47, CGEN is NP-complete. □

Chapter 5

Preliminary Results

We have stated GEN in terms of graph edit − the solution is an edit pair sequence that produces a graph with the desired NDL. In Section 5.1, we restate the problem in terms of NDL graph edit. In other words, we show that GEN is equivalent to determining whether there exists an LA edit pair sequence that is complete in the input graph and produces a graph with the desired NDL (Theorem 54). This is the first step toward designing an FPT solution for GEN as we will show in Chapter 7 that the search space of candidate LA edit pair sequences is bounded by a function of $\Delta(G_0) + \ell$.

In Section 5.2, we give a procedure − Compute-NDL − to compute the NDL of a $(G, \mathcal{E})$-produced graph using only $NDL(G)$ and $\mathcal{E}$ when $\mathcal{E}$ is complete in G, and show that every $(G, \mathcal{E})$-produced graph has the same NDL (Corollary 58). This implies every $(G_0, \langle \mathcal{E}_i \rangle_0^t)$-produced graph has the same NDL, and this NDL is determined by $NDL(G_0)$ and $\langle \mathcal{E}_i \rangle_0^t$ (Corollaries 59 and 60).

5.1 Restatement of GEN and CGEN in terms of NDL graph edit

Lemma 50. *If G' is a $(G, \mathcal{E})$-produced graph for an LA edit pair $\mathcal{E}$ and a graph G, then there exists an edit pair (A, B) such that $G' = gedit(G, (A, B))$.*

Proof. Suppose $G' = ndledit(\Phi)$, where $\Phi = (G, \mathcal{E}, H, f)$ and $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$ is complete in G. We know that $(X; \Lambda_X) \simeq H$ with respect to f (Definition 16), and so X is isomorphic to H with respect to f.

Note that the NDL-tuple Φ fully specifies the modifications to G and to $NDL(G)$, which means H contains vertices of G whose NDSs, but not degrees, are modified. We

exclude such vertices in our construction of an edit pair (A, B), which specifies only the modifications to G. Then we show that $G' = \mathit{gedit}(G, (A, B))$.

Let V_D contain the vertices of G whose degrees in G and G' differ, or, $V_D = \{f(r) \mid r \in M_{deg}(\mathcal{E})\}$ (Fact 27).

Let A be the subgraph induced in G by $V(A) = V_-(\Phi) \cup V_D$ and let B be the subgraph induced in G' by $V(B) = V_+(\Phi) \cup V_D$. We show (A, B) is an edit pair. If $\mathcal{E}$ is a vertex deletion where $z \in V_-(\mathcal{E})$, then since X is isomorphic to an induced subgraph of G and $N_X(z) = M_{deg}(\mathcal{E})$ by Definition 12, $N_G(f(z)) = V_D$. Also, $f(z) \notin V(B)$, and hence (A, B) is an edit pair by Definition 8. For other edit operations, we may verify from Definition 8 that (A, B) is an edit pair.

First, we show that (A, B) fits G by proving (A, B) satisfies the three conditions in Definition 9.

1. Clearly, A is an induced subgraph of G.

2. We know $V(B) \setminus V(A) = V_+(\Phi) = V(G') \setminus V(G)$, which means $(V(B) \setminus V(A)) \cap V(G) = \emptyset$.

3. Observe that $V(A) \setminus V(B) = V_-(\mathcal{E})$. For $z \in V_-(\mathcal{E})$, if any, we know $N_X(z) = M_{deg}(\mathcal{E})$ by Definition 12 and we have shown $N_G(f(z)) = V_D$. Since we have defined $V_D = \{f(r) \mid r \in M_{deg}(\mathcal{E})\}$, it follows that $|N_X(z)| = |N_G(f(z))|$, i.e., $d_X(z) = d_G(f(z))$.

Now, to prove that $\mathit{gedit}(G, (A, B)) = G'$, we first note that $V_+(\Phi) = V(B) \setminus V(A)$ and $V_-(\Phi) = V(A) \setminus V(B)$, which implies $V(G') = (V(G) \setminus V(A)) \cup V(B)$ (Definition 18).

To complete the proof, we show $E_+(\Phi) = E(B) \setminus E(A)$ and $E_-(\Phi) = E(A) \setminus E(B)$, which would imply $E(G') = (E(G) \setminus E(A)) \cup E(B)$ as needed.

First, to show $E_+(\Phi) = E(B) \setminus E(A)$, we consider the subgraph C of B induced by V_D. Note that the inserted vertex in $V_+(\Phi) = V(B) \setminus V(A)$, if any, is isolated. Therefore, $E(C) = E(B)$ and $V(C) \subseteq V(A)$. Since C is also an induced subgraph of G', for the inserted edge e, if any, $e \in E(C) \setminus E(A)$ if and only if $e \in E(G') \setminus E(G) = E_+(\Phi)$, which implies $E(B) \setminus E(A) = E_+(\Phi)$.

Next, to show $E_-(\Phi) = E(A) \setminus E(B)$, we note that $V(A) \setminus V(B) = V_-(\Phi)$. Also, for $z \in V_-(\Phi)$, if any, we delete along with z all edges incident with z from G. Therefore $E(A) \setminus E(B) = E(G) \setminus E(G') = E_-(\Phi)$. □

Lemma 50 can be extended to apply to LA edit pair sequences.

Corollary 51. *Suppose an LA edit pair sequence $\langle \mathcal{E}_i \rangle_0^t$ is complete in a graph G_0 and G_{t+1} is a $(G_0, \langle \mathcal{E}_i \rangle_0^t)$-produced graph for some $t \geq 0$. Then there exists an edit pair sequence $\langle (\mathcal{X}[i], \mathcal{Y}[i]) \rangle_0^t$ such that $gedit^t(G_0, \langle (\mathcal{X}[i], \mathcal{Y}[i]) \rangle_0^t) = G_{t+1}$.*

Proof. Since $\langle \mathcal{E}_i \rangle_0^t$ is complete in G_0, by Definition 40, there exists a sequence of graphs $\langle G_i \rangle_0^{t+1}$ such that $\mathcal{E}_i$ is complete in G_i and G_{i+1} is a $(G_i, \mathcal{E}_i)$-produced graph for each $0 \leq i \leq t$. We know by Lemma 50 that there exists an edit pair $(\mathcal{X}[i], \mathcal{Y}[i])$ such that $gedit(G_i, (\mathcal{X}[i], \mathcal{Y}[i])) = G_{i+1}$ for each $0 \leq i \leq t$. Therefore, $gedit^t(G_0, \langle (\mathcal{X}[i], \mathcal{Y}[i]) \rangle_0^t) = G_{t+1}$. □

The converse of Lemma 50 holds as well.

Lemma 52. *Suppose $gedit(G, (A, B)) = G'$ for a graph G and an edit pair (A, B). Then there exists an LA edit pair $\mathcal{E}$ that is complete in G such that G' is a $(G, \mathcal{E})$-produced graph.*

Proof. Suppose $gedit(G, (A, B)) = G'$. Then A is an induced subgraph of G by Definition 9 and B is an induced subgraph of G' by Fact 10. We will construct an LA edit pair $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$ by "expanding" A and B to include vertices of G whose NDSs alone are changed. We then prove that $\mathcal{E}$ is complete in G and G' is a $(G, \mathcal{E})$-produced graph.

Let D contain the "degree-modified" vertices of G, i.e., $D = \{w \in V(A) \cap V(B) \mid d_A(w) \neq d_B(w)\}$. Let L contain the "NDS-modified" neighbours of the vertices in D. More precisely, $L = \left(\bigcup_{w \in D} N_G(w)\right) \setminus \Big(V(A) \cup V(B)\Big)$. Now, let X be the subgraph induced by $V(A) \cup L$ in G and Y be the subgraph induced by $V(B) \cup L$ in G'. Furthermore, let Λ_X and Λ_Y be list-attributions given by $\Lambda_X(x) = NDS(G, x)$ for each $x \in V(X)$ and $\Lambda_Y(y) = NDS(G', y)$ for each $y \in V(Y)$, respectively.

First, we claim that $\mathcal{E}$ fits G. By our definition of Λ_X, $(X; \Lambda_X) \simeq X$ with respect to the identity function f_{id}. For the deleted vertex $a \in V_-(\mathcal{E}) = V(A) \setminus V(B)$, if any, it holds that $d_A(a) = d_G(a)$ because (A, B) fits G (Definition 9). From our definition of X, then, it follows that $d_X(a) = d_A(a) = d_G(a)$. Finally, note that $M_{deg}(\mathcal{E}) = D$ because L is disjoint from $V(A) \cup V(B)$, and so the degrees of the vertices in L are unaffected by the graph edit. Now, for each $w \in D$, $N_X(w) = N_G(w)$ by our definitions of L and X, and so $d_X(w) = d_G(w)$. Hence, $\mathcal{E}$ fits G by Definition 16.

Next, it follows from our definition of Λ_Y and Definition 26 that $\mathcal{E}$ is also complete in G.

Finally, observe that $\mathcal{E}$ preserves the modifications specified by (A, B). This is because

1. $V_+(\mathcal{E}) = V(Y) \setminus V(X) = V(B) \setminus V(A)$,

2. $V_-(\mathcal{E}) = V(X) \setminus V(Y) = V(A) \setminus V(B)$, and

3. $E_+(\mathcal{E}) = E(B) \setminus E(A)$ and $E_-(\mathcal{E}) = E(A) \setminus E(B)$ because A and B are induced subgraphs of X and Y, respectively, by our definitions of X and Y.

Thus, $ndledit(\Phi) = G'$, where $\Phi = (G, \mathcal{E}, X, f_{id})$. □

Corollary 53. *Suppose* $gedit^t(G_0, \langle(\mathcal{X}[i], \mathcal{Y}[i])\rangle_0^t) = G_{t+1}$*, for some graph* G_0 *and edit pair sequence* $\langle(\mathcal{X}[i], \mathcal{Y}[i])\rangle_0^t$*. Then, there exists an LA edit pair sequence* $\langle\mathcal{E}_i\rangle_0^t$ *that is complete in* G_0 *such that* G_{t+1} *is a* $(G_0, \langle\mathcal{E}_i\rangle_0^t)$*-produced graph.*

Proof. Since $(\mathcal{X}[i], \mathcal{Y}[i])\rangle_0^t$ fits G_0, there exists a sequence of graphs $\langle G_i\rangle_0^{t+1}$ such that $G_{i+1} = gedit(G_i, (\mathcal{X}[i], \mathcal{Y}[i]))$ for each $0 \leq i \leq t$ (Definition 38). We know by Lemma 52 that there exists an LA edit pair $\mathcal{E}_i$ that is complete in G_i such that G_{i+1} is a $(G_i, \mathcal{E}_i)$-produced graph for each $0 \leq i \leq t$. Therefore, the LA edit pair sequence $\langle\mathcal{E}_i\rangle_0^t$ is complete in G_0 (Definitions 39 and 40) and G_{t+1} is a $(G_0, \langle\mathcal{E}_i\rangle_0^t)$-produced graph. □

Theorem 54 follows from Corollaries 51 and 53.

Theorem 54. *A* GEN *instance* $(G_0, \mathcal{T}, \ell)$ *is a* YES*-instance if and only if there exists an LA edit pair sequence* $\langle\mathcal{E}_i\rangle_0^t$*, where* $t < \ell$*, such that* $\langle\mathcal{E}_i\rangle_0^t$ *is complete in* G_0 *and there exists a* $(G_0, \langle\mathcal{E}_i\rangle_0^t)$*-produced graph whose NDL is* $\mathcal{T}$*.*

5.2 NDL of $(G, \mathcal{E})$-produced graphs

We give a procedure − COMPUTE-NDL − to compute the NDL of a $(G, \mathcal{E})$-produced graph, where $\mathcal{E}$ is complete in G, and prove its correctness in Lemma 55. Furthermore, we show in Corollary 58 that every $(G, \mathcal{E})$-produced graph has the same NDL.

Suppose $ndledit(\Phi) = G'$, where $\Phi = (G, \mathcal{E}, H, f)$. Since $\mathcal{E}$ is complete in G, we know that Λ_X gives the NDSs of the vertices of the "replaced" subgraph H (Definitions 16 and 18) and Λ_Y gives the NDSs of the vertices of the "replacing" subgraph of G' (Definition 26 and Fact 29). Moreover, the NDSs of the "untouched" vertices of G, i.e., the vertices in $V(G) \setminus V(H)$ remain unchanged in G' (Fact 28). In the procedure COMPUTE-NDL, we remove the NDSs of the vertices of H from $NDL(G)$ before adding the nonincreasing lists of Λ_Y to obtain $NDL(G')$.

Lemma 55. *The procedure* COMPUTE-NDL *correctly determines the NDL of a* $(G, \mathcal{E})$*-produced graph, where* $\mathcal{E}$ *is complete in* G*.*

Algorithm 1 Compute the NDL of a $(G, \mathcal{E})$-produced graph

1: **procedure** COMPUTE-NDL($NDL(G)$,$\mathcal{E}$)
2: **Input:** The NDL of a graph G and an LA edit pair $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$.
3: **Output:** $NDL(G')$, where G' is a $(G, \mathcal{E})$-produced graph.
4: Set $\mathcal{M} = NDL(G)$
5: **for** each vertex $x \in V(X)$ **do**
6: Remove a nonincreasing list equal to $\Lambda_X(x)$ from $\mathcal{M}$
7: Set $\mathcal{M}$ to the resulting Young tableau
8: **for** each vertex $y \in V(Y)$ **do**
9: Add the nonincreasing list $\Lambda_Y(y)$ to $\mathcal{M}$
10: Set $\mathcal{M}$ to the resulting Young tableau
11: **return** $\mathcal{M}$

Proof. Suppose $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$ is complete in a graph G, and $G' = ndledit(\Phi)$, where $\Phi = (G, \mathcal{E}, H, f)$. Suppose too that the Young tableau $\mathcal{M}$ is the output of COMPUTE-NDL$(G, \mathcal{E})$. We first claim that, for each vertex $v \in V(G')$, $NDS(G', v)$ is present in $\mathcal{M}$. Then, we claim that each nonincreasing list of $\mathcal{M}$ corresponds to the NDS of a vertex of G'. Together, the claims imply each nonincreasing list of $\mathcal{M}$ corresponds to a unique vertex of G' and vice versa, and so $\mathcal{M} = NDL(G')$.

We divide the vertices of G' into three kinds as in Lemma 23 and prove our first claim for each kind. Let $v \in V(G')$.

Case 1. $v \in V(G') \setminus V(G)$. Then, v is the inserted vertex and $NDS(G', v)$ is an empty list, and so there is nothing to prove.

Case 2. $v \in V(G) \setminus V(H)$. Then, we know $NDS(G, v) = NDS(G', v)$ by Fact 28. Since $v \notin V(H)$, we do not remove $NDS(G, v)$ in the procedure, and so $NDS(G, v)$ is present in $\mathcal{M}$.

Case 3. $v = f(w)$ for some retained vertex $w \in V_R(\mathcal{E})$. Then $\Lambda_Y(w) = NDS(G', f(w))$ because $\mathcal{E}$ is complete in G (Definition 26), and $\Lambda_Y(w)$ is present in $\mathcal{M}$ (Line 9).

Now, we show that each nonincreasing list $\mathcal{L}$ of $\mathcal{M}$ corresponds to the NDS of a vertex of G'. Observe that either $\mathcal{L} = \Lambda_Y(y)$ for some $y \in V(Y)$ (Line 9) or $\mathcal{L} = NDS(G, u)$ for some $u \in V(G)$ because we initially set $\mathcal{M} = NDL(G)$ (Line 4).

If $\mathcal{L} = \Lambda_Y(y)$ for some $y \in V(Y)$, then $y \in V(X)$ because, otherwise, $\Lambda_Y(y)$ would be empty (Definition 26). Hence, $y \in V(X) \cap V(Y) = V_R(\mathcal{E})$, which implies $\mathcal{L} = NDS(G', f(y))$ (Definition 26).

On the other hand, if $\mathcal{L} = NDS(G, u)$ for some $u \in V(G)$, then obviously there exists no vertex $x \in V(X)$ such that $u = f(x)$ because, otherwise, we would have removed $\Lambda_X(x) = NDS(G, u)$ (Line 6). Therefore, $u \notin V(H)$, and so u is an "untouched" vertex, or, $u \in V(G) \setminus V(H)$. Hence, $NDS(G, u) = NDS(G', u)$ by Fact 28, which implies $\mathcal{L} = NDS(G', u)$. □

Lemma 56. *The running time of* Compute-NDL *is polynomial in $|V(G)|$, where G is the input graph.*

Proof. Since a vertex can have at most $\Delta(G)$ neighbours in G, its NDS has size at most $\Delta(G)$. Hence, each nonincreasing list of $\mathcal{M} = NDL(G)$ has size at most $\Delta(G)$. Since $\mathcal{E}$ fits G, $|V(X)|$ and $|V(Y)|$ are at most $\Delta(G)^2 + 1$ (Fact 19). Therefore, we remove at most $\Delta(G)^2 + 1$ nonincreasing lists from $\mathcal{M}$ (Line 5) before adding at most $\Delta(G)^2 + 1$ nonincreasing lists to $\mathcal{M}$ (Line 8). This takes at most $2(\Delta(G)^2 + 1)$ addition and removal operations in total and each operation takes time linear in $|NDL(G)| = |V(G)|$ (see Section 3.3), and so the running time is bounded by $\mathcal{O}(\Delta(G)^2 \cdot |V(G)|)$, which is polynomial in $|V(G)|$ since $\Delta(G) \leq |V(G)|$. □

Corollaries 57 and 58 follow from the fact that Compute-NDL is well-defined and takes only $NDL(G)$ and $\mathcal{E}$ as inputs.

Corollary 57. *If G' is a $(G, \mathcal{E})$-produced graph, where $\mathcal{E}$ is complete in G, then $NDL(G')$ is determined by $NDL(G)$ and $\mathcal{E}$.*

Corollary 58. *If an LA edit pair $\mathcal{E}$ is complete in a graph G, then every $(G, \mathcal{E})$-produced graph has the same NDL.*

We extend the above corollaries to sequences of LA edit pairs.

Corollary 59. *If G_{t+1} is a $(G_0, \langle \mathcal{E}_i \rangle_0^t)$-produced graph, where $\langle \mathcal{E}_i \rangle_0^t$ is complete in G_0, then $NDL(G_{t+1})$ is determined by $NDL(G_0)$ and $\langle \mathcal{E}_i \rangle_0^t$.*

Proof. Since $\langle \mathcal{E}_i \rangle_0^t$ is complete in G_0, there exists a sequence of graphs $\langle G_j \rangle_0^{t+1}$ such that $\mathcal{E}_i$ is complete in G_i and G_{i+1} is a $(G_i, \mathcal{E}_i)$-produced graph for each $0 \leq i \leq t$ (Definition 40). We know by Corollary 57 that $NDL(G_{i+1})$ is determined by $NDL(G_i)$ and $\mathcal{E}_i$ for each $0 \leq i \leq t$, which completes the proof. □

Corollary 60 follows from Corollary 58.

