Reconfiguration of Subgraphs Satisfying Specific Properties

by
Haruka Mizuta

Submitted to
Department of System Information Sciences
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy (Information Sciences)

Graduate School of Information Sciences
Tohoku University, Japan

2020

Acknowledgements

First of all, I would like to express the deepest appreciation to my supervisor, Professor Xiao Zhou. He always gave me valuable guidances and comments that made my research more sophisticate, and helpful suggestion for my research direction. He also provided a pleasant environment for my research in the laboratory. Without his encouragement, this thesis would not have been materialized.

I would like to express very special thanks to my thesis advisor, Associate Professor Takehiro Ito. He always helped me on the presentations in conferences and on the writing style of my papers. He also gave me a lot of opportunities of cooperative research with researchers in other universities. All experience in these opportunities helped on my growing as a researcher.

I would also like to thank Associate Professor Akira Suzuki for his helpful suggestions and guidance. He took a lot of time to discuss with me, and I learned from him how to proceed with the research. I am also very grateful to his support in my laboratory life.

I would also like to show my special thanks to the other members of my graduate committee, Professor Ayumi Shinohara and Professor Takuo Suganuma for their insightful suggestions and comments.

I am deeply grateful to Tatsuhiko Hatanaka. He had discussion with me many times, and gave me valuable comments and suggestions.

My heartfelt gratitude goes to Professor Naomi Nishimura and her students Benjamin Moore, Vijay Subramanya and Krishna Vaidyanathan at University of Waterloo, and Research Associate Tesshu Hanaka at Chuo University. Without valuable discussions with them, this thesis would not have been completed.

Also, I would like to offer my special thanks to all members of the project of Development of Algorithmic Techniques for Combinatorial Reconfiguration (DATCORE project). I really enjoyed the discussions with them, and they have greatly influenced me.

Finally, I would like to acknowledge with sincere thanks the all-out cooperation and services rendered by the members of Algorithm Theory Laboratory for many things.

Abstract

Combinatorial reconfiguration is one of the well-studied topics in the field of theoretical computer science, which was introduced by Ito et al. in 2008. This topic deals with a "dynamic transformation" of feasible solutions; more specifically, in a (combinatorial) reconfiguration problem, we wish to find a step-by-step transformation between feasible solutions such that all intermediate ones in the transformation are also feasible solutions. In this thesis, we mainly studied reconfiguration problems from the following two viewpoints. The first one is to develop polynomial-time algorithms for reconfiguration problems whose feasible solutions are connected subgraphs. From the previous studies, it is known to be hard to construct an efficient algorithm solving a reconfiguration problem if its feasible solutions are connected subgraphs. We focused on several graph properties which require the connectivity, and analyzed the computational complexity of reconfiguration problems of subgraphs satisfying the property. Throughout this analysis, we developed many polynomial-time algorithms solving the reconfiguration problems with connected subgraphs. The second one is to introduce a new variant of reconfiguration problems, called "optimization variant," in which we do not need to specify a target solution and are asked for a desirable solution that is reachable from an initial solution. As the first example of this variant, we applied this variant to the reconfiguration problem of independent sets which is one of the most well-studied reconfiguration problems. We then analyzed its polynomial-time solvability and parameterized complexity.

Contents

List of Figures

Chapter 1 Introduction

1.1 Combinatorial reconfiguration

Combinatorial reconfiguration [18] is one of the well-studied topics in the field of theoretical computer science. (See surveys [17, 29].) This topic deals with a "dynamic transformation" of feasible states; more specifically, we wish to find a step-by-step transformation between feasible states such that all intermediate ones in the transformation are also feasible states. Combinatorial reconfiguration has several applications such as dynamic puzzle games (e.g. 15 puzzle [20]) and a transition of system states which deliver continuous service.

As an example in the later application, consider a power supply system [18] in which power stations with fixed capacity provide power to customers with fixed demand. (See Figure 1.1.) In this system, each customer has several power stations as candidates of supply source, and is indeed provided power from exactly one of the candidates. Then it is necessary that for each power station, the sum of demands of customers to which the station provides power is at most the capacity of the station; such a supply is called a "feasible supply." As an example, Figure 1.1 illustrates four feasible supplies (a), (b), (c) and (d). We now consider a dynamic transformation of states of the power supply system; we wish to transform a current feasible supply into another feasible supply. To minimize interruption, this transformation needs to be done by repeatedly switching the source power station of a single customer at a time, while keeping feasibility during the transformation. For example, Figure 1.1 illustrates a transformation between the feasible supplies (a) and (d).