Corollary 60. *If an LA edit pair sequence $\langle \mathcal{E}_i \rangle_0^t$ is complete in a graph G_0, then every $(G_0, \langle \mathcal{E}_i \rangle_0^t)$-produced graph has the same NDL.*

Chapter 6

Provisional Completeness of an LA Edit Pair Sequence

To determine the completeness of an LA edit pair $\mathcal{E}$ in a graph G using Definition 26, we need to check whether the conditions in the definition hold for each $(G, \mathcal{E})$-produced graph. Thus we need to first compute all $(G, \mathcal{E})$-produced graphs. In this chapter, we give a procedure − Check-Provisional-Completeness − to determine the provisional completeness of $\mathcal{E}$ (Definition 30). Then, we prove that, if $\mathcal{E}$ fits G, checking whether $\mathcal{E}$ is complete in G is equivalent to checking its provisional completeness (Lemma 64). Finally, we show that the relation between provisional completeness and completeness in a graph can be extended to LA edit pair sequences (Lemma 65).

6.1 Check-Provisional-Completeness procedure

The procedure Check-Provisional-Completeness checks the conditions under which an LA edit pair $\mathcal{E}$ that fits a graph G is complete in G. By Definition 30, therefore, the procedure checks the provisional completeness of $\mathcal{E}$. If $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$ is complete in G and G' is a $(G, \mathcal{E})$-produced graph, then Y is list-isomorphic to an induced subgraph H' of G' (Fact 24), which means Λ_Y correctly specifies the NDSs of the vertices of H'. We specify how the list-attributions Λ_Y and Λ_X must be related for this condition to hold.

We use the fact that the vertices of Y can be divided into the inserted vertex, if any, and the retained vertices (Definition 14). In turn, the retained vertices are divided into degree-modified vertices and the list-modified vertices (Fact 13). We check three sets of conditions pertaining to these types of vertices.

1. $y \in V(Y)$ is the inserted vertex. Then, we check whether $\Lambda_Y(y)$ is empty.

2. $p \in V(Y)$ is a degree-modified vertex. Then, its NDS is changed in three cases. We perform the changes specified in the cases and check whether the nonincreasing list obtained equals $\Lambda_Y(p)$.

   (a) The deleted vertex is a neighbour of p. Then, we remove the degree of the deleted vertex from $\Lambda_X(p)$.

   (b) An edge incident with p is inserted or deleted. Then, we add the degree of the other end-vertex of the edge to $\Lambda_X(p)$ or remove it from $\Lambda_X(p)$, respectively.

   (c) A neighbour of p is degree-modified. Then, we update the degree of the degree-modified neighbour in $\Lambda_X(p)$.

3. $q \in V(Y)$ is a list-modified vertex. Then, we update the degrees of its degree-modified neighbours in $\Lambda_X(q)$ and check whether the nonincreasing list thus obtained equals $\Lambda_Y(q)$.

In Check-Provisional-Completeness, we define the variables *ins*, *degmod*, and *listmod* to correspond, respectively, to the three conditions above. Their boolean values indicate whether the corresponding condition holds. We output yes if and only if all three conditions hold.

To check Conditions (2) and (3), we assign $\mathcal{L}_p$ and $\mathcal{L}_q$ to $\Lambda_X(p)$ and $\Lambda_X(q)$, respectively. We then modify them and check whether they are equal to $\Lambda_Y(p)$ and $\Lambda_X(q)$, respectively.

Lemma 61. *Given an LA edit pair $\mathcal{E}$ that fits a graph G, the procedure* Check-Provisional-Completeness *correctly determines whether $\mathcal{E}$ is complete in G.*

Proof. Let $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$ be an LA edit pair that fits a graph G. We show that the conditions for the completeness of $\mathcal{E}$ in G (Definition 26) are satisfied if and only if the conditions checked by Check-Provisional-Completeness are satisfied.

We know $V(Y) = V_R(\mathcal{E}) \cup V_+(\mathcal{E})$ (Definition 14). Therefore, $V(Y) = M_{deg}(\mathcal{E}) \cup M_{list}(\mathcal{E}) \cup V_+(\mathcal{E})$ because $V_R(\mathcal{E}) = M_{deg}(\mathcal{E}) \cup M_{list}(\mathcal{E})$ (Fact 13).

Suppose $\mathcal{E}$ fits G and $G' = \mathit{ndledit}(\Phi)$, where $\Phi = (G, \mathcal{E}, H, f)$. We know $(X; \Lambda_X) \simeq H$ with respect to f (Definition 16), and so $\Lambda_X(x) = \mathit{NDS}(G, f(x))$ for each $x \in V(X)$. To check whether $\mathcal{E}$ is complete in G, by Definition 26, we need to check whether $\Lambda_Y(y)$ is an empty list for $y \in V_+(\mathcal{E})$ and whether $\Lambda_Y(r) = \mathit{NDS}(G', f(r))$ for each $r \in V_R(\mathcal{E})$. Since $V_R(\mathcal{E}) = M_{deg}(\mathcal{E}) \cup M_{list}(\mathcal{E})$, these can be split into three conditions.

1. $\Lambda_Y(y)$ is an empty list for $y \in V_+(\mathcal{E})$,

Algorithm 2 Check provisional completeness of an LA edit pair

procedure CHECK-PROVISIONAL-COMPLETENESS($\mathcal{E}$)
  Input: An LA edit pair $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$.
  Output: YES if $\mathcal{E}$ is provisionally complete; NO otherwise.
  Set *ins* = *degmod* = *listmod* = FALSE
  ▷ Initialize boolean variables
  if $V_+(\mathcal{E})$ is empty **or** $\Lambda_Y(y)$ is empty for $y \in V_+(\mathcal{E})$ **then**
  ▷ Condition (1)
    Set *ins* = TRUE
  for each degree-modified vertex $p \in M_{deg}(\mathcal{E})$ **do**
    Set $\mathcal{L}_p = \Lambda_X(p)$
    for the deleted neighbour $z \in V_-(\mathcal{E}) \cap N_X(p)$, if any, **do**
      $\mathcal{L}_p = \mathcal{L}_p \ominus \Lambda_X(z)$
    for each degree-modified neighbour $r \in M_{deg}(\mathcal{E}) \cap N_X(p)$ **do**
      if $(p, r) \in E_+(\mathcal{E})$ **then**
        $\mathcal{L}_p = \mathcal{L}_p \oplus \Lambda_Y(r)$
      else if $(p, r) \in E_-(\mathcal{E})$ **then**
        $\mathcal{L}_p = \mathcal{L}_p \ominus \Lambda_Y(r)$
      else
        $\mathcal{L}_p = (\mathcal{L}_p \ominus d_X(r)) \oplus d_Y(r)$
  if $M_{deg}(\mathcal{E}) = \emptyset$ **or** $\Lambda_Y(p) = \mathcal{L}_p$ for each $p \in M_{deg}(\mathcal{E})$ **then**
  ▷ Condition (2)
    Set *degmod* = TRUE
  for each list-modified vertex $q \in M_{list}(\mathcal{E})$ **do**
    Set $\mathcal{L}_q = \Lambda_X(q)$
    for each degree-modified neighbour $r \in M_{deg}(\mathcal{E}) \cap N_X(q)$ **do**
      $\mathcal{L}_q = (\mathcal{L}_q \ominus d_X(r)) \oplus d_Y(r)$
  if $M_{list}(\mathcal{E}) = \emptyset$ **or** $\Lambda_Y(q) = \mathcal{L}_q$ for each $q \in M_{list}(\mathcal{E})$ **then**
  ▷ Condition (3)
    Set *listmod* = TRUE
  if *ins* = TRUE **and** *degmod* = TRUE **and** *listmod* = TRUE **then**
    return YES
  else
    return NO

2. $\Lambda_Y(p) = NDS(G', f(p))$ for each $p \in M_{deg}(\mathcal{E})$, and

3. $\Lambda_Y(q) = NDS(G', f(q))$ for each $q \in M_{list}(\mathcal{E})$.

We show that CHECK-PROVISIONAL-COMPLETENESS checks these conditions.

We check Condition (1) on Line 6.

To check Condition (2), we initialize $\mathcal{L}_p = \Lambda_X(p)$ for each degree-modified vertex p. We note that the NDS of $f(p)$ is modified in one of the three edit operations:

Case 1. Vertex deletion: a neighbour $z \in N_X(p)$ is deleted.

Now, the following hold:

1. The element $d_G(f(z))$ of $NDS(G, f(p))$ is absent from $NDS(G', f(p))$. Now, we know that $d_X(z) = d_G(f(z))$ because $\mathcal{E}$ fits G and $z \in V_-(\mathcal{E})$ (Definition 16). We ensure on Line 12 that $d_X(z)$ is absent from $\Lambda_Y(p)$ too.
2. Each $r \in N_X(z) \cap N_X(p)$ is degree-modified, i.e., $r \in M_{deg}(\mathcal{E})$. Therefore, the element $d_G(f(r))$ in $NDS(G, f(p))$ is updated to $d_{G'}(f(r))$ in $NDS(G', f(p))$. We know that $d_X(r) = d_G(f(r))$ since $\mathcal{E}$ fits G (Definition 16), and also that $d_Y(r) = d_{G'}(f(r))$ (Fact 25). Hence, we replace $d_X(r)$ in $\mathcal{L}_p$ with $d_Y(r)$ on Line 19.

Case 2. Edge insertion: the edge (p, r) is inserted for some $r \in V(X)$.

Obviously, r is degree-modified, or, $r \in M_{deg}(\mathcal{E})$, which implies $d_X(r) = d_G(f(r))$ and $d_Y(r) = d_{G'}(f(r))$ (Definition 16 and Fact 25, respectively). Now, $d_G(f(r))$ is absent from $NDS(G, f(p))$ but $d_{G'}(f(r))$ is present in $NDS(G', f(p))$. Therefore, we add $d_Y(r)$ to $\mathcal{L}_p$ on Line 15.

Case 3. Edge deletion: the edge (p, r) is deleted for some $r \in N_X(p)$.

We argue similarly to Case 2, except now $d_G(f(r))$ is present in $NDS(G, f(p))$ but $d_{G'}(f(r))$ is absent from $NDS(G', f(p))$. Therefore, we check on Line 17 whether $\Lambda_Y(p)$ is obtained from $\Lambda_X(p)$ by removing $d_X(r)$.

To check Condition (3), we initialize $\mathcal{L}_q = \Lambda_X(q)$ for each list-modified vertex q in $\mathcal{E}$. For each degree-modified neighbour $r \in M_{deg}(\mathcal{E})$, the element $d_G(f(r))$ in $NDS(G, f(q))$ is updated to $d_{G'}(f(r))$ in $NDS(G', f(q))$. Since we know that $d_X(r) = d_G(f(r))$ and $d_Y(r) = d_{G'}(f(r))$ (Definition 16 and Fact 25, respectively), we update $d_X(r)$ in $\mathcal{L}_q$ to $d_Y(r)$ on Line 26.

Finally, we output YES on Line 31 if and only if conditions (1), (2), and (3) are satisfied, i.e., $\mathcal{E}$ is complete in G. □

Corollary 62. CHECK-PROVISIONAL-COMPLETENESS *correctly determines the provisional completeness of an LA edit pair.*

Proof. Observe that CHECK-PROVISIONAL-COMPLETENESS takes only an LA edit pair $\mathcal{E}$ as input. By Lemma 61, this means CHECK-PROVISIONAL-COMPLETENESS outputs YES for an input $\mathcal{E}$ if and only if $\mathcal{E}$ is complete in every graph G it fits, or in other words, $\mathcal{E}$ is provisionally complete. □

Lemma 63. *Given an input* $\mathcal{E} = ((X;\Lambda_X),(Y;\Lambda_Y))$*, the procedure* CHECK-PROVISIONAL-COMPLETENESS *runs in time* $\mathcal{O}(|V(X)|^3)$*.*

Proof. Observe in CHECK-PROVISIONAL-COMPLETENESS that, for each $x \in V(X)$, an element of $\Lambda_X(x)$ is modified only if it corresponds to the degree of a neighbour of x in X. Since x can have at most $|V(X)|$ neighbours in X, $|\Lambda_X| \leq |V(X)|$ and we perform at most $2|V(X)|^2$ operations on the nonincreasing lists in Λ_X, considering an update of an element as a removal followed by an addition. Now, each operation takes time linear in the size of the list (see Section 3.3), i.e., linear in $|V(X)|$, which means the total time for performing all operations is in $\mathcal{O}(|V(X)|^3)$. Additionally, if a vertex y is inserted, then it takes $\mathcal{O}(1)$ time to check whether $\Lambda_Y(y)$ is empty. Therefore, the total time taken is in $\mathcal{O}(|V(X)|^3)$. □

Lemma 64. *An LA edit pair* $\mathcal{E}$ *is complete in a graph* G *if and only if* $\mathcal{E}$ *is provisionally complete and* $\mathcal{E}$ *fits* G*.*

Proof. If $\mathcal{E}$ fits G and $\mathcal{E}$ is provisionally complete, then $\mathcal{E}$ is complete in G (Definition 30).

On the other hand, if $\mathcal{E}$ is complete in G, then $\mathcal{E}$ must fit G (Definition 26). Since $\mathcal{E}$ fits G and is complete in G, we know by Lemma 61 that CHECK-PROVISIONAL-COMPLETENESS outputs YES, and so by Corollary 62, $\mathcal{E}$ is provisionally complete. □

Lemma 65. *An LA edit pair sequence is complete in a graph* G_0 *if and only if it is provisionally complete and fits* G_0*.*

Proof. Suppose an LA edit pair sequence $\langle\mathcal{E}_i\rangle_0^t$ is complete in a graph G_0. Then, $\langle\mathcal{E}_i\rangle_0^t$ fits G_0 and there exists a sequence of graphs $\langle G_i\rangle_0^{t+1}$ such that $\mathcal{E}_i$ is complete in G_i for each $0 \leq i \leq t$ (Definition 40). It follows from Lemma 64 that $\mathcal{E}_i$ is provisionally complete for each $0 \leq i \leq t$, and so $\langle\mathcal{E}_i\rangle_0^t$ is provisionally complete (Definition 41).

On the other hand, suppose $\langle\mathcal{E}_i\rangle_0^t$ is provisionally complete and fits G_0. Then, there exists a sequence of graphs $\langle G_j\rangle_0^{t+1}$ such that $\mathcal{E}_i$ fits G_i for $0 \leq i \leq t$ (Definition 39). Now, since $\mathcal{E}_i$ is provisionally complete, $\mathcal{E}_i$ is complete in G_i for $0 \leq i \leq t$ (Definition 41). Therefore, $\langle\mathcal{E}_i\rangle_0^t$ is complete in G_0 (Definition 40). □

Theorem 66 follows from Theorem 54 and Lemma 65.

Theorem 66. *A GEN instance $(G_0, \mathcal{T}, \ell)$ is a* YES*-instance if and only if there exists a provisionally complete LA edit pair sequence $\langle \mathcal{E}_i \rangle_0^t$ which fits G_0 such that there exists a $(G_0, \langle \mathcal{E}_i \rangle_0^t)$-produced graph with NDL $\mathcal{T}$.*

Chapter 7

Determining the completeness of an LA edit pair sequence in a graph

We have reduced GEN to determining the existence of a provisionally complete LA edit pair sequence that fits the input graph and produces a graph with the desired NDL (Theorem 66). Given an input graph G_0 and an LA edit pair sequence $\langle \mathcal{E}_i \rangle_0^t$ for $t \geq 0$, Compute-NDL computes the NDL of the $(G_0, \langle \mathcal{E}_i \rangle_0^t)$-produced graphs, if any, and Check-Provisional-Completeness checks whether $\langle \mathcal{E}_i \rangle_0^t$ is provisionally complete. In this chapter, we describe our method for determining whether a provisionally complete LA edit pair fits a graph, i.e., whether it is complete in the graph.

Now, $\langle \mathcal{E}_i \rangle_0^t$ fits G_0 only if there exists a sequence of graphs $\langle G_j \rangle_0^{t+1}$ such that $\mathcal{E}_i$ fits G_i for each $0 \leq i \leq t$ (Definition 39). However, the only graph we know in $\langle G_j \rangle_0^{t+1}$ is G_0. This is because, for any $0 \leq i \leq t$, $G_{i+1} = \mathit{ndledit}(\Phi_i)$, where $\Phi_i = (G_i, \mathcal{E}_i, H_i, f_i)$, and H_i and f_i are not fixed. Nevertheless, if $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ is provisionally complete and fits G_i, then $\mathcal{E}_i$ is complete in G_i, and so we know that $(\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]})$ is list-isomorphic to an induced subgraph of G_{i+1} (Fact 29).

Given that $\langle \mathcal{E}_j \rangle_0^i$ fits G_0, we reduce determining whether $\mathcal{E}_{i+1}$ fits G_{i+1} to a condition on G_0. To check whether $\mathcal{E}_{i+1}$ fits G_{i+1}, we need to determine whether $(\mathcal{X}[i+1]; \Lambda_{\mathcal{X}[i+1]})$ is list-isomorphic to an induced subgraph of G_{i+1} (Definition 16). If the latter is true, then we know that $\mathcal{X}[i+1]$ and $\mathcal{Y}[i]$ are isomorphic to induced subgraphs of G_{i+1}, and these subgraphs may share vertices and edges. Hence, we introduce the concept of merging two LA graphs by identifying subsets of their vertex-sets and edge-sets such that the nonincreasing lists of the identified vertices match. This allows us to construct a merge graph from $(\mathcal{X}[i+1]; \Lambda_{\mathcal{X}[i+1]})$ and $(\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]})$ that is list-isomorphic to a subgraph of G_{i+1}.

Next, we "undo" the NDL graph edit specified by $\mathcal{E}_i$ in the merge graph to obtain

its antecedent. Lemma 79, which we prove later, implies $(\mathcal{X}[i+1]; \Lambda_{\mathcal{X}[i+1]})$ is list-isomorphic to a subgraph of G_{i+1} if and only if the antecedent is list-isomorphic to a subgraph of G_i. Hence, we have reduced checking list-isomorphism with a subgraph of G_{i+1} to a list-isomorphism condition on G_i. Continuing this process of "undoing" NDL graph edits, we obtain an antecedent sequence and a list-isomorphism condition on G_0.

The antecedent sequence can be used to trace the history of a vertex and "undo" the changes to its degree. Since a graph edit is seen as "replacement" of a subgraph with another graph, a vertex may have been replaced multiple times. We call the vertex replaced a proxy of the vertex that replaces it.

This chain of proxies extends as far back as either the earliest antecedent or a vertex insertion, and we call the earliest proxy of a vertex its origin. The set of origins of the vertices of $\mathcal{X}[i+1]$ in the earliest antecedent induce what we call an origin graph. We show in Theorem 96 that the list-isomorphism and degree conditions on $(\mathcal{X}[i+1]; \Lambda_{\mathcal{X}[i+1]})$ and G_{i+1} can be reduced to list-isomorphism and degree conditions on the origin graph and G_0.

7.1 Merging LA graphs

Given two LA graphs $(A; \Lambda_A)$ and $(B; \Lambda_B)$, merging can be visualized as "laying" A over B such that the overlapping subgraphs of A and B are isomorphic and the corresponding nonincreasing lists match. We denote the vertex sets of these subgraphs as S_A and S_B, respectively, and the bijection from S_A to S_B as μ.

We obtain the merge graph $(M; \Lambda_M)$ by retaining the subgraph of A and removing the subgraph of B. But this also removes the edges of B that have exactly one end-vertex in S_B. To fix this, for each $a \in S_A$, we introduce edges between a and the neighbours of $\mu(a)$ in $V(B) \setminus S_B$. In Definition 67, E_{AB} contains the edges thus introduced.

Since S_A, S_B, and μ are not fixed, a family of merge graphs is obtained by merging $(A; \Lambda_A)$ and $(B; \Lambda_B)$, which we denote as $(A; \Lambda_A) \leftrightarrow (B; \Lambda_B)$.

Definition 67. Let $(A; \Lambda_A)$ and $(B; \Lambda_B)$ be LA graphs. Let $S_A \subseteq V(A)$ and $S_B \subseteq V(B)$ such that $|S_A| = |S_B|$ and $\mu : S_A \to S_B$ be a bijection such that

1. $A[S_A]$ and $B[S_B]$ are isomorphic with respect to μ, and
2. $\Lambda_A(v) = \Lambda_B(\mu(v))$ for each $v \in S_A$.

Merging $(A; \Lambda_A)$ and $(B; \Lambda_B)$ with respect to S_A, S_B, and μ produces an LA graph $(M; \Lambda_M)$ given by

$$\begin{aligned} V(M) &= V(A) \cup (V(B) \setminus S_B) \\ E(M) &= E(A) \cup E(B[V(B) \setminus S_B]) \cup E_{AB} \\ \Lambda_M(m) &= \begin{cases} \Lambda_A(m) & \text{if } m \in V(M) \cap V(A) \\ \Lambda_B(m) & \text{if } m \in V(M) \cap V(B) \end{cases} \end{aligned}$$

where $E_{AB} = \{(a, \widehat{b}) \mid a \in S_A \wedge \widehat{b} \in V(B) \setminus S_B \wedge (\mu(a), \widehat{b}) \in E(B)\}$. We also refer to the *merge graph* $(M; \Lambda_M)$ by $((A; \Lambda_A) \leftrightarrow (B; \Lambda_B), S_A, S_B, \mu)$.

Notice that the vertices in S_B do not appear in the merge graph. Instead, the vertices in S_A "substitute" for them in the merge graph. Let $\mu(a) = b$ for $a \in S_A$, $b \in S_B$, and a bijection μ. By Definition 67, for each neighbour v of a in M, either v or $\mu(v)$ is a neighbour of b in B. Hence, in a sense, a "substitutes" for b in M.

Definition 68. Let $(A; \Lambda_A)$ and $(B; \Lambda_B)$ be LA graphs and $(M; \Lambda_M) = ((A; \Lambda_A) \leftrightarrow (B; \Lambda_B), S_A, S_B, \mu)$. The *substitution function* for $(M; \Lambda_M)$ is a bijection $s_M : V(B) \to (V(B) \setminus S_B) \cup S_A$ defined as

$$s_M(b) = \begin{cases} a \text{ such that } \mu(a) = b & \text{if } b \in S_B \\ b & \text{if } b \in V(B) \setminus S_B \end{cases}$$

The vertex $s_M(b)$ is called the *substitute* of b in $(M; \Lambda_M)$. The *substitute set* of a subset $C \subseteq V(B)$, denoted $\sigma_M(C)$, is the set of substitutes of the vertices in C.

Fact 69. *Given LA graphs $(A; \Lambda_A)$ and $(B; \Lambda_B)$, the sets S_A and S_B, and μ, the merge graph $((A; \Lambda_A) \leftrightarrow (B; \Lambda_B), S_A, S_B, \mu)$ can be computed in time $\mathcal{O}(|V(A)|^2 + |V(B)|^2)$.*

Proof. We refer to Definition 67 for the definition of $(M; \Lambda_M) = ((A; \Lambda_A) \leftrightarrow (B; \Lambda_B), S_A, S_B, \mu)$.

$V(M)$ can be computed in time $\mathcal{O}(|V(A)| + |V(B)|)$. Each edge in E_{AB} corresponds to an edge of B, so $E(M)$ can be computed in time $\mathcal{O}(|E(A)| \cup |E(B)|)$, or, $\mathcal{O}(|V(A)|^2 + |V(B)|^2)$. Finally, Λ_M can be computed in time $\mathcal{O}(|V(A)| + |V(B)|)$, which completes the proof. □

Lemma 70. *The size of the family $(A; \Lambda_A) \leftrightarrow (B; \Lambda_B)$ is bounded by a function of $|V(A)| + |V(B)|$.*

Proof. We note that, for a given pair of subsets $S_A \subseteq V(A)$ and $S_B \subseteq V(B)$ such that $|S_A| = |S_B|$, there are $|S_A|!$ bijections possible. Also, for a given size of the subsets $i = |S_A| = |S_B|$, where $0 \leq i \leq \min\{|V(A)|, |V(B)|\}$, there are $\binom{|V(A)|}{i}$ and $\binom{|V(B)|}{i}$ choices for the subsets S_A and S_B, respectively. Therefore, the number of merge graphs with i as the size of the subsets is $i! \cdot \binom{|V(A)|}{i} \cdot \binom{|V(B)|}{i}$. Summing over all values of i, the total number of LA graphs obtained by merging is $\sum_{i=0}^{\min\{|V(A)|,|V(B)|\}} i! \cdot \binom{|V(A)|}{i} \cdot \binom{|V(B)|}{i}$, which is bounded by $\sum_{i=0}^{\min\{|V(A)|,|V(B)|\}} (|V(A)|! \cdot |V(B)|!)$, which in turn is bounded by $\mathcal{O}(n \cdot (n!)^2)$, where $n = \max\{|V(A)|, |V(B)|\}$. Now, $n! \leq n^n$ for all $n > 0$, and so the size of $(A; \Lambda_A) \leftrightarrow (B; \Lambda_B)$ is bounded, less tightly, by $\mathcal{O}(n^{2n+1})$. Since $n \leq |V(A)| + |V(B)|$, the size of $(A; \Lambda_A) \leftrightarrow (B; \Lambda_B)$ is bounded by a function of $|V(A)| + |V(B)|$. □

Lemma 71. *For LA graphs $(A; \Lambda_A)$ and $(B; \Lambda_B)$, if $(M; \Lambda_M) \in (A; \Lambda_A) \leftrightarrow (B; \Lambda_B)$, then A is an induced subgraph of M and B is isomorphic to $M[\sigma_M(V(B))]$ with respect to the substitution function s_M.*

Proof. Let $(M; \Lambda_M) = ((A; \Lambda_A) \leftrightarrow (B; \Lambda_B), S_A, S_B, \mu)$. We refer to Definition 67 throughout the proof.

We know $V(A) \subseteq V(M)$ and $E(A) \subseteq E(M)$, and so A is a subgraph of M. We claim A is also an induced subgraph of M. Note that $E(M) \setminus E(A) = E(B[V(B) \setminus S_B]) \cup E_{AB}$. Obviously, $E(B[V(B) \setminus S_B])$ contains edges both of whose end-vertices are in $V(B)$ and, by our definition of E_{AB}, edges in E_{AB} have exactly one end-vertex in $V(B)$. Therefore, there exists no edge $(a, a') \in E(M) \setminus E(A)$ such that $a, a' \in V(A)$, which implies A is also an induced subgraph of M.

To prove B is isomorphic to $M[\sigma_M(V(B))]$ with respect to s_M, we consider the three types of vertex-pairs $b, \widehat{b} \in V(B)$ and show that $(s_M(b), s_M(b')) \in E(M)$ if and only if $(b, \widehat{b}) \in E(B)$.

Case 1. $b, \widehat{b} \in V(B) \setminus S_B$.

Then, $(s_M(b), s_M(b')) = (b, b')$ and we know $(s_M(b), s_M(b')) = (b, b') \in E(M)$ if and only if $(b, b') \in E(B[V(B) \setminus S_B]) \subseteq E(B)$.

Case 2. $b \in S_B$ and $\widehat{b} \in V(B) \setminus S_B$.

Then, $(s_M(b), s_M(b')) = (a, b')$, where $\mu(a) = b$. Now, $(a, b') \in E_{AB}$ if and only if $(b, b') \in E(B)$, and we know $(a, b') \in E(M)$ if and only if $(a, b') \in E_{AB} \subseteq E(B)$.

Case 3. $b \in V(B) \setminus S_B$ and $\widehat{b} \in S_B$.

The argument is similar to that in Case (2) with b and $\widehat{b}$ interchanged.

Case 4. $b, \widehat{b} \in S_B$.

Then, $(s_M(b), s_M(b')) = (a, a')$, where $\mu(a) = b$ and $\mu(a') = b'$. Since $A[S_A]$ and $B[S_B]$ are isomorphic with respect to μ, $(a, a') \in E(A)$ if and only if $(b, b') \in E(B)$, and we know $(a, a') \in E(M)$ if and only if $(a, a') \in E(A)$.

□

Given two LA graphs that are list-isomorphic to subgraphs of a graph G, not every merge graph is list-isomorphic to a subgraph of G, but we will show that there exists at least one that is list-isomorphic to a subgraph of G (Lemma 73).

Definition 72. Let $(A; \Lambda_A)$ and $(B; \Lambda_B)$ be LA graphs, each of which is list-isomorphic to a subgraph of G. We call $(M; \Lambda_M) \in (A; \Lambda_A) \leftrightarrow (B; \Lambda_B)$ that is list-isomorphic to a subgraph of G a *G-realized merge graph.*

Lemma 73. *Given a graph G and LA graphs $(A; \Lambda_A)$ and $(B; \Lambda_B)$, each of which is list-isomorphic to a subgraph of G, $(A; \Lambda_A) \leftrightarrow (B; \Lambda_B)$ contains a G-realized merge graph.*

Proof. Suppose there exist subgraphs P and Q of G such that $(A; \Lambda_A) \simeq P$ with respect to a bijection $\pi : V(A) \to V(P)$ and $(B; \Lambda_B) \simeq Q$ with respect to a bijection $\chi : V(B) \to V(Q)$. Let D be the graph defined by $V(D) = V(P) \cup V(Q)$ and $E(D) = E(P) \cup E(Q)$. Obviously, D is a subgraph of G. We will construct a merge graph $(M; \Lambda_M)$ and show that it is list-isomorphic to D.

Let $(M; \Lambda_M) = ((A; \Lambda_A) \leftrightarrow (B; \Lambda_B), S_A, S_B, \mu)$, where $S_A = \{a \in V(A) \mid \pi(a) \in V(P) \cap V(Q)\}$, $S_B = \{b \in V(B) \mid \chi(b) \in V(P) \cap V(Q)\}$, and $\mu : S_A \to S_B$ is a bijection such that $\pi(a) = \chi(\mu(a))$ for each $a \in S_A$. Observe that the sets S_A and S_B are mapped to the same set $V(P) \cap V(Q)$. Furthermore, since A and B are isomorphic to P and Q respectively, $(a, b) \in E_{AB}$ if and only if $(\pi(a), \chi(b)) \in E(Q)$ for any $a \in S_A$ and $b \in V(B) \setminus S_B$ (Definition 67). Hence, it follows that M is isomorphic to D.

Furthermore, $\Lambda_M(m)$ equals $NDS(G, \pi(m))$ if $m \in V(A)$ and $NDS(G, \chi(m))$ if $m \in V(B)$ by Definition 67 and the fact that $(A; \Lambda_A)$ and $(B; \Lambda_B)$ are list-isomorphic to subgraphs of G. Since $V(D) = V(P) \cup V(Q) = \{\pi(m) \mid m \in V(A)\} \cup \{\chi(m) \mid m \in V(B)\}$, it is true that $(M; \Lambda_M) \simeq D$. □

7.2 Antecedent of a merge graph

Given an LA edit pair $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$ and a merge graph $(M; \Lambda_M)$ obtained by merging $(Y; \Lambda_Y)$ with an LA graph, the antecedent of $(M; \Lambda_M)$ is obtained by

performing the "reverse" NDL graph edit in $(M;\Lambda_M)$, i.e., by deleting (inserting) the vertices and edges that are inserted (respectively, deleted) in $\mathcal{E}$. However, we require that the inserted vertex, if any, is identified only with an isolated vertex when merging because, otherwise, performing the "reverse" NDL graph edit would delete this vertex but keep the edges incident with it in the merge graph intact, resulting in "hanging edges".

We will show in Lemma 79 that checking whether $(M;\Lambda_M)$ is list-isomorphic to a subgraph of a $(G,\mathcal{E})$-produced graph is equivalent to checking whether its antecedent is list-isomorphic to a subgraph of G.

Definition 74. Let $\mathcal{E} = ((X;\Lambda_X),(Y;\Lambda_Y))$ be an LA edit pair and $(A;\Lambda_A)$ be an LA graph. Let $(M;\Lambda_M) = ((Y;\Lambda_Y) \leftrightarrow (A;\Lambda_A), S_Y, S_A, \mu)$ be such that for $y \in V_+(\mathcal{E}) \cap S_Y$, if any, $\mu(y)$ is an isolated vertex. The *antecedent* of $(M;\Lambda_M)$, denoted $(\alpha(M);\Lambda_{\alpha(M)})$, is the LA graph given by

$$\begin{aligned} V(\alpha(M)) &= \Big(V(M) \setminus V_+(\mathcal{E})\Big) \cup V_-(\mathcal{E}) \\ E(\alpha(M)) &= \Big(E(M) \setminus E_+(\mathcal{E})\Big) \cup E_-(\mathcal{E}) \\ \Lambda_{\alpha(M)}(v) &= \begin{cases} \Lambda_X(v) & \text{if } v \in V(\alpha(M)) \cap V(X) \\ \Lambda_M(v) & \text{otherwise} \end{cases} \end{aligned}$$

We use the fact that $V_+(\mathcal{E})$, $V_-(\mathcal{E})$, and $E_+(\mathcal{E})$ have at most one element each and $E_-(\mathcal{E})$ has at most $\Delta(X) \leq |V(X)|$ edges (Definition 14) to bound the time for computing the antecedent.

Fact 75. *Given an LA edit pair $\mathcal{E} = ((X;\Lambda_X),(Y;\Lambda_Y))$ and a merge graph $(M;\Lambda_M) \in (Y;\Lambda_Y) \leftrightarrow (A;\Lambda_A)$, $(\alpha(M);\Lambda_{\alpha(M)})$ can be computed in time $\mathcal{O}(|V(X)|)$.*

Fact 76. *Let $\mathcal{E} = ((X;\Lambda_X),(Y;\Lambda_Y))$ be an LA edit pair and $(A;\Lambda_A)$ be an LA graph. For a merge graph $(M;\Lambda_M) \in (Y;\Lambda_Y) \leftrightarrow (A;\Lambda_A)$, $|V(\alpha(M)| \leq |V(M)| + 1$.*

Proof. By Definition 74, $V(\alpha(M)) \setminus V(M) = V_-(\mathcal{E})$. We know $|V_-(\mathcal{E})| = |V(X) \setminus V(Y)| \leq 1$ since at most one vertex is deleted (Definitions 12 and 14), which proves our claim. □

Fact 77. *Let $\mathcal{E} = ((X;\Lambda_X),(Y;\Lambda_Y))$ be an LA edit pair, $G' = ndledit(\Phi)$, where $\Phi = (G,\mathcal{E},H,f)$, and $(A;\Lambda_A)$ be an LA graph. For any $(M;\Lambda_M) \in (Y;\Lambda_Y) \leftrightarrow (A;\Lambda_A)$, the following hold:*

1. *a vertex $v \in V(\alpha(M)) \setminus V(M)$ if and only if $f(v) \in V(G) \setminus V(G')$,*

2. *there exists a bijection from* $V(M) \setminus V(\alpha(M))$ *to* $V(G') \setminus V(G)$,

3. *an edge* $(u, v) \in E(\alpha(M)) \setminus E(M)$ *if and only if* $(f(u), f(v)) \in E(G) \setminus E(G')$, *and*

4. *an edge* $(u, v) \in E(M) \setminus E(\alpha(M))$ *if and only if* $(f(u), f(v)) \in E(G') \setminus E(G)$.

Proof. Suppose $G' = ndledit(\Phi)$, where $\Phi = (G, \mathcal{E}, H, f)$.

1. Note that $V(\alpha(M)) \setminus V(M) = V_-(\mathcal{E})$ (Definition 74). Now, we know f gives a bijection from $V_-(\mathcal{E})$ to $V_-(\Phi)$ (Definition 17), and since $V(G) \setminus V(G') = V_-(\Phi)$ (Definition 18), f gives a bijection from $V(\alpha(M)) \setminus V(M)$ to $V(G) \setminus V(G')$.

2. Note that $V(M) \setminus V(\alpha(M)) = V_+(\mathcal{E})$ and $E(M) \setminus E(\alpha(M)) = E_+(\mathcal{E})$ (Definition 74). By Definition 17, there exists a bijection from $V_+(\mathcal{E})$ to $V_+(\Phi)$, and so from $V(M) \setminus V(\alpha(M))$ to $V(G') \setminus V(G)$.

3. Note that $E(\alpha(M)) \setminus E(M) = E_-(\mathcal{E})$ (Definition 74) and $(u, v) \in E_-(\mathcal{E})$ if and only if $(f(u), f(v)) \in E_-(\Phi)$ (Definition 17). Now, since $E(G) \setminus E(G') = E_-(\Phi)$ (Definition 18), our claim holds.

4. Note that $E(M) \setminus E(\alpha(M)) = E_+(\mathcal{E})$ (Definition 74) and $(u, v) \in E_+(\mathcal{E})$ if and only if $(f(u), f(v)) \in E_+(\Phi)$ (Definition 17). Since $E_+(\Phi) = E(G') \setminus E(G)$ (Definition 18), our claim holds.

□

Lemma 78. *Let* $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$ *be an LA edit pair and* $(A; \Lambda_A)$ *an LA graph. For any* $(M; \Lambda_M) \in (Y; \Lambda_Y) \leftrightarrow (A; \Lambda_A)$, X *is an induced subgraph of* $\alpha(M)$.