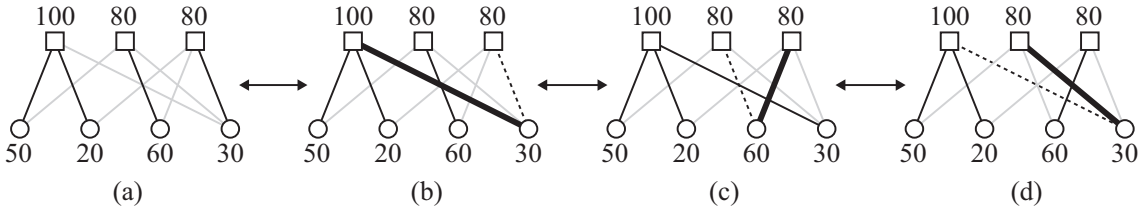


Figure 1.1: An example of a transformation from feasible supplies (a) to (d), where square vertices correspond to power stations, circle vertices to customers, lines to candidates for each customer, and black solid lines to feasible supply.

In general, a transformation of system states consists of the following framework: We need to transform a feasible state into another feasible state by repeatedly applying a prescribed change operation at a time. In 2008, Ito et al. [18] formulated such a dynamic situation as combinatorial reconfiguration. Generally, a (combinatorial) *reconfiguration problem* studies the reachability/connectivity of feasible solutions in a "solution space." A *solution space* is defined as a graph such that the vertex set corresponds to a set of all feasible states, and there is an edge between two feasible states that are "adjacent," according to a prescribed *adjacency relation*. Feasible states in a solution space are often defined as feasible solutions of an instance of a traditional search problem, and hence feasible states are often called *feasible solutions*. Then, a *reachability variant* of a reconfiguration problem asks to determine whether or not there is a path in a solution space from a specified solution (called an *initial solution*) to another specified solution (called a *target solution*). We call such a path a *reconfiguration sequence* between the initial and target solutions, and if it exists, we say that the initial and target solutions are *reachable*. There is another variant of reconfiguration problems, called a *connectivity variant*. This variant asks to determine whether a reconfiguration graph is connected or not, that is, any two feasible solutions are reachable or not. Since the 2000s, several reconfiguration problems are introduced based on traditional search problems and studied its computational complexity; for example, VERTEX COLORING [6, 9, 35], VERTEX COVER [18, 25], INDEPENDENT SET [18, 22], MATCHING [18], SHORTEST PATH [4, 5, 21], CLIQUE [19],

INDUCED TREE [34], and HAMILTONIAN CYCLE [33].

In this thesis, we mainly studied reconfiguration problems from the following two viewpoints. The first one is to develop efficient (polynomial-time) algorithms solving reconfiguration problems whose feasible solutions are connected subgraphs. The second one is to introduce a new variant of reconfiguration problems called "optimization variant," in which we do not need to specify a target solution and are asked for a desirable solution that is reachable from an initial solution.

1.2 Reconfiguration of (connected) subgraphs

In previous studies, many reconfiguration problems take subgraphs as its feasible solutions, and many efficient algorithms were developed for these problems. However, there are few algorithms for reconfiguration problems whose feasible solutions are "connected" subgraphs. We thus see that it is hard to develop an algorithm for reconfiguration problem with connected subgraphs.

In the field of search problems, it is also known that developing an algorithm for problem with connected subgraphs are quite hard. For example, in 1983, Baker [2] gave a general approach for deriving a polynomial-time approximation schemes on planar graphs. His method can be applied for many search problems, however, it sometimes doesn't work for problems with connectivity. Then, in 2007 (after more than twenty years), Borradaile et al. [7] provided a new technique which deriver a polynomial-time approximation schemes on planar graphs for a problem with connectivity. Similar breakthrough was also occurred in the area of parameterized complexity. It had been believed until very recently that there is no algorithm in time faster than $O^*(tw^{tw})$ with respect to the treewidth tw of an input graph for problems with connectivity However, in 2011, Cygan et al. [11] gave an algorithm method which solves a problem with connectivity in time $O^*(c^{tw})$ for some constant c. Such a breakthrough is also desired in the area of combinatorial reconfiguration.

1.2.1 Our problems

In Chapter 3 and 4, we aim to develop efficient (polynomial-time) algorithms for reconfiguration problems that take connected subgraphs as its feasible solutions. We use the term SUBGRAPH RECONFIGURATION to describe a family of reconfiguration problems with (not necessary connected) subgraphs. Each of the individual problems in the family can be defined by specifying a solution space, that is, specifying the vertex set (feasible solutions) and the edge set (adjacency relation) of the solution space. In particular, feasible solutions are defined in terms of a graph structure property Π which subgraphs must satisfy; for example, "a graph is a tree," "a graph is edgeless (an independent set)," and so on. In this thesis, we study the following two problems that belong to SUBGRAPH RECONFIGURATION.

Reconfiguration of fundamental graphs.

In the first problem, we focus on six fundamental graph properties as a property Π; path, cycle, tree, clique, bi-clique, and diameter-two. For each property Π, we consider a reconfiguration problem whose solution space is defined as follows.

Feasible solutions (i.e., vertices in the solution space) are defined as subgraphs satisfying Π. By the choice of how to represent the subgraphs, we introduce three versions; in each version, subgraphs are represented by vertex subsets or edge subsets, as follows.