Proof. We know by Lemma 71 that Y is an induced subgraph of M, which means $V(Y) \subseteq V(M)$ and $E(Y) \subseteq E(M)$. Therefore, from Definitions 14 and 74, it follows that $V(X) \subseteq V(\alpha(M))$ and $E(X) \subseteq E(\alpha(M))$, and hence X is a subgraph of $\alpha(M)$.

We show X is also an induced subgraph of $\alpha(M)$ by referring to Definition 74 in the rest of the proof. Given a pair of vertices $u, v \in V(X)$, there are two cases:

Case 1. $(u, v) \in E(X)$.

Case i. $(u, v) \in E(Y)$.
Then, $(u, v) \in E(X) \cap E(Y)$, and so $(u, v) \notin E_+(\mathcal{E})$ (Definition 14). Now, $(u, v) \in E(M)$ because $E(Y) \subseteq E(M)$. Hence, $(u, v) \in E(\alpha(M))$.

Case ii. $(u, v) \notin E(Y)$.

In this case, $(u, v) \in E(X) \setminus E(Y) = E_{-}(\mathcal{E})$, and so $(u, v) \in E(\alpha(M))$.

Case 2. $(u, v) \notin E(X)$.

Case i. $(u, v) \in E(Y)$.

Then, $(u, v) \in E(Y) \setminus E(X) = E_{+}(\mathcal{E})$, and so $(u, v) \notin E(\alpha(M))$.

Case ii. $(u, v) \notin E(Y)$.

Since Y is an induced subgraph of M (Lemma 71), $(u, v) \notin E(M)$. Moreover, $(u, v) \notin E(X) \cup E(Y)$ implies $(u, v) \notin E_{-}(\mathcal{E})$ (Definition 14). Hence, $(u, v) \notin E(\alpha(M))$.

□

Lemma 79. *Suppose G' is a $(G, \mathcal{E})$-produced graph, where $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$ is complete in G. Then, an LA graph $(A; \Lambda_A)$ is list-isomorphic to a subgraph of G' if and only if, for any G'-realized merge graph $(M; \Lambda_M) \in (Y; \Lambda_Y) \leftrightarrow (A; \Lambda_A)$, $(\alpha(M); \Lambda_{\alpha(M)})$ is list-isomorphic to a subgraph of G.*

Proof. Let $G' = ndledit(\Phi)$, where $\Phi = (G, \mathcal{E}, H, f)$. Since $\mathcal{E}$ fits G, we know $(X; \Lambda_X) \simeq H$ with respect to f (Definition 16), and since $\mathcal{E}$ is complete in G, we know $(Y; \Lambda_Y)$ is list-isomorphic to an induced subgraph of G' (Fact 29).

Suppose $(A; \Lambda_A)$ is list-isomorphic to a subgraph of G' and $(M; \Lambda_M) \in (Y; \Lambda_Y) \leftrightarrow (A; \Lambda_A)$ is a G'-realized merge graph. We know by Lemma 78 that X is an induced subgraph of $\alpha(M)$.

First, we show $\alpha(M)$ is isomorphic to a subgraph of G. We know that there exist bijections from $V(\alpha(M)) \setminus V(M)$ to $V(G) \setminus V(G')$ and from $E(\alpha(M)) \setminus E(M)$ to $E(G) \setminus E(G')$ (Fact 77). By Fact 77 again, there exist bijections from $V(M) \setminus V(\alpha(M))$ to $V(G') \setminus V(G)$ and from $E(M) \setminus E(\alpha(M))$ to $E(G') \setminus E(G)$. Therefore, since M is isomorphic to a subgraph of G', each vertex (edge) of $\alpha(M)$ corresponds to a vertex (respectively, edge) of G, which implies $\alpha(M)$ is isomorphic to a subgraph of G.

Moreover, we know $NDS(G, u) = NDS(G', u)$ for each $u \in V(G) \setminus V(H)$ (Fact 28). Now, since $(M; \Lambda_M)$ is list-isomorphic to a subgraph of G', $\Lambda_M(m)$ equals the NDS of the vertex corresponding to m in G' for each $m \in V(M)$. This implies $\Lambda_{\alpha(M)}(v)$ gives the NDS of the vertex corresponding to v in G for each $v \in V(\alpha(M)) \setminus V(X)$ because $(X; \Lambda_X) \simeq H$. Furthermore, for each $x \in V(X)$, we know $\Lambda_X(x) = NDS(G, f(x))$ because $(X; \Lambda_X) \simeq H$ with respect to f. Hence, $(\alpha(M), \Lambda_{\alpha(M)})$ is also list-isomorphic to a subgraph of G.

To prove the other direction, consider a G'-realized merge graph $(M;\Lambda_M) \in (Y;\Lambda_Y) \leftrightarrow (A;\Lambda_A)$ such that $(\alpha(M);\Lambda_{\alpha(M)})$ is list-isomorphic to a subgraph of G. Since there exist bijections from $V(M) \setminus V(\alpha(M))$ to $V(G') \setminus V(G)$, from $E(M) \setminus E(\alpha(M))$ to $E(G') \setminus E(G)$, from $V(\alpha(M)) \setminus V(M)$ to $V(G) \setminus V(G')$, and from $E(\alpha(M)) \setminus E(M)$ to $E(G) \setminus E(G')$, each vertex (edge) of M corresponds to a vertex (respectively, edge) of G'. Hence, M is isomorphic to a subgraph of G'.

We show that $(M;\Lambda_M)$ is also list-isomorphic to a subgraph of G'. For each $v \in V(M) \setminus V(Y)$, $v \notin V(X)$ because otherwise $v \in V_-(\mathcal{E})$ (Definition 14), which would imply $v \notin V(M)$ (Definition 74). By Fact 28, $NDS(G,u) = NDS(G',u)$ for each $u \in V(G) \setminus V(H)$ and we know $(X;\Lambda_X) \simeq H$. Therefore, $\Lambda_M(v)$ gives the NDS of the vertex corresponding to v in G' for each $v \in V(M) \setminus V(Y)$. Furthermore, since $\mathcal{E}$ is complete in G, $\Lambda_Y(y)$ gives the NDS of the vertex corresponding to y in G' for each $y \in V_+(\mathcal{E}) \cup V_R(\mathcal{E})$. By Definition 14, $V(Y) = V_+(\mathcal{E}) \cup V_R(\mathcal{E})$, which implies $\Lambda_M(v)$ gives the NDS of the vertex corresponding to v for each $v \in V(M)$, which proves our claim. □

We wish to reduce determining whether $(\mathcal{X}[i];\Lambda_{\mathcal{X}[i]})$ is list-isomorphic to an induced subgraph of G_i to determining whether a certain LA graph is list-isomorphic to an induced subgraph of G_{i-1}. For this, we use Lemma 80, which states that, given a $(G,\mathcal{E})$-produced graph G' and a subgraph $(Z;\Lambda_Z)$ of a G'-realized merge graph $(M;\Lambda_M)$, checking whether $(Z;\Lambda_Z)$ is list-isomorphic to an induced subgraph of G' can be reduced to checking whether the subgraph induced by the substitute set of $V(Z)$ in $(\alpha(M);\Lambda_{\alpha(M)})$ is list-isomorphic to an induced subgraph of G.

Lemma 80. *Suppose the following:*

1. *a $(G,\mathcal{E})$-produced graph G', where $\mathcal{E} = ((X;\Lambda_X),(Y;\Lambda_Y))$,*
2. *an LA graph $(A;\Lambda_A)$ that is list-isomorphic to a subgraph of G',*
3. *a G'-realized merge graph $(M;\Lambda_M) \in (Y;\Lambda_Y) \leftrightarrow (A;\Lambda_A)$, and*
4. *an LA graph $(Z;\Lambda_Z)$ such that Z is an induced subgraph of A and $\Lambda_Z(z) = \Lambda_A(z)$ for each $z \in V(Z)$.*

Then, $(Z;\Lambda_Z)$ is list-isomorphic to an induced subgraph of G' if and only if $(R;\Lambda_R)$ is list-isomorphic to an induced subgraph of G, where R is the subgraph induced by the substitute set of $V(Z)$ in $\alpha(M)$, or, $R = \alpha(M)[\sigma_M(V(Z)) \cap V(\alpha(M))]$, and $\Lambda_R(r) = \Lambda_{\alpha(M)}(r)$ for each $r \in V(R)$.

Proof. Suppose $(Z;\Lambda_Z)$ is list-isomorphic to an induced subgraph of G'. Since $(A;\Lambda_A)$ is list-isomorphic to a subgraph of G', we know that $(\alpha(M);\Lambda_{\alpha(M)})$ is list-isomorphic to a subgraph of G (Lemma 79). Furthermore, because $R = \alpha(M)[\sigma_M(V(Z)) \cap V(\alpha(M))]$ is an induced subgraph of $\alpha(M)$ and Λ_R retains the nonincreasing lists given by $\Lambda_{\alpha(M)}$, $(R;\Lambda_R)$ is list-isomorphic to a subgraph of G with respect to a bijection, say, g.

We show this subgraph of G is induced. Suppose $(u,v) \notin E(R)$ for some $u, v \in V(R)$, which means $(u,v) \notin E(\alpha(M))$ and so $(u,v) \notin E_-(\mathcal{E})$. By Definition 74, then, there are two cases:

Case 1. $(u,v) \in E(M)$.

This implies $(u,v) \in E_+(\mathcal{E})$. Then, $(g(u),g(v)) \in E_+(\Phi)$ and so $(g(u),g(v)) \notin E(G)$ (Definitions 17 and 18).

Case 2. $(u,v) \notin E(M)$.

We first note that, since $u, v \in \sigma_M(V(Z))$, there exist $w, z \in V(Z)$ such that $u = s_M(w)$ and $v = s_M(z)$ (Definition 68). Since Z is isomorphic to an induced subgraph of G', $(g(u),g(v)) \notin E(G')$. Now, because $(u,v) \notin E_-(\mathcal{E})$, $(g(u),g(v)) \notin E_-(\Phi)$ and hence $(g(u),g(v)) \notin E(G)$ (Definitions 17 and 18).

Therefore, $(R;\Lambda_R)$ is list-isomorphic to an induced subgraph of G.

To prove the other direction, suppose $(R;\Lambda_R)$ is list-isomorphic to an induced subgraph of G. Let $(M;\Lambda_M)$ be list-isomorphic to a subgraph of G' with respect to a bijection, say, h. Therefore, $(\widehat{M};\Lambda_{\widehat{M}})$ is list-isomorphic to a subgraph of G', where $\widehat{M} = M[\sigma_M(V(Z))]$ and $\Lambda_{\widehat{M}}(m) = \Lambda_M(m)$ for each $m \in V(\widehat{M})$.

We show this subgraph of G' is induced. Suppose $(u,v) \notin E(\widehat{M})$ for some $u, v \in V(\widehat{M})$, which means $(u,v) \notin E(M)$. We claim $(h(u),h(v)) \notin E(G')$.

Since Y is an induced subgraph of M by Lemma 71, if $u, v \in V(Y)$, then $(u,v) \notin E(Y)$, which implies $(u,v) \notin E_+(\mathcal{E})$ (Definition 14). By Definition 74, then, there are two cases:

Case 1. $(u,v) \in E(\alpha(M))$.

Since $(u,v) \notin E(M)$, this implies $(u,v) \in E_-(\mathcal{E})$ (Definition 74). Then, $(h(u),h(v)) \in E_-(\Phi)$ and so $(h(u),h(v)) \notin E(G')$ (Definitions 17 and 18).

Case 2. $(u,v) \notin E(\alpha(M))$.

This implies $(u,v) \notin E(P)$. Since P is isomorphic to an induced subgraph of G, $(h(u),h(v)) \notin E(G)$. Now, since $(u,v) \notin E_+(\mathcal{E})$, it follows from Definition 17 that $(h(u),h(v)) \notin E_+(\Phi)$, and so $(h(u),h(v)) \notin E(G')$ (Definition 18).

Hence, $(\widehat{M}; \Lambda_{\widehat{M}})$ is list-isomorphic to an induced subgraph of G'. Notice that Z is isomorphic to $\widehat{M}$ by Lemma 71 and $\Lambda_Z(z) = \Lambda_A(z)$ for each $z \in V(Z)$ by our definition of Z. By Definition 67, $\Lambda_A(a) = \Lambda_M(s_M(a))$ for each $a \in V(A)$. Therefore, $\Lambda_Z(z) = \Lambda_M(s_M(z))$ for each $z \in V(Z)$, and so $(Z; \Lambda_Z)$ is list-isomorphic to an induced subgraph of G'. □

Given a $(G, \mathcal{E})$-produced graph G', we reduce a condition on the degree of a vertex in G' to a condition on the degree of a vertex in G in Lemma 81.

Lemma 81. *Let $G' = ndledit(G, \mathcal{E}, H, f)$, where $\mathcal{E} = ((X; \Lambda_X), (Y; \Lambda_Y))$, and $(A; \Lambda_A)$ be an LA graph. Also, let $(M; \Lambda_M) \in (Y; \Lambda_Y) \leftrightarrow (A; \Lambda_A)$ be list-isomorphic to a subgraph of G' with respect to a bijection g. Then, $d_M(v) = d_{G'}(g(v))$ if and only if $d_{\alpha(M)}(v) = d_G(g(v))$ for each $v \in V(M) \cap V(\alpha(M))$.*

Proof. Consider a vertex $v \in V(M) \cap V(\alpha(M))$. We know that $(Y; \Lambda_Y)$ is list-isomorphic to an induced subgraph of G' such that each $y \in V_R(\mathcal{E})$ is mapped to $f(y)$ in G' (Fact 29). Hence, if $v \in V(Y) \cap V(X) = V_R(\mathcal{E})$, then we know $g(v) = f(v)$.

We consider two cases for v. If $v \in V_R(\mathcal{E})$, then we show the inserted and deleted edges incident with $g(v)$ correspond to the edges incident with v in $E(M) \setminus E(\alpha(M))$ and $E(\alpha(M)) \setminus E(M)$, respectively. If $v \notin V_R(\mathcal{E})$, then we show the degrees of v in M and $\alpha(M)$ are equal, and so are the degrees of $g(v)$ in G' and G.

Suppose $v \in V_R(\mathcal{E})$, which means $g(v) = f(v)$. Now, by Fact 77, there exists $u \in V(M)$ such that $(v, u) \in E(M) \setminus E(\alpha(M))$ if and only if $(f(v), f(u)) \in E(G') \setminus E(G)$. Furthermore, there exists $w \in V(M)$ such that $(v, w) \in E(\alpha(M)) \setminus E(M)$ if and only if $(f(v), f(w)) \in E(G) \setminus E(G')$. Hence, $d_M(v) = d_{G'}(f(v))$ if and only if $d_{\alpha(M)}(v) = d_G(f(v))$. Since $g(v) = f(v)$, we obtain $d_M(v) = d_{G'}(g(v))$ if and only if $d_{\alpha(M)}(v) = d_G(g(v))$.

Suppose $v \notin V_R(\mathcal{E})$. We know $v \notin V_-(\mathcal{E})$ and $v \notin V_+(\mathcal{E})$ because $v \in V(M) \cap V(\alpha(M))$ (Definition 74). Therefore, $v \notin V(X) \cup V(Y)$ by Definition 14. This implies the neighbourhoods of v in M and $\alpha(M)$ are equal. Therefore, $d_M(v) = d_{\alpha(M)}(v)$ and $d_{G'}(g(v)) = d_G(g(v))$, which means $d_M(v) = d_{G'}(g(v))$ if and only if $d_{\alpha(M)}(v) = d_G(g(v))$. □

7.3 Origin graph

The origin graph is a graph on the "origins" of the vertices in an LA edit pair sequence. Given an LA edit pair sequence $\langle \mathcal{E}_i \rangle_0^t$, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$, the "replacements" of subgraphs as specified by the LA edit pair sequence are "undone"

to trace the orgins of the vertices of $\mathcal{X}[i]$ for $i > 0$. This sequence of "undo" procedures is formalized by the antecedent and merge sequences.

7.3.1 Antecedent and merge sequences

The ith merge graph is obtained by merging the ith antecedent with $(\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]})$, and the $(i-1)$th antecedent is the antecedent of the ith merge graph.

Definition 82. Let $\langle \mathcal{E}_i \rangle_0^t$ be an LA edit pair sequence, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$. For $1 \leq i \leq t$, $\mathcal{A}_i = \langle (A[j]; \Lambda_{A[j]}) \rangle_0^i$ is an *antecedent sequence* and $\mathcal{M}_i = \langle (M[j]; \Lambda_{M[j]}) \rangle_1^i$ is a *merge sequence* if and only if

1. $(A[i]; \Lambda_{A[i]}) = (\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]})$, and
2. $(M[j]; \Lambda_{M[j]}) \in (\mathcal{Y}[j-1]; \Lambda_{\mathcal{Y}[j-1]}) \leftrightarrow (A[j]; \Lambda_{A[j]})$ and $(A[j-1]; \Lambda_{A[j-1]}) = \alpha(M[j])$ for each $1 \leq j \leq i$.

Definition 83. Let $\langle \mathcal{E}_i \rangle_0^t$ be an LA edit pair sequence, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$. For $1 \leq i \leq t$, we say an antecedent sequence $\mathcal{A}_i = \langle (A[j]; \Lambda_{A[j]}) \rangle_0^i$ and a merge sequence $\mathcal{M}_i = \langle (M[j]; \Lambda_{M[j]}) \rangle_1^i$ are *good* with respect to a graph sequence $\langle G_k \rangle_0^n$ if and only if

1. $(A[j]; \Lambda_{A[j]})$ and $(M[j]; \Lambda_{M[j]})$ are list-isomorphic to subgraphs of G_j for each $1 \leq j \leq i$, and
2. $(A[0]; \Lambda_{A[0]})$ is list-isomorphic to a subgraph of G_0.

We bound the sizes of the graphs in the antecedent and merge sequences in Fact 84 using the bound on the size of the antecedent (Fact 76). Next, we bound the number of choices for $\mathcal{A}_i$ and $\mathcal{M}_i$ in Fact 85 using the bound on the size of the family of merge graphs (Lemma 70).

Fact 84. *Let $\langle \mathcal{E}_i \rangle_0^t$ be an LA edit pair sequence, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$. For some $1 \leq i \leq t$, let $\mathcal{A}_i = \langle (A[j]; \Lambda_{A[j]}) \rangle_0^i$ be an antecedent sequence and $\mathcal{M}_i = \langle (M[j]; \Lambda_{M[j]}) \rangle_1^i$ be a merge sequence. Then, for each $1 \leq j \leq i$, $|V(M[j])| \leq k_i^j$ and $|V(A[j-1])| \leq k_i^j + 1$, where $k_i^j = |V(\mathcal{X}[i])| + \sum_{p=j-1}^{i-1} |V(\mathcal{Y}[p])| + i - j$.*

Proof. Consider an arbitrary $1 \leq i \leq t$. For each $1 \leq j \leq i$, we show that $|V(M[j])| \leq k_i^j$ and $|V(A[j-1])| \leq k_i^j + 1$ by induction on j.