- **Edge version**: An edge subset induces a subgraph that satisfies Π.
- **Induced version**: A vertex subset induces a subgraph that satisfies Π.
- **Spanning version**: A vertex subset induces a subgraph that contains at least one spanning subgraph satisfying Π.

As an example, Figure 1.2 illustrates several feasible solutions in each version with the property "a graph is a path." In this figure, V_1' is feasible in the spanning

version, while is not feasible in the induced version, because it does not induce a path. As can be seen by this simple example, in the spanning variant, we need to pay attention to the additional complexity of finding a spanning subgraph and the complications resulting from the fact that the subgraph induced by the vertex subset may contain more than one spanning subgraph which satisfies Π.

We then define an adjacency relation (i.e., edges in the solution space). Since we represent a feasible solution by a set of vertices (or edges) in any version, we can consider that tokens are placed on each vertex (resp., edge) in the feasible solution. Then, we mainly deal with the following two well-known adjacency relation [19, 22]:

- **Token-jumping** (TJ, for short): We say that two solutions are *adjacent under* TJ if one can be obtained from the other one by moving a single token to any other vertex (edge) in a given graph.
- **Token-sliding** (TS, for short): We say that two solutions are *adjacent under* TS if one can be obtained from the other one by moving a single token to an adjacent vertex (adjacent edge, that is sharing a common vertex).

As an example, Figure 1.2 shows reconfiguration sequences in each version under TJ. (We note that in Chapter 5, we consider another well-studied adjacency relation, called *token-addition-and-removal* (TAR, for short) [18, 19, 22], where we can add or remove a single token at a time.)

In Chapter 3, we study the reachability variant of reconfiguration problems whose solution spaces are defined above, that is, we are given two feasible solutions and asked to determine whether or not there is a path between them in the solution space.

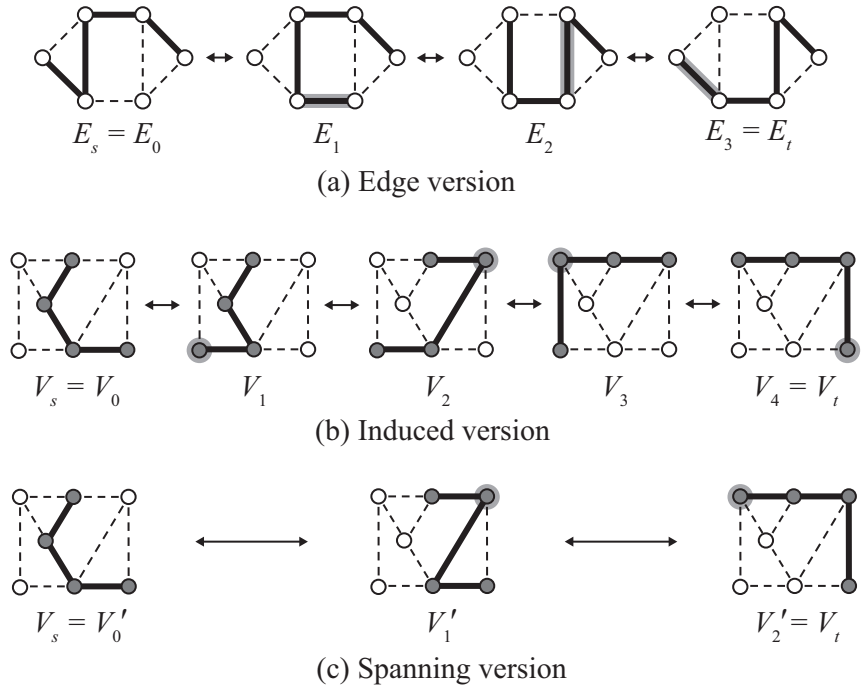


Figure 1.2: Reconfiguration sequences in the three versions under TJ with the property "a graph is a path," where the subgraphs satisfying the property by thick lines and in (b) and (c), vertices forming solutions are depicted by gray circles.

Reconfiguration of Steiner trees.

In the second problem, we focus on the concept of "Steiner tree" as a property Π; note that in the two breakthrough papers for search problems mentioned at the beginning of this section, the search problem of Steiner trees is considered as a typical problem with connectivity. For an unweighted graph G and a vertex subset $S \subseteq V(G)$, a *Steiner tree* of G for S is a subtree of G which includes all vertices in S; each vertex in S is called a *terminal*. For example, Figure 1.3 illustrates four Steiner trees. Then the second problem takes Steiner trees as a property Π. More specifically, we consider reconfiguration problems whose solution spaces are defined, as follows.

Feasible solutions (i.e., vertices in the solution space) are defined as all Steiner trees of a given graph. Note that unlike the previous problem, a feasible solution is a graph itself, not a vertex subset or an edge subset.

We then introduce adjacency relations. In this problem, we consider the following

four adjacency relations:

- **Vertex exchange** (VE, for short): We say that two Steiner trees T and T' of G for S are *adjacent under* VE if there exist two vertices $v \in V(T)$ and $v' \in V(T')$ such that their removal results in the common vertex subset; it is equivalent to the condition $|V(T) \setminus V(T')| = |V(T') \setminus V(T)| \leq 1$.
- **Local vertex exchange** (LVE, for short): We say that two Steiner trees T and T' of G for S are *adjacent under* LVE if there exist two vertices $v \in V(T)$ and $v' \in V(T')$ such that their removal results in the common subgraph of T and T'.
- **Local vertex exchange without changing neighbors** (LVE-N for short): We say that two Steiner trees T and T' of G for S are *adjacent under* LVE-N if there exist two vertices $v \in V(T)$ and $v' \in V(T')$ such that (a) their removal results in the common subgraph of T and T', and (b) the neighborhood of v in T is equal to that of v' in T'.
- **Edge exchange** (EE, for short): We say that two Steiner trees T and T' of G for S are *adjacent under* EE if there exist two edges $e \in E(T)$ and $e' \in E(T')$ such that their removal results in the common edge subset (and hence, common subgraph of T and T'); it is equivalent to the condition $|E(T) \setminus E(T')| = |E(T') \setminus E(T)| \leq 1$.

As an example, in Figure 1.3, any two consecutive Steiner trees are adjacent under VE. However, considering LVE, T_1 and T_2 are not adjacent because removing v_2 from T_1 and v_2' from T_3 result in the different subgraphs. Under LVE-N, only two Steiner trees T_0 and T_1 are adjacent, because v_1 in T_0 has the same neighborhood with v_1' in T_1. Under EE, only two Steiner trees T_2 and T_3 are adjacent, because the subgraph of T_2 obtained by removing e_3 is equal to the subgraph of T_3 obtained by removing e_3'. It should be note that v and v' can be the same vertex in VE, LVE, and LVE-N, and e and e' can be the same edge in EE. (See T_2 and T_3 under VE or LVE in Figure 1.3.) However, under LVE-N and EE, such an exchange must leads $T = T'$.

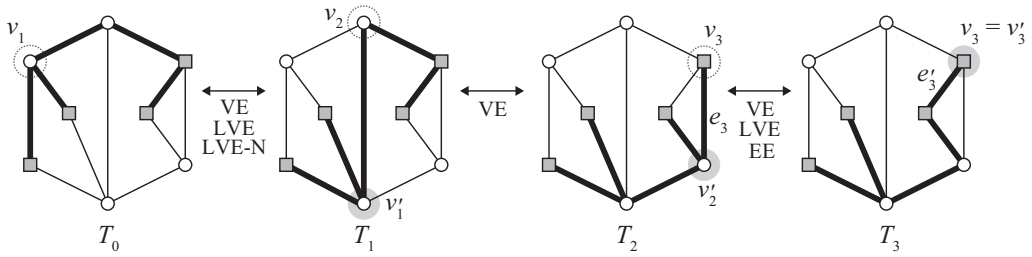


Figure 1.3: A sequence of Steiner trees, where the terminals are depicted by gray squares, non-terminals by white circles, the edges in Steiner trees by thick lines.

In Chapter 4, we study the reachability variant of reconfiguration problems whose feasible solutions are Steiner trees, that is we are given two Steiner trees, and asked to determine whether or not there is a path between them in the solution space defined above.

1.2.2 Known and related results

Although there has been previous work that can be categorized as SUBGRAPH RECONFIGURATION, most of the related results appear under the name of the property Π under consideration. Accordingly, we can view reconfiguration of independent sets [18, 22] as the induced version of SUBGRAPH RECONFIGURATION such that the property Π is "a graph is edgeless." Other examples can be found in Table 1.1. We here explain only known results which are directly related to our contributions.

Reconfiguration of cliques can be seen as both the spanning and the induced version; the problem is PSPACE-complete under any adjacency relation, even when restricted to perfect graphs [19]. Indeed, for this problem, the rules TAR, TJ, and TS have all been shown to be equivalent from the viewpoint of polynomial-time solvability. It is also known that reconfiguration of cliques can be solved in polynomial time for several well-known graph classes [19].

Wasa et al. [34] considered the induced version under the TJ and TS rules with the property Π being "a graph is a tree." They showed that this version under each of the TJ and TS rules is PSPACE-complete, and is also W[1]-hard when parameterized

Table 1.1: Subgraph representations and versions

Representations	Versions	Known reconfiguration problems
edge subset	edge	spanning tree [18] matching [18, 27], and b-matching [27]
vertex subset	induced	clique [19] independent set [18, 22] induced forest [26] induced bipartite [26] induced tree [34] shortest path [4, 5, 21]
	spanning	clique [19] shortest path [4, 5, 21]

by both the size of a solution and the length of a reconfiguration sequence. They also gave a fixed-parameter algorithm when parameterized by both the size of a solution and the maximum degree of an input graph, under both the TJ and TS rules. In closely related work, Mouawad et al. [26] considered the induced versions under the TAR rule with the properties Π being either "a graph is a forest" or "a graph is bipartite." They showed that these versions are W[1]-hard when parameterized by the size of a solution plus the length of a reconfiguration sequence.