By Definition 67, $|V(M[i])| \leq |V(\mathcal{X}[i])| + |V(\mathcal{Y}[i-1])| = k_i^i$ and by Fact 76, $|V(A[i-1])| \leq |V(\mathcal{X}[i])| + |V(\mathcal{Y}[i-1])| + 1 = k_i^i + 1$, and so the claim holds for i.

Suppose the claim holds for some $1 < j < i$, i.e., $|V(M[j])| \leq k_i^j$ and $|V(A[j-1])| \leq k_i^j + 1$. Then, by Definitions 67 and 82, we know

$$\begin{aligned} |V(M[j-1])| &\leq |V(A[j-1])| + |V(\mathcal{Y}[j-2])| \\ &\leq k_i^j + 1 + |V(\mathcal{Y}[j-2])| \\ &= k_i^{j-1} \end{aligned}$$

By Fact 76, $|V(A[j-2])| \leq k_i^{j-1} + 1$. So the claim holds for $j-1$. □

(Disclaimer: Fact 85 through Theorem 97 have not been verified for correctness by the supervisor.)

Fact 85. *Let $\langle \mathcal{E}_i \rangle_0^t$ be an LA edit pair sequence, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$. For $1 \leq i \leq t$, the number of choices for $\mathcal{A}_i$ and $\mathcal{M}_i$ is bounded by a function of $i + k_i^1$, where $k_i^1 = |V(\mathcal{X}[i])| + \sum_{p=0}^{i-1} |V(\mathcal{Y}[p])| + i - 1$.*

Proof. Consider an arbitrary $1 \leq i \leq t$. Let $k_i^j = |V(\mathcal{X}[i])| + \sum_{p=j-1}^{i-1} |V(\mathcal{Y}[p])| + i - j$ for each $1 \leq j \leq i$ as stated in Fact 84. We show that the number of choices for $(M[j]; \Lambda_{M[j]})$ is bounded by a function of $\prod_{p=j}^{i} k_i^p$ for each $1 \leq j \leq i$ by induction on j. Since we compute $\mathcal{M}_i$ "backwards", or, compute $(M[j]; \Lambda_{M[j]})$ from $(M[j+1]; \Lambda_{M[j+1]})$, the number of choices for $(M[1]; \Lambda_{M[1]})$ also gives the number of choices for $\mathcal{M}_i$. On the other hand, by Definition 82, $(A[j-1]; \Lambda_{A[j-1]})$ is determined by $(M[j]; \Lambda_{M[j]})$ for each $1 \leq j \leq i$, and so the number of choices for $\mathcal{A}_i$ equals that for $\mathcal{M}_i$.

By Lemma 70, the number of choices for $(M[i]; \Lambda_{M[i]})$ is bounded by a function of $|V(\mathcal{X}[i])| + |V(\mathcal{Y}[i-1])| = k_i^i < k_i^1$, and so our claim holds for i.

Suppose the claim holds for some $1 < j < i$. By Definition 82 and Lemma 70, for a fixed $(A[j-1]; \Lambda_{A[j-1]})$, the number of choices for $(M[j-1]; \Lambda_{M[j-1]})$ is bounded by a function of $|V(A[j-1])| + |V(\mathcal{Y}[j-2])|$, i.e., by Fact 84, a function of $k_i^j + 1 + |V(\mathcal{Y}[j-2])| = k_i^{j-1}$. Now, since the number of choices for $(A[j-1]; \Lambda_{A[j-1]})$ is bounded by a function of $\prod_{p=j}^{i} k_i^p$, the number of choices for $(M[j-1]; \Lambda_{M[j-1]})$ is bounded by a function of $\left(\prod_{p=j}^{i} k_i^p\right) \cdot k_i^{j-1} = \prod_{p=j-1}^{i} k_i^p$. Therefore, the number of choices for $(M[1]; \Lambda_{M[1]})$, and hence for $\mathcal{A}_i$ and $\mathcal{M}_i$, is bounded by a function of $\prod_{p=1}^{i} k_i^p$. Since $k_i^p \leq k_i^1$ for each $1 \leq p \leq i$, $\prod_{p=1}^{i} k_i^p \leq (k_i^1)^i$, which is a function of $i + k_i^1$.

Thus, the number of choices for $\mathcal{A}_i$ and $\mathcal{M}_i$ is bounded by a function of $i + k_i^1$. □

Fact 86. *Let $\langle \mathcal{E}_i \rangle_0^t$ be an LA edit pair sequence, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$, which is complete in a graph G_0. For $1 \leq i \leq t$, $k_i^1 = |V(\mathcal{X}[i])| + \sum_{p=0}^{i-1} |V(\mathcal{Y}[p])| + i - 1$ is bounded by a function of $\Delta(G_0) + t$.*

Proof. We recall that $\langle \mathcal{E}_i \rangle_0^t$ is complete in G_0 only if $\langle \mathcal{E}_i \rangle_0^t$ fits G_0 (Definition 40), which in turn holds only if $|V(\mathcal{X}[i])|$ and $|V(\mathcal{Y}[i])|$ are at most $(\Delta(G_0) + t + 1)^2 + 1$ for each $0 \leq i \leq t$ (Fact 43). Hence, we assume $|V(\mathcal{X}[i])|$ and $|V(\mathcal{Y}[i])|$ are bounded by $(\Delta(G_0)+t+1)^2+1$, which is in $\mathcal{O}((\Delta(G_0)+t)^2)$. Therefore, k_i^1 is in $\mathcal{O}(i \cdot (\Delta(G_0)+t)^2)$, or, $\mathcal{O}(t \cdot (\Delta(G_0) + t)^2)$, since $i \leq t$. In other words, k_i^1 is bounded by a function of $\Delta(G_0) + t$. □

Fact 87. *Let $\langle \mathcal{E}_i \rangle_0^t$ be an LA edit pair sequence, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$. For some $1 \leq i \leq t$, if $\mathcal{A}_i = \langle (A[j]; \Lambda_{A[j]}) \rangle_0^i$ is an antecedent sequence, then, $\mathcal{X}[j]$ is an induced subgraph of $A[j]$ for each $0 \leq j \leq i$.*

Proof. Let $\mathcal{A}_i = \langle (A[j]; \Lambda_{A[j]}) \rangle_0^i$ be an antecedent sequence. By Definition 82, $(A[i]; \Lambda_{A[i]}) = (\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]})$, and so $\mathcal{X}[i] = \mathcal{K}[i]$.

For each $0 \leq j < i$, it follows from the fact that $(A[j]; \Lambda_{A[j]}) = \alpha(M[j+1])$ and $(\mathcal{Y}[j]; \Lambda_{\mathcal{Y}[j]})$ is an induced subgraph of $(M[j+1]; \Lambda_{M[j+1]})$ (Definition 82) that $\mathcal{X}[j]$ is an induced subgraph of $A[j]$ (Lemma 78). □

We now reduce the condition for goodness of a pair of antecedent and merge sequences with respect to a graph sequence to a list-isomorphism condition on G_0, the first graph in the sequence.

Fact 88. *Let G_0 be a graph and $\langle \mathcal{E}_i \rangle_0^t$ be an LA edit pair sequence. For some $1 \leq i \leq t$, suppose $\langle G_j \rangle_0^i$ is a sequence of graphs such that $\langle \mathcal{E}_j \rangle_0^{i-1}$ is complete in G_0 with respect to $\langle G_j \rangle_0^i$. Then, for any $\mathcal{A}_i = \langle (A[j]; \Lambda_{A[j]}) \rangle_0^i$ and $\mathcal{M}_i = \langle (M[j]; \Lambda_{M[j]}) \rangle_1^i$, $\mathcal{A}_i$ and $\mathcal{M}_i$ are good with respect to $\langle G_j \rangle_0^i$ if and only if $(A[0]; \Lambda_{A[0]})$ is list-isomorphic to a subgraph of G_0.*

Proof. Let $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$. For an arbitrary $1 \leq i \leq t$, suppose $\langle \mathcal{E}_j \rangle_0^{i-1}$ is complete in G_0. Then, we know $\mathcal{E}_{j-1}$ is complete in G_{j-1} for each $1 \leq j \leq i$ (Definition 40), G_j is a $(G_{j-1}, \mathcal{E}_{j-1})$-produced graph (Definition 39), and $(\mathcal{Y}[j-1]; \Lambda_{\mathcal{Y}[j-1]})$ is list-isomorphic to an induced subgraph of G_j (Fact 29). We show that $(A[j]; \Lambda_{A[j]})$ and $(M[j]; \Lambda_{M[j]})$ are list-isomorphic to subgraphs of G_j if and only if $(A[j-1]; \Lambda_{A[j-1]})$ is list-isomorphic to a subgraph of G_{j-1} for each $1 \leq j \leq i$. This implies the fact that $(A[j]; \Lambda_{A[j]})$ and $(M[j]; \Lambda_{M[j]})$ are list-isomorphic to subgraphs of

G_j for each $1 \leq j \leq i$ if and only if $(A[0]; \Lambda_{A[0]})$ is list-isomorphic to a subgraph of G_0. Our result, then, follows from Definition 83.

Suppose $(A[j]; \Lambda_{A[j]})$ and $(M[j]; \Lambda_{M[j]})$ are list-isomorphic to subgraphs of G_j. By Lemma 79, then, $\alpha(M[j]) = (A[j-1]; \Lambda_{A[j-1]})$ is list-isomorphic to a subgraph of G_{j-1} since G_j is a $(G_{j-1}, \mathcal{E}_{j-1})$-produced graph.

On the other hand, suppose $(A[j-1]; \Lambda_{A[j-1]})$ is list-isomorphic to a subgraph of G_{j-1}. We know $(A[j-1]; \Lambda_{A[j-1]}) = \alpha(M[j])$ by Definition 82. Comparing Definitions 18 and 74, we obtain the fact that $(M[j]; \Lambda_{M[j]})$ is list-isomorphic to a subgraph of G_j because the modifications to G_{j-1} that produce G_j are "undone" in $M[j]$ to obtain $\alpha(M[j])$. Furthermore, since $\mathcal{E}_{j-1}$ is complete in G_{j-1}, $(A[j]; \Lambda_{A[j]})$ is list-isomorphic to a subgraph of G_j (Lemma 79). □

7.3.2 Origin graph

Given an LA edit pair sequence $\langle \mathcal{E}_i \rangle_0^t$, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$, let $(M[i-1]; \Lambda_{M[i-1]})$ belong to a merge sequence $\mathcal{M}_i$ for some $1 \leq i \leq t$. By Definition 68, we know $\sigma_{M[i]}(V(\mathcal{X}[i]))$ contains the substitutes of the vertices of $\mathcal{X}[i]$ in $M[i]$, and the substitutes that are also present in the antecedent of $(M[i]; \Lambda_{M[i]})$ "substitute" for the vertices of $\mathcal{X}[i]$ in the antecedent. Now, we know this antecedent is $A[i-1]$, and we call such a vertex that "substitutes" for a vertex of $\mathcal{X}[i]$ its proxy in $A[i-1]$. Continuing similarly, we obtain the proxies of the vertices of $\mathcal{X}[i]$ in $A[0]$, the set of which we call the primary proxy set of $\mathcal{X}[i]$.

The origin of a vertex of $\mathcal{X}[i]$ is its "earliest" proxy. The origin graph is the subgraph of $A[0]$ induced by the origins present in $A[0]$.

Definition 89. Let $\langle \mathcal{E}_i \rangle_0^t$ be an LA edit pair sequence, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$. For a given $1 \leq i \leq t$, let $\mathcal{A}_i = \langle (A[j]; \Lambda_{A[j]}) \rangle_0^i$ be an antecedent sequence and $\mathcal{M}_i = \langle (M[j]; \Lambda_{M[j]}) \rangle_1^i$ be a merge sequence. We define a sequence of vertex-sets $\mathcal{P}_i$ as follows:

1. $\mathcal{P}_i[i] = V(\mathcal{X}[i])$ and
2. $\mathcal{P}_i[j] = \sigma_{M[j+1]}(\mathcal{P}_i[j+1]) \cap V(A[j])$ for each $0 \leq j < i$.

The origin of a vertex $x_i \in \mathcal{X}[i]$ is its proxy in $\mathcal{P}_i[j]$, where $\mathcal{P}_i[j]$ is the first set in the sequence to contain a proxy of x_i.

Definition 90. Let $\langle \mathcal{E}_i \rangle_0^t$ be an LA edit pair sequence, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$. For a given $1 \leq i \leq t$, let $\mathcal{A}_i = \langle (A[j]; \Lambda_{A[j]}) \rangle_0^i$ be an antecedent sequence and $\mathcal{M}_i = \langle (M[j]; \Lambda_{M[j]}) \rangle_1^i$ be a merge sequence. The *proxy*

of a vertex $x_i \in V(\mathcal{X}[i])$ in $\mathcal{P}_i[j]$ for some $j \leq i$, denoted $p(x_i, j)$, is a vertex x_j such that there exists a sequence of vertices $x_j, x_{j+1}, \ldots, x_i$, where

1. $x_k \in \mathcal{P}_i[k]$ for each $j \leq k \leq i$, and
2. $x_{k-1} = s_{M[k]}(x_k)$ for each $j \leq k \leq i$.

The *origin* of x_i, denoted $\omega(x_i)$, is the vertex $p(x_i, j)$ such that either $s_{M[j]}(x_j) \notin \mathcal{P}_i[j-1]$ or $j = 0$.

Fact 91, which states that $\mathcal{P}_i[0]$ contains the origins of the vertices of $\mathcal{X}[i]$ that belong to $A[0]$, follows from Definitions 89 and 90. This is because the set $\mathcal{P}_i[j]$ contains the proxies of the vertices of $\mathcal{X}[i]$ that are present in $A[j]$, and the proxies present in $A[0]$ are origins by Definition 90.

Fact 91. *Let $\langle \mathcal{E}_i \rangle_0^t$ be an LA edit pair sequence, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$. For a given $1 \leq i \leq t$, let $\mathcal{A}_i = \langle (A[j]; \Lambda_{A[j]}) \rangle_0^i$ be an antecedent sequence and $\mathcal{M}_i = \langle (M[j]; \Lambda_{M[j]}) \rangle_1^i$ be a merge sequence. Then, $\mathcal{P}_i[0] = \{\omega(x_i) \mid x_i \in V(\mathcal{X}[i])\} \cap V(A[0])$.*

Definition 92. Let $\langle \mathcal{E}_i \rangle_0^t$ be an LA edit pair sequence, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$. For a given $1 \leq i \leq t$, let $\mathcal{A}_i = \langle (A[j]; \Lambda_{A[j]}) \rangle_0^i$ be an antecedent sequence and $\mathcal{M}_i = \langle (M[j]; \Lambda_{M[j]}) \rangle_1^i$ be a merge sequence. Then, the *origin graph* of $(\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]})$, denoted $(\Omega(\mathcal{X}[i]); \Lambda_{\Omega(\mathcal{X}[i])})$, is given by $\Omega(\mathcal{X}[i]) = (A[0])[\mathcal{P}_i[0]]$ and $\Lambda_{\Omega(\mathcal{X}[i])}(x) = \Lambda_{A[0]}(x)$ for each $x \in \Omega(\mathcal{X}[i])$.

An LA edit pair sequence $\langle \mathcal{E}_i \rangle_0^t$ fits G_0 with respect to a graph sequence $\langle G_j \rangle_0^t$ only if $\mathcal{E}_i$ fits G_i for each $0 \leq i \leq t$ (Definition 39). To determine whether $\mathcal{E}_i$ fits G_i, by Definition 16, we need to check a list-isomorphism condition and a degree condition. In Lemma 93, we reduce the list-isomorphism condition on $(\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]})$ and G_i to a list-isomorphism condition on the origin graph of $(\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]})$ and G_0 by applying Lemma 80.

Lemma 93. *Let $\langle \mathcal{E}_i \rangle_0^t$ be an LA edit pair sequence, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$. For $1 \leq i \leq t$, suppose there exists a sequence of graphs $\langle G_j \rangle_0^i$ such that $\langle \mathcal{E}_j \rangle_0^{i-1}$ is complete in G_0 with respect to $\langle G_j \rangle_0^i$.*

Then, for any good antecedent and merge sequences with respect to $\langle G_j \rangle_0^i$, $(\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]})$ is list-isomorphic to an induced subgraph of G_i if and only if $(\Omega(\mathcal{X}[i]); \Lambda_{\Omega(\mathcal{X}[i])})$ is list-isomorphic to an induced subgraph of G_0.

Proof. Consider an arbitrary $1 \leq i \leq t$. Since $\langle \mathcal{E}_j \rangle_0^{i-1}$ is complete in G_0, we know $\mathcal{E}_j$ is complete in G_j for each $0 \leq j \leq i-1$ (Definition 40), and so $(\mathcal{Y}[j]; \Lambda_{\mathcal{Y}[j]})$ is list-isomorphic to an induced subgraph of G_{j+1} (Fact 29).

Let $\mathcal{A}_i = \langle (A[j]; \Lambda_{A[j]}) \rangle_0^i$ and $\mathcal{M}_i = \langle (M[j]; \Lambda_{M[j]}) \rangle_1^i$ be good with respect to $\langle G_j \rangle_0^i$.

Let $\langle (B[j]; \Lambda_{B[j]}) \rangle_0^i$ be a sequence of LA graphs defined by $B[j] = (A[i])[\mathcal{P}_i[j]]$ and $\Lambda_{B[j]}(b) = \Lambda_{A[j]}(b)$ for each $b \in V(B[j])$. Note that $V(B[j]) = \mathcal{P}_i[j]$. Also, observe that $(\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}) = (B[i]; \Lambda_{B[i]})$ and $(\Omega(\mathcal{X}[i]); \Lambda_{\Omega(\mathcal{X}[i])}) = (B[0]; \Lambda_{B[0]})$ by Definitions 82 and 89.

We will show that $(B[j]; \Lambda_{B[j]})$ is list-isomorphic to an induced subgraph of G_j if and only if $(B[j-1]; \Lambda_{B[j-1]})$ is list-isomorphic to an induced subgraph of G_{j-1} for each $1 \leq j \leq i$ by directly applying Lemma 80. This will imply $(B[i]; \Lambda_{B[i]}) = (\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]})$ is list-isomorphic to an induced subgraph of G_i if and only if $(B[0]; \Lambda_{B[0]}) = (\Omega(\mathcal{X}[i]); \Lambda_{\Omega(\mathcal{X}[i])})$ is list-isomorphic to an induced subgraph of G_0, and complete our proof.

We show that if we substitute $(B[j]; \Lambda_{B[j]})$ for $(Z; \Lambda_Z)$, then $(B[j-1]; \Lambda_{B[j-1]})$ can substitute for $(R; \Lambda_R)$ in Lemma 80.