Bonsma [4, 5] studied the induced and spanning versions under the TJ rule with the property Π being "a graph is a shortest st-path;" note that these two versions are equivalent for this property because of the shortestness of paths. He showed that the problem is PSPACE-complete even for bipartite graphs [4], while polynomial-time solvable for chordal graphs [4], craw-free graphs [4], and planar graphs [5]. Wrochna [35] also studied this problem and showed that it is PSPACE-complete even for bounded bandwidth graphs. Although the hardness results for bipartite graphs and bounded bandwidth graphs has been shown independently, a simple observation implies that this problem remains PSPACE-complete even for bipartite graphs with bounded bandwidth graph. We should note that this problem is special case of the reconfiguration problem of Steiner trees under three adjacency relations,

VE, LVE, and LVE-N; the reconfiguration problem of Steiner trees is equivalent to that of shortest st-paths if the number of terminals are exactly two and given two Steiner trees have minimum number of edges. Therefore, the hardness results for the reconfiguration problem of shortest st-paths also hold for that of Steiner trees.

1.2.3 Our contribution

We first describe our results for the reconfiguration problems with six fundamental graph properties. (Our results are summarized in Table 1.2, together with known results, where an (i, j)-*biclique* is a complete bipartite graph with the bipartition of i vertices and j vertices.) As mentioned above, since we consider the TJ and TS rules, it suffices to deal with subgraphs having the same number of vertices or edges. Subgraphs of the same size may be isomorphic for certain properties Π, such as "a graph is a path" and "a graph is a clique," because there is only one choice of a path or a clique of a particular size. On the other hand, for the property "a graph is a tree," there are several choices of trees of a particular size. (We will show an example in Section 3.3 with Figure 3.2.)

As shown in Table 1.2, we systematically clarify the complexity of reconfiguration problems for the six fundamental graph properties. In particular, we show that the edge version under TJ is computationally intractable for the property "a graph is a path" but tractable for the property "a graph is a tree." This implies that the computational (in)tractability for various properties Π does not follow directly from the inclusion relationships among the graph classes specified in the properties; one possible explanation is that the path property implies a specific graph, whereas the tree property allows several choices of trees, making the problem easier. To the best of the author's knowledge, entries missing from Table 1.2 (such as the examination of several properties under TS) remain open problems.

We then describe our results for the reconfiguration problem of Steiner trees. As

Table 1.2: Previous and new results

Property Π	**Edge version**	**Induced version**	**Spanning version**
path	NP-hard (TJ) [Thm. 3.2]	PSPACE-c. (TJ, TS) [Thm. 3.7, 3.9]	PSPACE-c. (TJ, TS) [Thm. 3.7, 3.9]
cycle	P (TJ, TS) [Thm. 3.3]	PSPACE-c. (TJ, TS) [Thm. 3.8, 3.9]	PSPACE-c. (TJ, TS) [Thm. 3.8, 3.9]
tree	P (TJ) [Thm. 3.6]	PSPACE-c. (TJ, TS) [34]	P (TJ) PSPACE-c. (TS) [Thm. 3.11, 3.10]
(i, j)**-biclique**	P (TJ, TS) [Thm. 3.5]	PSPACE-c. for $i = j$ (TJ) PSPACE-c. for fixed i (TJ) [Cor. 3.1, Thm. 3.12]	NP-hard for $i = j$ (TJ) P for fixed i (TJ) [Thm. 3.13, 3.14]
clique	P (TJ, TS) [Thm. 3.4]	PSPACE-c. (TJ, TS) [19]	PSPACE-c. (TJ, TS) [19]
diameter two		PSPACE-c. (TS) [Thm. 3.15]	PSPACE-c. (TS) [Thm. 3.15]
any property	XP for solution size (TJ, TS) [Thm. 3.1]	XP for solution size (TJ, TS) [Thm. 3.1]	XP for solution size (TJ, TS) [Thm. 3.1]

we mentioned, the problems under VE, LVE and LVE-N are generalizations of the reconfiguration problem of shortest *st*-paths. Following the problem of shortest *st*-paths, we studied the problems of Steiner trees with respect to graph classes. We first showed that the problem under each of VE, LVE and EE is PSPACE-complete even for split graphs and planar graphs, while polynomial-time solvable for cographs, interval graphs and cactus graphs. (See Figure 1.4.) We further studied the special case where given two Steiner trees have minimum number of edges. Then we showed that the problem under EE is polynomial-time solvable for any graph, while that under VE and LVE has the same computational complexity with the general (not necessary minimum) case. For this special case, we also studied under LVE-N. Specifically, we showed that the reconfiguration problem of minimum Steiner trees under LVE-N is polynomial-time solvable for chordal graphs and planar graphs. (See Figure 1.5.)

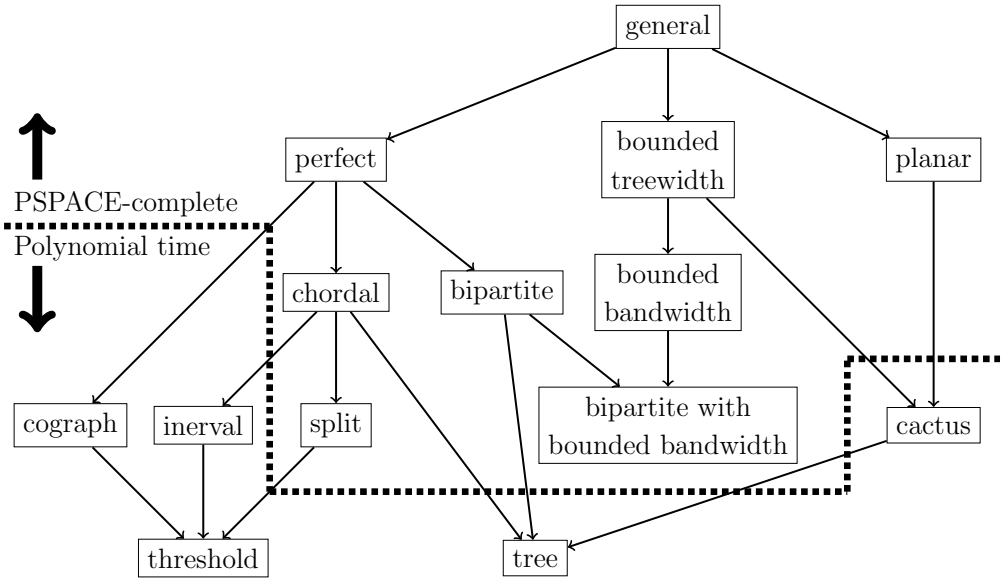


Figure 1.4: Our results for the reconfiguration problem of Steiner trees under VE, LVE and EE, where each square corresponds to a graph class, and $A \to B$ means that A contains B as a subclass.

By considering several properties with connectivity, we proposed several polynomial-time algorithms solving reconfiguration problems whose feasible solutions are connected subgraphs.

1.3 New variant of reconfiguration problems

In the reachability variant of reconfiguration problems, we need to specify a target solution. However, there may exist (possibly, exponentially many) desirable solutions (candidates of a target solution); even if we cannot reach a given target solution from the initial solution, there may exist another desirable solution which is reachable.

1.3.1 Our problem

In this thesis, we propose a new variant of reconfiguration problems in which we are only given an initial solution and asked to find more desirable solution that is reachable from the initial solution. We call this variant the *optimization variant*. As

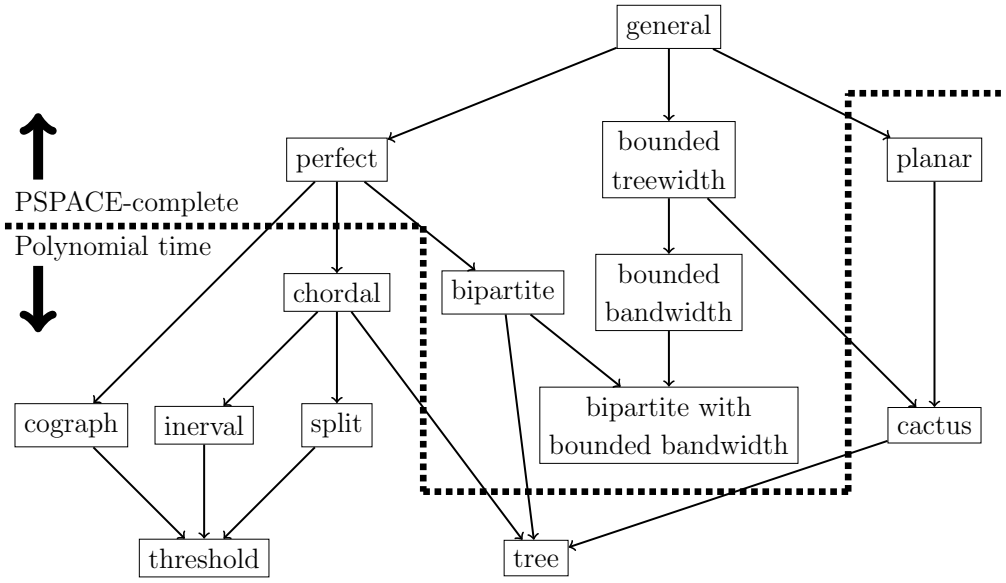


Figure 1.5: Our results for the reconfiguration problem of minimum Steiner trees under LVE-N, where each square corresponds to a graph class, and $A \to B$ means that A contains B as a subclass.

the first example of this variant, we consider INDEPENDENT SET RECONFIGURATION (the reconfiguration problem of independent sets) under TAR rule [18, 22] because it is one of the most well-studied reconfiguration problems.