In Lemma 80, we assign $G = G_{j-1}$, $\mathcal{E} = \mathcal{E}_{j-1}$, $G' = G_j$, $(A; \Lambda_A) = (A[j]; \Lambda_{A[j]})$, $(M; \Lambda_M) = (M[j]; \Lambda_{M[j]})$, and $(Z; \Lambda_Z) = (B[j]; \Lambda_{B[j]})$.

Now, we show $(R; \Lambda_R) = (B[j-1]; \Lambda_{B[j-1]})$. First, we begin with $R = \alpha(M)[\sigma_M(V(Z)) \cap V(\alpha(M))]$ as given in Lemma 80 and show that $R = B[j-1]$. Note that $\alpha(M) = \alpha(M[j]) = (A[j-1]; \Lambda_{A[j-1]})$ by Definition 82, which means $R = (A[j-1])[\sigma_M(V(Z)) \cap V(\alpha(M))]$. Moreover, since $V(Z) = V(B[j]) = \mathcal{P}_i[j]$, using the substitutions stated above, we obtain the substitute set $\sigma_M(V(Z)) \cap V(\alpha(M)) = \sigma_{M[j]}(\mathcal{P}_i[j]) \cap V(A[j-1])$. By Definition 89, we know this set equals $\mathcal{P}_i[j-1]$. Hence, $R = (A[j-1])[\mathcal{P}_i[j-1]] = B[j-1]$. Furthermore, as defined in Lemma 80, $\Lambda_R(r) = \Lambda_{\alpha(M)}(r) = \Lambda_{A[j-1]}(r)$ for each $r \in V(R)$. Now, $\Lambda_{A[j-1]}(r) = \Lambda_{B[j-1]}(r)$ because $B[j-1]$ is a subgraph of $A[j-1]$, which implies $\Lambda_R(r) = \Lambda_{B[j-1]}(r)$ for each $r \in V(R)$. Therefore, $(R; \Lambda_R) = (B[j-1]; \Lambda_{B[j-1]})$.

First, we show that the four assumptions of Lemma 80 hold. Since $\langle \mathcal{E}_j \rangle_0^{i-1}$ fits G_0, G_j is a $(G_{j-1}, \mathcal{E}_{j-1})$-produced graph for each $1 \leq j \leq i$ (Definition 39). Since $\mathcal{A}_i$ and $\mathcal{M}_i$ are good with respect to $\langle G_j \rangle_0^i$, $(A[j]; \Lambda_{A[j]})$ and $(M[j]; \Lambda_{M[j]})$ are list-isomorphic to subgraphs of G_j, i.e., $(M[j]; \Lambda_{M[j]})$ is G_j-realized. Finally, $B[j]$ is an induced subgraph of $A[j]$ and hence satisfies $\Lambda_{B[j]}(b) = \Lambda_{A[j]}(b)$ for each $b \in V(B[j])$.

Next, applying Lemma 80, we obtain $(Z; \Lambda_Z) = (B[j]; \Lambda_{B[j]})$ is list-isomorphic to an induced subgraph of G_j if and only if $(R; \Lambda_R) = (B[j-1]; \Lambda_{B[j-1]})$ is list-isomorphic to an induced subgraph of G_{j-1} for each $1 \leq j \leq i$. Continuing similarly, we obtain $(B[j]; \Lambda_{B[j]})$ is list-isomorphic to an induced subgraph of G_i if and only if $(B[0]; \Lambda_{B[0]})$ list-isomorphic to an induced subgraph of G_0 for each $1 \leq j \leq i$. Setting $j = i$ proves our result. $\square$

To determine whether $\mathcal{E}_i$ fits G_i, we also need to check the degree condition in Definition 16 on the vertices in $V_{-}(\mathcal{E}_i) \cup M_{deg}(\mathcal{E}_i)$ with respect to G_i. In Lemma 94, we reduce it to a degree condition on the origins of these vertices with respect to G_0.

Furthermore, for a vertex $x \in V(\mathcal{X}[i])$, the reduced condition requires that the degree of a proxy of x in $A[j]$ be the same as the degree of the substitute of the proxy in $M[j]$ for each $j \leq i$.

For each vertex of $\mathcal{X}[i]$ whose origin is not present in $A[0]$, we show the degree condition with respect to G_i holds.

Lemma 94. *Let $\langle \mathcal{E}_i \rangle_0^t$ be an LA edit pair sequence, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$. For a given $1 \leq i \leq t$, suppose there exists a sequence of graphs $\langle G_j \rangle_0^i$ such that $\langle \mathcal{E}_j \rangle_0^{i-1}$ is complete in G_0 with respect to $\langle G_j \rangle_0^i$. Also suppose antecedent and merge sequences $\mathcal{A}_i = \langle (A[j]; \Lambda_{A[j]}) \rangle_0^i$ and $\mathcal{M}_i = \langle (M[j]; \Lambda_{M[j]}) \rangle_1^i$ are good with respect to $\langle G_j \rangle_0^i$.*

Suppose $(\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]})$ is list-isomorphic to an induced subgraph of G_i with respect to a bijection f_i and $(A[0]; \Lambda_{A[0]})$ is list-isomorphic to an induced subgraph of G_0 with respect to a bijection f_0. Consider a vertex $x \in V(\mathcal{X}[i])$ and let $\omega(x) \in \mathcal{P}_i[k]$ for some $k \geq 0$. Then, $d_{\mathcal{X}[i]}(x) = d_{G_i}(f_i(x))$ if and only if

1. *$d_{A[0]}(\omega(x)) = d_{G_0}(f_0(\omega(x)))$ if $k = 0$, and*
2. *$d_{A[j]}(p(x, j)) = d_{M[j]}(s_{M[j]}(p(x, j)))$ for each $k \leq j \leq i$.*

Proof. Since $\mathcal{A}_i$ and $\mathcal{M}_i$ are good with respect to $\langle G_j \rangle_0^i$, $(A[j]; \Lambda_{A[j]})$ and $(M[j]; \Lambda_{M[j]})$ are list-isomorphic to subgraphs of G_j with respect to bijections, say, a_j and m_j, respectively, for each $0 \leq j \leq i$ (Definition 83). Note that $(A[i]; \Lambda_{A[i]}) = (\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]})$ by Definition 82. Hence, $a_i = f_i$ and $a_0 = f_0$.

To prove the lemma, we prove Claim 95, which reduces a degree condition on a vertex v of $A[j]$ with respect to G_j to a degree condition on the substitute of v with respect to $M[j]$ and G_{j-1}.

Observe that, by Definition 90, the sequence $s_{M[j]}(v), s_{M[j-1]}(s_{M[j]}(v)), \ldots$ is the same as $p(v, j), p(v, j-1), \ldots$ because, for any $0 \leq h \leq j$, the sequence of vertices $s_{M[h]}(\ldots(s_{M[j]}(v))\ldots), \ldots, s_{M[j]}(v)$ satisfies Conditions (1) and (2) of Definition 90. Therefore, continuing the sequence, we obtain the origin $\omega(v) = p(v, k)$ (Definition 90). Furthermore, $\mathcal{P}_i[0]$ contains the origins of the vertices of $\mathcal{X}[i]$ that are present in $A[0]$. Therefore, since $s_{M[h]}(\ldots(s_{M[j]}(v))\ldots) = p(v, h)$ for $0 \leq h \leq j$, Conditions (1) and (2) of the lemma follow from Claim 95. Later, we will show that, when $k > 0$, the condition analogous to Condition (1) automatically holds.

Claim 95. For each $1 \le j \le i$, $d_{A[j]}(v) = d_{G_j}(a_j(v))$ if and only if $d_{A[j-1]}(s_{M[j]}(v)) = d_{G_{j-1}}(a_{j-1}(s_{M[j]}(v)))$ and $d_{A[j]}(v) = d_{M[j]}(s_{M[j]}(v))$ for each $v \in \mathcal{P}_i[j]$ such that $s_{M[j]}(v) \in V(A[j-1])$.

Proof of Claim 95. Consider $v \in \mathcal{P}_i[j]$ such that $s_{M[j]}(v) \in V(A[j-1])$. We know $s_{M[j]}(v) \in V(M[j])$ by Definition 68, which means $s_{M[j]}(v) \in V(M[j]) \cap V(A[j-1])$. This further implies $m_j(s_{M[j]}(v)) = a_{j-1}(s_{M[j]}(v))$ because they both refer to the same vertex in $V(G_j) \cap V(G_{j-1})$.

We use Lemma 81 to prove either direction of our claim. In Lemma 81, we substitute $G = G_{j-1}$, $G' = G_j$, $\mathcal{E} = \mathcal{E}_{j-1}$, $(M; \Lambda_M) = (M[j]; \Lambda_{M[j]})$, $(A; \Lambda_A) = (A[j]; \Lambda_{A[j]})$, and the function $g = m_j$. Then, $\alpha(M) = \alpha(M[j]) = (A[j-1]; \Lambda_{A[j-1]})$ by Definition 82. Note that the assumption of Lemma 81 holds because $(M[j]; \Lambda_{M[j]})$ is list-isomorphic to a subgraph of G_j with respect to m_j. Furthermore, we know $s_{M[j]}(v) \in V(M[j]) \cap V(A[j-1]) = V(M) \cap V(\alpha(M))$.

To show the forward direction, we suppose $d_{A[j]}(v) = d_{G_j}(a_j(v))$. Since $(M[j]; \Lambda_{M[j]}) \in (\mathcal{Y}[j-1]; \Lambda_{\mathcal{Y}[j-1]}) \leftrightarrow (A[j]; \Lambda_{A[j]})$, we know by Lemma 71 that $A[j]$ is isomorphic to an induced subgraph of $M[j]$ with respect to the bijection $s_{M[j]}$, which implies $a_j(v) = m_j(s_{M[j]}(v))$. Therefore, since $M[j]$ is isomorphic to a subgraph of G_j, we obtain $d_{M[j]}(s_{M[j]}(v)) = d_{G_j}(m_j(s_{M[j]}(v)))$, which implies $d_{A[j]}(v) = d_{M[j]}(s_{M[j]}(v))$ since $a_j(v) = m_j(s_{M[j]}(v))$.

Now, we prove $d_{A[j-1]}(s_{M[j]}(v)) = d_{G_{j-1}}(a_{j-1}(s_{M[j]}(v)))$ using Lemma 81. Since we know $d_{M[j]}(s_{M[j]}(v)) = d_{G_j}(m_j(s_{M[j]}(v)))$, applying Lemma 81 in the forward direction leads to $d_{A[j-1]}(s_{M[j]}(v)) = d_{G_{j-1}}(m_j(s_{M[j]}(v)))$. But we also know $m_j(s_{M[j]}(v)) = a_{j-1}(s_{M[j]}(v))$, which means $d_{G_{j-1}}(m_j(s_{M[j]}(v))) = d_{G_{j-1}}(a_{j-1}(s_{M[j]}(v)))$. Therefore, we obtain $d_{A[j-1]}(s_{M[j]}(v)) = d_{G_{j-1}}(a_{j-1}(s_{M[j]}(v)))$.

To prove the backward direction, suppose $d_{A[j-1]}(s_{M[j]}(v)) = d_{G_{j-1}}(a_{j-1}(s_{M[j]}(v)))$ and $d_{A[j]}(v) = d_{M[j]}(s_{M[j]}(v))$. Since we know $m_j(s_{M[j]}(v)) = a_{j-1}(s_{M[j]}(v))$, our first supposition can be rewritten as $d_{A[j-1]}(s_{M[j]}(v)) = d_{G_{j-1}}(m_j(s_{M[j]}(v)))$. Now, applying Lemma 81 in the backward direction, we obtain $d_{M[j]}(s_{M[j]}(v)) = d_{G_j}(m_j(s_{M[j]}(v)))$. But since we have also supposed $d_{A[j]}(v) = d_{M[j]}(s_{M[j]}(v))$, we obtain $d_{A[j]}(v) = d_{G_j}(m_j(s_{M[j]}(v)))$. Finally, we know $a_j(v) = m_j(s_{M[j]}(v))$, which implies $d_{A[j]}(v) = d_{G_j}(a_j(v))$. □

Now, if $\omega(x) \in \mathcal{P}_i[k]$ for some $k > 0$, then we show $d_{A[k]}(\omega(z)) = d_{G_k}(a_k(\omega(z))) = 0$. In other words, we show an origin not present in $A[0]$ must be introduced in the sequence as an isolated vertex, and so it must satisfy the degree condition with respect to G_k.

Let $u = \omega(x) \in \mathcal{P}_i[k]$ be the origin of x. We know u is the "earliest" proxy of x, and so x has no proxy in $A[k-1]$ (Definition 90). In other words, $s_{M[k]}(u) \notin V(A[k-1])$ because, otherwise, $s_{M[k]}(u)$ would be a proxy of x in $A[k-1]$. Since $\mathcal{P}_i[k] \subseteq V(A[k])$

by item (2) of Definition 89 and $A[k]$ is isomorphic to an induced subgraph of $M[k]$ with respect to $s_{M[k]}$ by Lemma 71, it follows that $s_{M[k]}(u) \in V(M[k])$. Moreover, by Definition 82, we know $A[k-1] = \alpha(M[k])$, which means $s_{M[k]}(u) \notin V(\alpha(M[k])$ because we know $s_{M[k]}(u) \notin V(A[k-1])$. Therefore, $s_{M[k]}(u) \in V(M[k]) \setminus V(\alpha(M[k]))$, which implies $s_{M[k]}(u) \in V_+(\mathcal{E}_{k-1})$ by Definition 74. In other words, G_k is obtained by inserting the vertex $m_k(s_{M[k]}(u))$ in G_{k-1}, which means $d_{M[k]}(s_{M[k]}(u)) = d_{G_k}(m_k(s_{M[k]}(u))) = 0$ by Definition 12. But $s_{M[k]}(u)$ is the substitute of u in $M[k]$, and so $a_k(u)$ and $m_k(s_{M[k]}(u))$ refer to the same inserted vertex in G_k. Therefore, $d_{A[k]}(u) = d_{G_k}(a_k(u)) = 0$. □

Recall that by Theorem 66, to solve GEN, we need to determine whether there exists a provisionally complete LA edit pair sequence $\langle \mathcal{E}_i \rangle_0^t$ that fits an input graph G_0. We use Lemmas 93 and 94, and Fact 88 to reduce it to a condition on the origin graph and G_0.

Theorem 96. *Let $\langle \mathcal{E}_i \rangle_0^t$, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$, be a provisionally complete LA edit pair sequence. Then, $\langle \mathcal{E}_i \rangle_0^t$ is complete in a graph G_0 if and only if $\mathcal{E}_0$ fits G_0 and, for each $1 \leq i \leq t$, there exist an antecedent sequence $\mathcal{A}_i = \langle (A[j]; \Lambda_{A[j]}) \rangle_0^i$ and a merge sequence $\mathcal{M}_i = \langle (M[j]; \Lambda_{M[j]}) \rangle_1^i$ such that*

1. *$(A[0]; \Lambda_{A[0]})$ is list-isomorphic to a subgraph of G_0 with respect to a bijection f_0,*
2. *$(\Omega(\mathcal{X}[i]); \Lambda_{\Omega(\mathcal{X}[i])})$ is list-isomorphic to an induced subgraph of G_0, and*
3. *for each $x \in V_-(\mathcal{E}_i) \cup M_{deg}(\mathcal{E}_i)$, letting $\omega(x) \in \mathcal{P}_i[k]$,*
   (a) *$d_{A[0])}(\omega(x)) = d_{G_0}(f_0(\omega(x)))$ if $k = 0$, and*
   (b) *$d_{A[j]}(p(x,j)) = d_{M[j]}(s_{M[j]}(p(x,j)))$ for each $i \geq j \geq k$.*

Proof. First, we show that $\langle \mathcal{E}_i \rangle_0^t$ is complete in G_0 if and only if $\langle \mathcal{E}_j \rangle_0^i$ is complete in G_0 for each $0 \leq i \leq t$.

Suppose $\langle \mathcal{E}_i \rangle_0^t$ is complete in G_0. Then, there exists a graph sequence $\langle G_j \rangle_0^{t+1}$ such that G_{i+1} is a $(G_i, \mathcal{E}_i)$-produced graph and $\mathcal{E}_i$ is complete in G_i for each $0 \leq i \leq t$ (Definition 40). Hence, $\mathcal{E}_i$ fits G_i for each $0 \leq i \leq t$ (Definition 26). This implies $\langle \mathcal{E}_j \rangle_0^i$ fits G_0 and, since $\mathcal{E}_j$ fits G_j for each $0 \leq j \leq i$, $\langle \mathcal{E}_j \rangle_0^i$ is complete in G_0 for each $0 \leq i \leq t$ (Definition 40).

For the other direction, if $\langle \mathcal{E}_j \rangle_0^i$ is complete in G_0 for each $0 \leq i \leq t$, then obviously $\langle \mathcal{E}_i \rangle_0^t$ is complete in G_0.

Now, by induction on i, we show that $\langle \mathcal{E}_j \rangle_0^i$ is complete in G_0 for each $0 \leq i \leq t$ if and only if Conditions (1), (2), and (3) of the lemma are satisfied.

Base case. Since $\langle \mathcal{E}_i \rangle_0^t$ is provisionally complete, $\mathcal{E}_i$ is provisionally complete for each $0 \leq i \leq t$ (Definition 41), and so $\mathcal{E}_0$ is provisionally complete. Hence, $\mathcal{E}_0$ is complete if and only if $\mathcal{E}_0$ fits G_0 (Definitions 26 and 30). We do not need to argue further for the base case because the statement that $\mathcal{E}_0$ fits G_0 is a condition in the theorem.

Induction step. Suppose $\langle \mathcal{E}_j \rangle_0^{i-1}$ is complete in G_0 with respect to a graph sequence $\langle G_j \rangle_0^i$. By Definitions 39 and 40, $\langle \mathcal{E}_j \rangle_0^i$ is complete in G_0 if and only if $\mathcal{E}_i$ fits G_i and is complete in G_i. But since $\mathcal{E}_i$ is provisionally complete, we only need to determine whether $\mathcal{E}_i$ fits G_i (Definition 30). By Definition 16, then, this reduces to determining whether $(\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]})$ is list-isomorphic to an induced subgraph of G_i and $d_{\mathcal{X}[i]}(x) = d_{G_i}(f_i(x))$ for each $x \in V_-(\mathcal{E}_i) \cup M_{deg}(\mathcal{E}_i)$.