For a graph G, an *independent set* of G is a vertex subset $I \subseteq V(G)$ in which no two vertices are adjacent. Then, INDEPENDENT SET RECONFIGURATION is a reconfiguration problem whose reconfiguration graph is defined as follows: The vertex set is a set of all independent sets whose cardinalities are at least a given number l, and two independent sets are adjacent if the size of symmetric difference of them is exactly one; this adjacency relation is one of the well-studied relation, called*tokne-addtion-and-removal* (TAR, for short) [18, 19, 22]. Figure 1.6 illustrates an example of reconfiguration sequence between two independent sets I_0 and I_3 under TAR for the lower bound $l = 1$.

Then we consider the optimization variant of INDEPENDENT SET RECONFIGURATION; for simple notation, we denoted by OPT-ISR the optimization variant of INDEPENDENT SET RECONFIGURATION and by REACH-ISR the original reachabil-

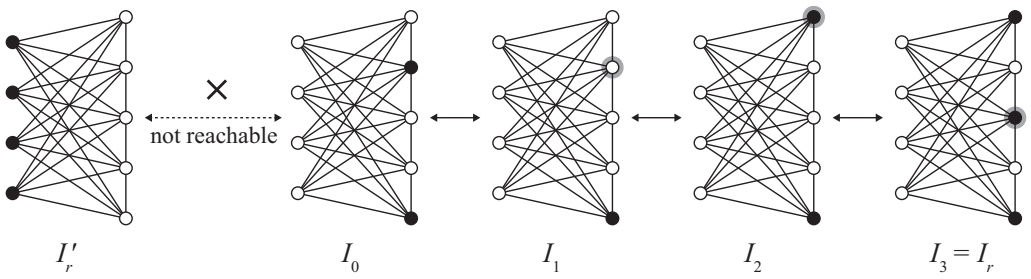


Figure 1.6: A sequence $\langle I_0, I_1, I_2, I_3 \rangle$ of independent sets under TAR for the lower bound $l = 1$, where the vertices in independent sets are colored with black.

ity variant of INDEPENDENT SET RECONFIGURATION. In OPT-ISR, we are given a single solution I_0 (an independent set of cardinality at least l), and asked to find another solution I_{sol} (an independent set) such that it can be reachable from the initial solution and the cardinality is maximized; we call such a solution I_{sol} an *output solution*.

Note that I_{sol} is not always a maximum independent set of an input graph G. For example, the graph in Figure 1.6 has a unique maximum independent set I'_r of size four (consisting of the vertices on the left side), but I_0 cannot be transformed into it. Indeed, one of output solutions is I_3 for this example when $l = 1$.

In Chapter 5, we study OPT-ISR from the viewpoints of polynomial-time solvability and fixed-parameter (in)tractability.

1.3.2 Known and related results

Although OPT-ISR is being introduced in this thesis, some previous results for REACH-ISR are related in the sense that they can be converted into results for OPT-ISR. We present such results here.

Ito et al. [18] showed that REACH-ISR under TAR is PSPACE-complete. On the other hand, Kamiński et al. [22] proved that any two independent sets of size at least $l+1$ are reachable under TAR with the lower bound l for even-hole-free graphs.

REACH-ISR has been studied well from the viewpoint of fixed-parameter

(in)tractability. Mouawad et al. [26] showed that Reach-ISR under TAR is W[1]-hard when parameterized by the lower bound l and the length of a desired sequence (i.e., the number of token additions and removals). Lokshtanov et al. [24] gave a fixed-parameter algorithm to solve Reach-ISR under TAR when parameterized by the lower bound l and the degeneracy d of an input graph.

From our problem setting, one may be reminded of the concept of local search algorithms [1, 30]. To the best of our knowledge, known results for this concept do not have direct relations to our problem, because they are usually evaluated experimentally. In addition, note that our problem assumes that an initial independent set I_0 is given as an input. In contrast, a local search algorithm is allowed to choose the initial solution, sometimes randomly.

1.3.3 Our contribution

We first study the polynomial-time solvability of Opt-ISR with respect to graph classes, as summarized in Figure 1.7. More specifically, we show that Opt-ISR is PSPACE-hard even for bounded pathwidth graphs, and remains NP-hard even for planar graphs. On the other hand, we give a linear-time algorithm to solve the problem for chordal graphs. We note that our algorithm indeed works in polynomial time for even-hole-free graphs (which form a larger graph class than that of chordal graphs) if the problem of finding a maximum independent set is solvable in polynomial time for even-hole-free graphs; currently, its complexity status is unknown.

We next study the fixed-parameter (in)tractability of Opt-ISR, as summarized in Table 1.3. In this paper, we consider mainly the following three parameters: the degeneracy d of an input graph, a lower bound l on the size of independent sets, and the size s of an output solution reachable from a given independent set I_0. As shown in Table 1.3, we completely analyze the fixed-parameter (in)tractability of the problem according to these three parameters; details are explained below.