By Fact 88, for any graph sequence $\langle G_j \rangle_0^i$, $\mathcal{A}_i$ and $\mathcal{M}_i$ are good with respect to $\langle G_j \rangle_0^i$ if and only if $(A[0]; \Lambda_{A[0]})$ is list-isomorphic to a subgraph of G_0 (Condition (1)). If there exist such good sequences $\mathcal{A}_i$ and $\mathcal{M}_i$, then we know by Lemma 93 that $(\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]})$ is list-isomorphic to an induced subgraph of G_i with respect to a bijection, say, f_i, if and only if $(\Omega(\mathcal{X}[i]); \Lambda_{\Omega(\mathcal{X}[i])})$ is list-isomorphic to an induced subgraph of G_0 (Condition (2)). Finally, for each vertex $x \in V_-(\mathcal{E}_i) \cup M_{deg}(\mathcal{E}_i)$, $x \in V(\mathcal{X}[i])$ because $V_-(\mathcal{E}_i) \subseteq V(\mathcal{X}[i])$ (Definition 14) and $M_{deg}(\mathcal{E}_i) \subseteq V_R(\mathcal{E}_i) = V(\mathcal{X}[i]) \cap V(\mathcal{Y}[i])$ (Fact 13). By (1) in Lemma 94, we know $d_{\mathcal{X}[i]}(x) = d_{G_i}(f_i(x))$ if and only if Conditions (3a) and (3b).

Hence, $\mathcal{E}_i$ fits G_i, and so is complete in G_i, if and only if Conditions (1), (2), and (3) are satisfied. This means, for each $1 \leq i \leq t$, $\langle \mathcal{E}_j \rangle_0^i$ is complete in G_0 if and only if the three conditions are satisfied.

Thus, $\langle \mathcal{E}_i \rangle_0^t$ is complete in G_0 if and only if $\mathcal{E}_0$ fits G_0 and Conditions (1), (2), and (3) are satisfied for each $1 \leq i \leq t$. □

Theorem 97. *Let $\langle \mathcal{E}_i \rangle_0^t$, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ for $0 \leq i \leq t$, be a provisionally complete LA edit pair sequence. Determining whether $\langle \mathcal{E}_i \rangle_0^t$ is complete in a graph G_0 is in FPT when parameterized by $\Delta(G_0) + t$.*

Proof. To show $\langle \mathcal{E}_i \rangle_0^t$ is complete in a graph G_0, we need to check whether $\mathcal{E}_0$ fits G_0 and the three conditions in Theorem 96 hold for each $1 \leq i \leq t$. For a given i, we will show that each condition can be checked in FPT time for the parameter $\Delta(G_0) + t$ using the fact that $k_i^1 = |V(\mathcal{X}[i])| + \sum_{p=0}^{i-1} |V(\mathcal{Y}[p])|$ is bounded by a function of $\Delta(G_0) + t$ (Fact 86).

First, we bound the time for computing all possibilities of antecedent and merge sequences $\mathcal{A}_i$ and $\mathcal{M}_i$. By Fact 85, the number of choices for $\mathcal{A}_i$ and $\mathcal{M}_i$ is bounded by a function of $i + k_i^1$, which is a function of $\Delta(G_0) + t$ since $i \leq t$. Now, referring to Definition 82, $(M[j]; \Lambda_{M[j]})$ can be computed in time $\mathcal{O}(|V(\mathcal{Y}[j-1])|^2 + |V(A[j])|^2)$ (Fact 69) and $(A[j-1]; \Lambda_{A[j-1]})$ can be computed in time $\mathcal{O}(|V(\mathcal{X}[j-1])|)$ (Fact 75)

for each $1 \leq j \leq i$. We know $|V(A[j])| \leq k_i^j + 1 < k_i^1 + 1$ by Fact 84 and $\langle \mathcal{E}_i \rangle_0^t$ fits G_0 only if $|V(\mathcal{X}[j-1])|$ and $|V(\mathcal{Y}[j-1])|$ are each at most $(\Delta(G_0) + t + 1)^2 + 1$ (Fact 43), which is a function of $\Delta(G_0) + t$. Therefore, $(A[j-1]; \Lambda_{A[j-1]})$ and $(M[j]; \Lambda_{M[j]})$ can each be computed in time bounded by a function of $\Delta(G_0) + t$. Hence, $\mathcal{A}_i$ and $\mathcal{M}_i$ can be computed in time bounded by a function of $\Delta(G_0) + t$ because $i \leq t$. Therefore, since the number of choices for $\mathcal{A}_i$ and $\mathcal{M}_i$ is bounded by a function of $\Delta(G_0) + t$, the time for computing all possibilities of $\mathcal{A}_i$ and $\mathcal{M}_i$ is bounded by a function of $\Delta(G_0) + t$.

Observe that if $\mathcal{A}_i$ and $\mathcal{M}_i$ are given, then the proxy set $\mathcal{P}_i[j]$ for each $0 \leq j \leq i$ can be computed by Definition 89. Given the proxy sets, and the proxy $p(x, j)$ for each $x \in V(\mathcal{X}[i])$ can be computed by Definition 90. Given the proxies, the origin $\omega(x)$, which is the "earliest" proxy, is given for each $x \in V(\mathcal{X}[i])$.

To determine whether $\langle \mathcal{E}_i \rangle_0^t$ is complete in G_0, we try to find a choice of $\mathcal{A}_i$ and $\mathcal{M}_i$ that satisfies the three conditions in Theorem 96. We show that each condition can be checked in FPT time when parameterized by $\Delta(G_0) + t$.

(1) and (3a) We know $|V(A[0])| \leq k_i^1 + i + 1$ by substituting $j = 1$ in Fact 84. Since $i \leq t$, this implies there exists a function f such that

$$|V(A[0])| \leq f(\Delta(G_0) + t) \tag{7.3.1}$$

By Lemma 35, checking whether $(A[0]; \Lambda_{A[0]})$ is list-isomorphic to a subgraph of G_0 such that Condition (3a) in Theorem 96 holds is in FPT when parameterized by $\Delta(G_0) + |V(A[0])|$, and so is in FPT when parameterized by $\Delta(G_0) + t$.

(2) Since $(\Omega(\mathcal{X}[i]); \Lambda_{\Omega(\mathcal{X}[i])})$ is an induced subgraph of $A[0]$ (Definition 92), $|V(\Omega(\mathcal{X}[i]))| \leq |V(A[0])|$, and so by Equation 7.3.1 $|V(\Omega(\mathcal{X}[i]))|$ is bounded by a function of $\Delta(G_0) + t$. Hence, by Lemma 35, checking whether $(\Omega(\mathcal{X}[i]); \Lambda_{\Omega(\mathcal{X}[i])})$ is list-isomorphic to an induced subgraph of G_0 is in FPT when parameterized by $\Delta(G_0) + |V(\Omega(\mathcal{X}[i]))|$, and so is in FPT when parameterized by $\Delta(G_0) + t$.

(3b) For each $x \in V_-(\mathcal{E}_i) \cup M_{deg}(\mathcal{E}_i)$, checking whether $d_{A[j]}(p(x, j)) = d_{M[j]}(s_{M[j]}(p(x, j)))$ takes time bounded by a function of $\Delta(G_0) + t$. This is because $|V(A[j])|$ is bounded by a function of $k_i^{j+1} + 1 \leq k_i^1 + 1$ and $|V(M[j])|$ is bounded by a function of $k_i^j \leq k_i^1$ (Fact 84), and we know k_i^1 is bounded by a function of $\Delta(G_0) + t$. Therefore, the number of neighbours of a vertex of $A[j]$ or $M[j]$ is bounded by a function of $\Delta(G_0) + t$. Hence, checking (3b) takes time bounded by a function of $\Delta(G_0) + t$.

Thus, checking the three conditions for a given i takes FPT time when parameterized by $\Delta(G_0) + t$. Since $i \leq t$, checking them for all i also takes FPT time when parameterized by $\Delta(G_0) + t$. Furthermore, since the number of choices for $\mathcal{A}_i$ and $\mathcal{M}_i$ is bounded by a function of $\Delta(G_0) + t$, the total time taken to check the conditions for all choices of $\mathcal{A}_i$ and $\mathcal{M}_i$ is bounded by a function of $\Delta(G_0) + t$.

Finally, notice that we have yet to determine whether $\mathcal{E}_0$ is complete in G_0. Since $\mathcal{E}_0$ is provisionally complete by Definition 41, we only need to check whether $\mathcal{E}_0$ fits G_0. By Lemma 35, determining whether $(\mathcal{X}[0]; \Lambda_{\mathcal{X}[0]})$ is list-isomorphic to an induced subgraph of G_0 with respect to a bijection f such that $d_X(x) = d_G(f(x))$ for each $x \in V_-(\mathcal{E}) \cup M_{deg}(\mathcal{E})$ is in FPT when parameterized by $\Delta(G_0) + |V(\mathcal{X}[0])|$, or, when parameterized by $\Delta(G_0) + t$ since $|V(\mathcal{X}[0])|$ is bounded by $\mathcal{O}((\Delta(G_0) + t)^2)$. □

Chapter 8

FPT Algorithms for GEN and CGEN

8.1 FPT algorithm for GEN

To decide a GEN instance $(G_0, \mathcal{T}, \ell)$, recall that we need to determine whether there exists an LA edit pair sequence of length at most ℓ that is complete in G_0 and produces a graph with NDL $\mathcal{T}$ (see page 31 for the problem statement). Given a provisionally complete LA edit pair sequence of length t, we can check whether it is complete in G_0 (Theorem 96) and determine the NDL of the graph produced. But first, to decide $(G_0, \mathcal{T}, \ell)$ in FPT time for the parameter $\Delta(G_0)+\ell$, we must show that the search space of LA edit pair sequences of length at most ℓ is bounded by a function of $\Delta(G_0) + \ell$.

8.1.1 Bounding the search space

We know an LA edit pair sequence $\langle \mathcal{E}_i \rangle_0^t$ fits G_0 only if $|V(\mathcal{X}[i])|$ and $|V(\mathcal{Y}[i])|$ are at most $(\Delta(G_0) + t + 1)^2 + 1$ for each $0 \leq i \leq t$ (Fact 43). Hence, we limit the search space to LA edit pair sequences that satisfy this condition.

For a given $t \leq \ell$, we compute the number of list-attributed graphs on at most n vertices, where $n = \Delta(G_0 + t + 1)^2 + 1$.

First, we bound the number of list-attributions possible for such a graph, i.e., the number of possibilities for $\Lambda_{\mathcal{X}[i]}$ and $\Lambda_{\mathcal{Y}[i]}$. Suppose $\langle \mathcal{E}_i \rangle_0^t$ fits G_0 with respect to a graph sequence $\langle G_j \rangle_0^{t+1}$. Since $\Delta(G_{i+1}) \leq \Delta(G_i) + 1$ by Fact 22, $\Delta(G_{i+1}) \leq \Delta(G_0) + t$ for each $0 \leq i \leq t$. Now, since $(\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]})$ is list-isomorphic to an induced subgraph of G_i (Definition 16) and $(\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]})$ is list-isomorphic to an induced subgraph of G_{i+1} (Fact 29), $|\Lambda_{\mathcal{X}[i]}(x)| \leq \Delta(G_i) \leq \Delta(G_0) + t$ for each $x \in V(\mathcal{X}[i])$ and $|\Lambda_{\mathcal{Y}[i]}(y)| \leq \Delta(G_{i+1}) \leq$

$\Delta(G_0) + t$ for each $y \in V(\mathcal{Y}[i])$. Therefore, the number of possibilities for $\Lambda_{\mathcal{X}[i]}(x)$ and $\Lambda_{\mathcal{Y}[i]}(y)$ is bounded by $(\Delta(G_0) + t)^{(\Delta(G_0)+t)}$, which is a function of $\Delta(G_0) + t$. Since there are at most $(\Delta(G_0) + t + 1)^2 + 1$ vertices of $\mathcal{X}[i]$ and $\mathcal{Y}[i]$, the total number of possibilities for each $\Lambda_{\mathcal{X}[i]}$ and $\Lambda_{\mathcal{Y}[i]}$ is bounded by a function of $\Delta(G_0) + t$.

Next, for each $0 \leq m \leq n$, there are $2^{\binom{m}{2}} \leq 2^{\binom{n}{2}}$ graphs on m vertices. For each such graph, we have shown above that the number of list-attributions is bounded by a function of $\Delta(G_0) + t$. Thus, the total number of LA graphs on at most $n = \Delta(G_0 + t + 1)^2 + 1$ vertices is bounded by a function of $\Delta(G_0) + t$.

This implies the number of possibilities for $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$ is bounded by a function of $\Delta(G_0) + t$, and so the number of possible LA edit pair sequences is also bounded by a function of $\Delta(G_0) + t$. Hence, the total number of LA edit pair sequences of length at most ℓ in the search space is bounded by a function of $\Delta(G_0) + \ell$.

8.1.2 Solve-GEN procedure

In Solve-GEN, we iterate over all LA edit pair sequences in the search space. We output yes if we find a sequence that is provisionally complete (Line 5), produces a graph with NDL $\mathcal{T}$ (Line 10), and is complete in G_0 (Line 12), and no otherwise. The sequence of Young tableaux $\mathcal{N}$ stores the NDLs of the graphs in the graph sequence produced by an LA edit pair sequence.

Algorithm 3 Solve Graph-Edit-to-NDL

```
1:  procedure SOLVE-GEN(G_0,T,ℓ)
2:      for each ⟨E_i⟩_0^t in the search space do
3:          Declare sequence of Young tableaux N of length t
4:          for i = 0 to t do
5:              if CHECK-PROVISIONAL-COMPLETENESS(E_i)=NO then
6:                  return NO
7:          N[0] = NDL(G_0)
8:          for i = 0 to t do
9:              N[i+1] =COMPUTE-NDL(N[i],E_i)
10:         if N[t+1] ≠ T then
11:             return NO
12:         if E is complete in G_0 then
13:             return YES
14:     return NO
```

Theorem 98. SOLVE-GEN *correctly decides a GEN instance* $(G_0, \mathcal{T}, \ell)$ *in FPT time when parameterized by* $\Delta(G_0) + \ell$.

Proof. First, we show Solve-GEN correctly decides a GEN instance $(G_0, \mathcal{T}, \ell)$. Given an LA edit pair sequence $\langle \mathcal{E}_i \rangle_0^t$ in the search space, CHECK-PROVISIONAL-COMPLETENESS (on Line 5) correctly determines the provisional completeness of $\mathcal{E}_i$ for each $0 \leq i \leq t$ (Lemma 61), and so the provisional completeness of $\langle \mathcal{E}_i \rangle_0^t$ (Definition 41). Next, given the NDL of a $(G_0, \langle \mathcal{E}_i \rangle_0^n)$-produced graph, we know COMPUTE-NDL computes the NDL of a $(G_0, \langle \mathcal{E}_i \rangle_0^{n+1})$-produced graph (Lemma 55), and so the for-loop on Line 8 computes the NDL of a $(G_0, \langle \mathcal{E}_i \rangle_0^{n+1})$-produced graph for each $0 \leq n \leq t-1$. We check on Line 10 whether the NDL of a $(G_0, \langle \mathcal{E}_i \rangle_0^t)$-produced graph equals $\mathcal{T}$. Finally, we output YES if and only if $\langle \mathcal{E}_i \rangle_0^t$ is complete in G_0, which can be determined by Theorem 97 because $\langle \mathcal{E}_i \rangle_0^t$ is provisionally complete. By Theorem 66, then, SOLVE-GEN correctly decides $(G_0, \mathcal{T}, \ell)$.

To show a bound on the running time, we note that the number of LA edit pair sequences in the search space is bounded by a function of $\Delta(G_0) + \ell$.

For each candidate LA edit pair sequence $\langle \mathcal{E}_i \rangle_0^t$, since $t \leq \ell$, the **for** loop on Line 4 that checks whether $\langle \mathcal{E}_i \rangle_0^t$ is provisionally complete has $t + 1 \leq \ell + 1$ iterations. By Lemma 63, each iteration takes $\mathcal{O}(|V(\mathcal{X}[i])|^2)$ time, where $\mathcal{E}_i = ((\mathcal{X}[i]; \Lambda_{\mathcal{X}[i]}), (\mathcal{Y}[i]; \Lambda_{\mathcal{Y}[i]}))$. Since we know $|V(\mathcal{X}[i])| \leq (\Delta(G_0) + t + 1)^2 + 1$ for each $0 \leq i \leq t$ (Fact 43), the **for** loop takes $\mathcal{O}(t \cdot (\Delta(G_0) + t)^4)$ time, which is a function of $\Delta(G_0) + t$, and hence a function of $\Delta(G_0) + \ell$.

Next, the **for** loop on Line 8 that computes the NDL of a $(G_0, \langle \mathcal{E}_i \rangle_0^t)$-produced graph has $t + 1 \leq \ell + 1$ iterations, and the ith iteration takes time polynomial in $|V(G_i)|$ for each $0 \leq i \leq t$ (Lemma 56). We know $|V(G_i)| \leq |V(G_0)| + t$ since at most t vertices are inserted by an LA edit pair sequence of length t. Hence, the **for** loop takes time polynomial in $t \cdot (|V(G_0)| + t)$.

Finally, checking whether $\langle \mathcal{E}_i \rangle_0^t$ is complete in G_0 takes FPT time when parameterized by $\Delta(G_0) + t$ (Theorem 97), and so takes FPT time when parameterized by $\Delta(G_0) + \ell$.

Therefore, the total running time is bounded by $f(\Delta(G_0) + \ell) \cdot |V(G_0)|^{\mathcal{O}(1)}$ for some function f. □

8.2 FPT algorithm for CGEN

We adapt SOLVE-GEN to decide a CGEN instance $(G_0, \pi, \mathcal{T}, \ell)$ in FPT time when parameterized by $\Delta(G_0) + \ell$. If $\langle \mathcal{E}_i \rangle_0^t$ is a solution to a GEN instance $(G_0, \mathcal{T}, \ell)$, then $\langle \mathcal{E}_i \rangle_0^t$ is a solution to the CGEN instance $(G_0, \pi, \mathcal{T}, \ell)$ if and only if the NDL of a

$(G_0, \langle \mathcal{E}_i \rangle_0^n)$-produced graph satisfies π for each $0 \leq n \leq t$ (see page 31 for the definition of CGEN). Moreover, for any solution $\langle \mathcal{E}_i \rangle_0^t$ to $(G_0, \pi, \mathcal{T}, \ell)$, $\langle \mathcal{E}_i \rangle_0^t$ is a solution to the GEN instance $(G_0, \mathcal{T}, \ell)$. Therefore, the search space of LA edit pair sequences for $(G_0, \pi, \mathcal{T}, \ell)$ is the same as that for $(G_0, \mathcal{T}, \ell)$.

Theorem 99. *A CGEN instance $(G_0, \pi, \mathcal{T}, \ell)$ can be decided in FPT time when parameterized by $\Delta(G_0) + \ell$.*

Proof. Note that, given an LA edit pair sequence $\langle \mathcal{E}_i \rangle_0^t$ in the search space, SOLVE-GEN computes the NDL of a $(G_0, \langle \mathcal{E}_j \rangle_0^i)$-produced graph for each $0 \leq i \leq t$ (Line 9), which is stored in $\mathcal{N}[i]$. Therefore, to check whether an LA edit pair sequence $\langle \mathcal{E}_i \rangle_0^t$ is a solution to $(G_0, \pi, \mathcal{T}, \ell)$, we only need to check whether $\mathcal{N}[i]$ satisfies the NDL-property π for each $0 \leq i \leq t$.

Now, the definition of CGEN states that π is verifiable in polynomial time, i.e., in time polynomial in $|\mathcal{N}[i]|$. If $\langle G_j \rangle_0^{t+1}$ is the graph sequence produced by $\langle \mathcal{E}_i \rangle_0^t$, we know $|\mathcal{N}[i]| \leq (|V(G_i)|)^2$ because the NDS of each vertex of G_i has at most $|V(G_i)|$ elements and there are $|V(G_i)|$ such vertices. We also know that $|V(G_i)| \leq |V(G_0)| + t$ because at most t vertices are inserted by an LA edit pair sequence of length t. Therefore, we can verify whether every NDL in the sequence $\mathcal{N}$ satisfies π in time polynomial in $|V(G_0)| + t$, or, polynomial in $|V(G_0)| + \ell$ since $t \leq \ell$.

Thus, $(G_0, \pi, \mathcal{T}, \ell)$ can be decided in FPT time when parameterized by $\Delta(G_0) + \ell$. □

8.3 Implications

Consider a variant of GEN where we do not specify the NDL of the final graph in the sequence G_{t+1}, but only a property π that it needs to satisfy. By Theorem 99, this variant of GEN is in FPT when parameterized by $\Delta(G_0) + \ell$ because we can replace the condition $NDL(G_{t+1}) = \mathcal{T}$ in GEN with the condition that $NDL(G_{t+1})$ satisfies π.

This implies, for any graph property ψ which can be expressed as a Young property π of NDLs that is verifiable in polynomial time, the following problem is in FPT when parameterized by $\Delta(G_0) + \ell$.

Instance:	A triple (G_0, ψ, ℓ), where G_0 is a graph, ψ is a graph property which can be expressed as a Young property of NDLs, and ℓ is an integer.
Question:	Is there an edit pair sequence $\langle (\mathcal{X}[i], \mathcal{Y}[i]) \rangle_0^t$, where $t \leq \ell$, such that the graph $G_{t+1} = gedit^t(G_0, \langle (\mathcal{X}[i], \mathcal{Y}[i]) \rangle_0^t)$ satisfies ψ?

Recall that k-neighbourhood degree anonymity (Example 5), assortativity (Example 6), and average nearest neighbours degree (Example 7) are such graph properties.

Chapter 9

Conclusion

We have given an algorithm to show that the problems Graph-Edit-to-NDL and Constrained-Graph-Edit-to-NDL are in FPT when parameterized by $\Delta(G_0) + \ell$, where G_0 is the input graph and ℓ is the bound on the number of graph edits. We have also shown these problems are NP-complete. Moreover, our results imply, for any graph property that can be determined by the NDL alone, editing to a graph with the property is in FPT for the parameter $\Delta(G_0) + \ell$.

For the combined parameter $\Delta(G_0)+\ell$, our FPT solution is a stronger result than the FPT solution for editing to a graph with a given degree sequence [22] because the NDL is a stronger graph invariant than the degree sequence. However, to exploit this greater strength, we need to study in more detail the graph properties that can be expressed as properties of NDLs but not as properties of degree sequences. We have mentioned three such properties in this work – k-neighbourhood degree anonymity, assortativity, and average nearest neighbours degree. But the notion of NDL was introduced only recently [4], and the relationship between graph properties and properties of NDLs has not been studied.

Regarding the choice of our combined parameter, notice that we have not shown GEN is W-hard for $\Delta(G_0)$ or ℓ. However, we believe GEN is not in FPT when parameterized by ℓ since editing to a graph with a given degree sequence is W[1]-hard when parameterized by ℓ [22]. The complexity of GEN when parameterized by $\Delta(G_0)$ is a question for future study.

Finally, we believe the idea of determining whether a candidate sequence of graph edits is a solution by checking whether a certain graph is a subgraph of G_0 can be applied to various other graph editing problems. The only other problem we know where this idea is applied is graph editing to a k-degree anonymous graph [5]. We believe it can be applied to any editing problem where we do not need to compute the

intermediate graphs produced by a candidate sequence of graph edits to check whether they satisfy a certain graph property.

Notation

Antecedent	(Def. 74 on page 53)
Antecedent sequence	(Def. 82 on page 59)
Assortativity	(Example 6 on page 14)
Average nearest neighbours degree	(Example 7 on page 14)
CGEN	The problem Constrained-Graph-Edit-to-NDL. (Page 31)
Complete (LA edit pair sequence)	(Def. 40 on page 30)
Complete (LA edit pair)	(Def. 26 on page 25)
Degree anonymity	(Example 4 on page 13)
Edit pair	(Def. 8 on page 15)
Fits (edit pair sequence)	(Def. 38 on page 30)
Fits (edit pair)	(Def. 9 on page 15)
Fits (LA edit pair sequence)	(Def. 39 on page 30)
Fits (LA edit pair)	(Def. 16 on page 19)
GEN	The problem Graph-Edit-to-NDL. (Page 31)
Good	Property of an antecedent sequence or merge sequence with respect to a graph sequence. (Def. 83 on page 59
LA edit pair	(Def. 12 on page 17)

List isomorphism	(Def. 15 on page 19)
List-attributed (LA) graph	(Def. 11 on page 17)
Merge graph	(Def. 67 on page 49)
Merge sequence	(Def. 82 on page 59)
Neighbourhood degree anonymity	(Example 5 on page 13)
Provisionally complete (LA edit pair sequence)	(Def. 41 on page 30)
Provisionally complete (LA edit pair)	(Def. 30 on page 27)
Substitute	(Def. 68 on page 50)
Substitute set	(Def. 68 on page 50)
Young property	Property defined on Young tableaux. (Page 12)
Young tableau	(Page 12)

Greek Symbols

$\alpha(M)$	The antecedent of an LA merge graph $(M; \Lambda_M)$. (Def. 74 on page 53)
$\Omega(\mathcal{X}[n])$	The origin graph of $\mathcal{X}[n]$. (Def. 92 on page 63
$\omega(x_i)$	The origin of a vertex $x_i \in V(\mathcal{X}_i)$. (Def. 90 on page 62)
$\sigma_M(C)$	The substitute set of the vertex set C in $(M; \Lambda_M$. (Def. 68 on page 50)
φ'_L	Formula specifying list-isomorphism with a subgraph. (Def. 32 on page 28)
φ_L	Formula specifying list-isomorphism with an induced subgraph. (Def. 32 on page 28)

Mathematical Symbols

$(A; \Lambda) \simeq H$	$(A; \Lambda)$ is list-isomorphic to H. (Def. 15 on page 19)

$(A;\Lambda_A) \leftrightarrow (B;\Lambda_B)$	The family of merge graphs obtained by merging $(A;\Lambda_A)$ and $(B;\Lambda_B)$ (Page 49)
$(G,\mathcal{E})$-produced graph	(Def. 21 on page 21)
$(G,\mathcal{E},H,f)$	NDL-tuple. (Def. 16 on page 19)
$(G;\Lambda)$	A list-attributed graph. (Def. 11 on page 17)
$(G_0,\langle\mathcal{E}_i\rangle_0^t)$-produced graph	A graph produced by a sequence of NDL graph edits performed in G_0. (Def. 39 on page 30)
$\langle(\mathcal{X}[i],\mathcal{Y}[i])\rangle_0^t$	Edit pair sequence. (Def. 36 on page 30)
$\langle\mathcal{E}_i\rangle_0^t$	LA edit pair sequence. (Def. 37 on page 30)
$\mathcal{A}_i$	Antecedent sequence. (Def. 82 on page 59)
$\mathcal{E}$	LA edit pair. (Def. 12 on page 17)
$\mathcal{L} \ominus a$	List contraction: remove a from the nonincreasing list $\mathcal{L}$. (Page 12)
$\mathcal{L} \oplus a$	List extension: add a to the nonincreasing list $\mathcal{L}$. (Page 12)
$\mathcal{M}_i$	Merge sequence. (Def. 82 on page 59)
$\mathcal{P}_i$	The sequence of proxy sets of $\mathcal{X}[i]$. (Def. 89 on page 62)
$E_R(\mathcal{E})$	Set of retained edges. (Page 17)
G-realized merge graph	A merge graph which is list-isomorphic to a subgraph of G. (Lemma 72 on page 52)
$M_{deg}(\mathcal{E})$	Set of degree-modified vertices. (Page 17)
$M_{list}(\mathcal{E})$	Set of list-modified vertices. (Page 17)
$p(x,j)$	The proxy of $x \in V(\mathcal{X}_i)$ in $\mathcal{P}_i[j]$. (Def. 90 on page 62)

S_A, S_B	Vertex sets of subgraphs involved in merging. (Def. 67 on page 49)
$s_M(b)$	The substitute of the vertex b in $(M; \Lambda_M)$. (Def. 68 on page 50)
$V_+(\mathcal{E})$, $V_-(\mathcal{E})$, $E_+(\mathcal{E})$, and $E_-(\mathcal{E})$	(Def. 14 on page 18)
$V_+(\Phi)$, $V_-(\Phi)$, $E_+(\Phi)$, and $E_-(\Phi)$	(Def. 17 on page 19)
$V_R(\mathcal{E})$	Set of retained vertices. (Page 17)
Origin	(Def. 90 on page 62)
Origin graph	(Def. 92 on page 63)
Proxy	(Def. 90 on page 62)

Abbreviations

gedit	The graph edit operation. (Def. 9 on page 15)
ndledit	The NDL graph edit operation. (Def. 18 on page 20)
$NDL(G)$	Neighbourhood degree list of G. (Def. 3 on page 13)
$NDS(G, v)$	NDS of the vertex v in G. (Def. 2 on page 13)

References

[1] Achuthan, Nirmala. "Characterization of potentially connected integer-pair sequences." Combinatorics and Graph Theory. pp. 153-164. Springer Berlin Heidelberg (1981)

[2] Aho, Alfred V., Hopcroft, John E., and Ullman, Jeffrey D. The design and analysis of computer algorithms, 1974. Reading: Addison-Wesley, pp. 207–209 (1987)

[3] Asano, Takao, and Hirata, Tomio. "Edge-deletion and edge-contraction problems." Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing. pp. 245-254. ACM (1982)

[4] Barrus, Michael D., and Donovan, Elizabeth. "Neighborhood degree lists of graphs." arXiv preprint arXiv:1507.08212 (2015).

[5] Bazgan, Cristina, and Nichterlein, André. "Parameterized inapproximability of degree anonymization." International Symposium on Parameterized and Exact Computation. pp. 75-84. Springer International Publishing (2014)

[6] Bodlaender, Hans L., Tan, Richard B., and van Leeuwen, Jan. "Finding a Δ-regular supergraph of minimum order." Discrete applied mathematics 131.1: 3-9. (2003)

[7] Bose, Prosenjit, and Hurtado, Ferran. "Flips in planar graphs." Computational Geometry 42.1: 60-80. (2009)

[8] Choudum, S. A. "A simple proof of the Erdos-Gallai theorem on graph sequences." Bulletin of the Australian Mathematical Society 33.01: 67-70. (1986)

[9] Casas-Roma, Jordi, Herrera-Joancomartí, Jordi, and Torra, Vicenç. "A summary of k-degree anonymous methods for privacy-preserving on networks." Advanced Research in Data Privacy. pp. 231-250. Springer International Publishing (2015)

[10] Dabrowski, Konrad K., et al. "Editing to a planar graph of given degrees." International Computer Science Symposium in Russia. pp. 143-156. Springer International Publishing (2015)

[11] Das, Prabir. "Characterization of unigraphic and unidigraphic integer-pair sequences." Discrete Mathematics 37.1: 51-66. (1981)

[12] Diestel, Reinhard. "Graph theory. 2005." Grad. Texts in Math 101 (2005).

[13] Downey, Rodney G., and Fellows, Michael R.. Fundamentals of parameterized complexity. Vol. 4. London: Springer (2013)

[14] Eppstein, David. "Diameter and treewidth in minor-closed graph families." Algorithmica 27.3-4: 275-291. (2000)

[15] Ferrara, Michael. "Some problems on graphic sequences." Graph Theory Notes of New York 64: 19-25. (2013)

[16] Frick, Markus, and Grohe, Martin. "Deciding first-order properties of locally tree-decomposable structures." Journal of the ACM (JACM) 48.6: 1184-1206. (2001)

[17] Froese, Vincent, Nichterlein, André, and Niedermeier, Rolf. "Win-win kernelization for degree sequence completion problems." Scandinavian Workshop on Algorithm Theory. pp. 194-205. Springer International Publishing (2014)

[18] Gale, David. "A theorem on flows in networks." Pacific J. Math 7.2: 1073-1082. (1957)

[19] Garey, Michael R., Johnson, David S., and Stockmeyer, Larry. "Some simplified NP-complete problems." Proceedings of the Sixth Annual ACM Symposium on Theory of Computing. pp. 47-63. ACM, (1974)

[20] Gary, Michael R., and Johnson, David S. "Computers and Intractability: A Guide to the Theory of NP-completeness." (1979)

[21] Goldberg, Paul W., et al. "Four strikes against physical mapping of DNA." Journal of Computational Biology 2.1: 139-152. (1995)

[22] Golovach, Petr A., and Mertzios, George B. "Graph Editing to a Given Degree Sequence." International Computer Science Symposium in Russia. pp. 177-191. Springer International Publishing (2016)

[23] Golumbic, Martin Charles, Kaplan, Haim, and Shamir, Ron. "Graph sandwich problems." Journal of Algorithms 19.3: 449-473. (1995)

[24] Hakimi, S. Louis. "On realizability of a set of integers as degrees of the vertices of a linear graph. I." Journal of the Society for Industrial and Applied Mathematics 10.3: 496-506. (1962)

[25] Hammer, P. L., Ibaraki, T., and Simeone, B. "Threshold sequences." SIAM Journal on Algebraic Discrete Methods 2.1: 39-49. (1981)

[26] Harary, Frank. "A survey of the reconstruction conjecture." Graphs and Combinatorics. pp. 18-28. Springer Berlin Heidelberg (1974)

[27] Havel, Václav. "A remark on the existence of finite graphs." Casopis Pest. Mat 80.477-480: 1253. (1955)

[28] Heggernes, Pinar, et al. "A parameterized algorithm for chordal sandwich." International Conference on Algorithms and Complexity. pp. 120-130. Springer Berlin Heidelberg (2010)

[29] van den Heuvel, Jan. "The complexity of change." Surveys in Combinatorics 409: 127-160. (2013)

[30] Hüffner, Falk, et al. "Fixed-parameter algorithms for cluster vertex deletion." Theory of Computing Systems 47.1: 196-217. (2010)

[31] Ito, Takehiro, et al. "On the complexity of reconfiguration problems."Theoretical Computer Science 412.12: 1054-1065. (2011)

[32] Karp, Richard M. "Reducibility among combinatorial problems." Complexity of Computer Computations. Springer US: 85-103. (1972)

[33] Kelly, Paul J. "A congruence theorem for trees." Pacific J. Math 7.1: 961-968. (1957)

[34] Kamiński, Marcin, Medvedev, Paul, and Milanič, Martin. "Complexity of independent set reconfigurability problems." Theoretical computer science 439: 9-15. (2012)

[35] Lewis, John M., and Yannakakis, Mihalis. "The node-deletion problem for hereditary properties is NP-complete." Journal of Computer and System Sciences 20.2: 219-230. (1980)

[36] Li, Shuo-Yen R. “Graphic sequences with unique realization.” Journal of Combinatorial Theory, Series B 19.1: 42-68. (1975)

[37] Liu, Kun, and Terzi, Evimaria . “Towards identity anonymization on graphs.” Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. pp. 93-106. ACM (2008)

[38] Margot, Francois. “Some complexity results about threshold graphs.” Discrete applied mathematics 49.1-3: 299-308. (1994)

[39] Mathieson, Luke, and Szeider, Stefan . “Editing graphs to satisfy degree constraints: A parameterized approach.” Journal of Computer and System Sciences 78.1: 179-191. (2012)

[40] Moser, Hannes, and Thilikos, Dimitrios M. “Parameterized complexity of finding regular induced subgraphs.” Journal of Discrete Algorithms 7.2: 181-190. (2009)

[41] Mouawad, Amer E., Nishimura, Naomi, and Raman, Venkatesh. “Vertex cover reconfiguration and beyond.” International Symposium on Algorithms and Computation. pp. 452-463. Springer International Publishing. (2014)

[42] Natanzon, Assaf. Complexity and approximation of some graph modification problems. Diss. Tel Aviv University. (1999)

[43] Newman, Mark EJ. “Assortative mixing in networks.” Physical Review Letters 89.20: 208701. (2002)

[44] Noldus, Rogier, and Van Mieghem, Piet. “Assortativity in complex networks.” Journal of Complex Networks: cnv005. (2015)

[45] Pastor-Satorras, Romualdo, Vázquez, Alexei, and Vespignani, Alessandro. “Dynamical and correlation properties of the Internet.” Physical Review Letters 87.25: 258701. (2001)

[46] Patrinos, A. N., and Hakimi, S. L. “Relations between graphs and integer-pair sequences.” Discrete Mathematics 15.4: 347-358. (1976)

[47] Plesník, Ján. “A note on the complexity of finding regular subgraphs.” Discrete mathematics 49.2: 161-167. (1984)

[48] Ryser, H. J. “Combinatorial properties of matrices of zeros and ones.” Classic Papers in Combinatorics. pp. 269-275. Birkhäuser Boston (2009)

[49] Senior, James K. “Unimerism.” The Journal of Chemical Physics 19.7: 865-873. (1951)

[50] Stewart, Iain A. “Finding regular subgraphs in both arbitrary and planar graphs.” Discrete Applied Mathematics 68.3: 223-235. (1996)

[51] Tamassia, Roberto, Di Battista, Giuseppe, and Batini, Carlo. “Automatic graph drawing and readability of diagrams.” IEEE Transactions on Systems, Man, and Cybernetics 18.1: 61-79. (1988)

[52] Ulam, Stanislaw M. A collection of mathematical problems. Vol. 8. Interscience Publishers, 1960.

[53] Watanabe, Toshimasa, Ae, Tadashi, and Nakamura, Akira. “On the NP-hardness of edge-deletion and-contraction problems.” Discrete Applied Mathematics 6.1: 63-78. (1983)

[54] Wu, Wentao, et al. “K-symmetry model for identity anonymization in social networks.” Proceedings of the 13th International Conference on Extending Database Technology. pp. 111-122. ACM (2010)

[55] Young, Alfred. “On quantitative substitutional analysis.” Proceedings of the London Mathematical Society 2.1. pp. 196-230. (1932)

[56] Zhou, Bin, and Pei, Jian. “Preserving privacy in social networks against neighborhood attacks.” 2008 IEEE 24th International Conference on Data Engineering. pp. 506-515. IEEE (2008)