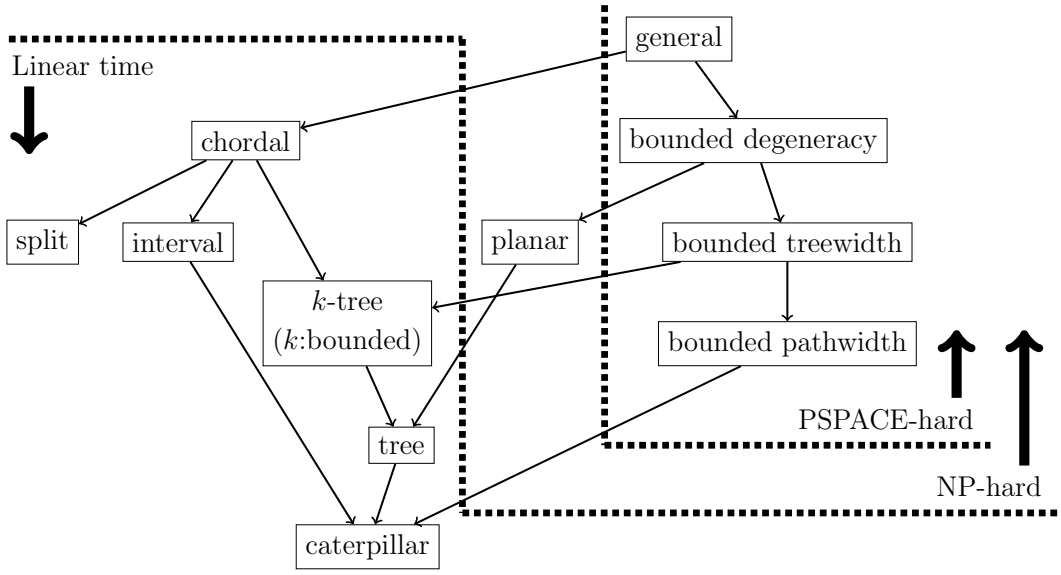


Figure 1.7: Our results for Opt-ISR, where each square corresponds to a graph class, and $A \to B$ means that A contains B as a subclass.

We first consider the problem parameterized by a single parameter. We show that the problem is fixed-parameter intractable when only one of d, l, and s is taken as a parameter. In particular, we prove that Opt-ISR is PSPACE-hard for a fixed constant d and remains NP-hard for a fixed constant l, and hence the problem does not admit even an XP algorithm for each single parameter d or l under the assumption that P $\neq$ PSPACE or P $\neq$ NP. On the other hand, Opt-ISR is W[1]-hard for s, and admits an XP algorithm with respect to s.

We thus consider the problem taking two parameters. However, the problem still remains NP-hard for a fixed constant $d+l$, and hence it does not admit even an XP algorithm for d+l under the assumption that P $\neq$ NP. Note that the combination of l and s is meaningless, since we can assume without loss of generality that $l+s \leq 2s$. On the other hand, we give a fixed-parameter algorithm when parameterized by $s+d$; this result implies that Opt-ISR parameterized only by s is fixed-parameter tractable for planar graphs, and for bounded treewidth graphs.

Table 1.3: Our results for Opt-ISR with respect to parameters.

	(no parameter)	lower bound l	solution size s
(no parameter)	—	NP-h. for fixed l (i.e., no FPT, no XP) [Corollary 5.1]	W[1]-h., **XP** (i.e., no FPT) [Theorems 5.4, 5.5]
degeneracy d	PSPACE-h. for fixed d (i.e., no FPT, no XP) [Theorem 5.2]	NP-h. for fixed $d + l$ (i.e., no FPT, no XP) [Corollary 5.1]	**FPT** [Theorem 5.6]

Chapter 2 Preliminaries

2.1 Graph-theoretical terminologies

In this section, we give the basic graph-theoretical terminologies [8, 10].

2.1.1 Graphs and subgraphs

A *graph* is a ordered pair (V, E), where V is a set of *vertices* and $E \subseteq V \times V$ is a set of *edges*; we simply denote by uv the edge (u, v). We sometimes denote by $V(G)$ and $E(G)$ the vertex set and the edge set of G, respectively. A graph G is called *simple* if G does not contain an edge $e = vv$ for all vertices $v \in V(G)$ and more than two edges $e = uv$ for any pair of two vertices $u, v \in V(G)$. A graph G is called *undirected* if $uv \in E(G)$ if and only if $vu \in E(G)$; in this case, we identify uv and vu. In this thesis, we consider simple undirected graphs unless otherwise stated.

For an edge $e = uv$, we say that u and v are *adjacent*; in this case, u is *neighbor of* v, and symmetrically v is neighbor of u. We call each of u and v an *endpoint* of e, and we say that e is *incident to* u and v. For a vertex v, the (*open*) *neighborhood* of v in G, denoted by $N_G(v)$, is the set of all neighbors of v in G, and the *closed neighborhood* of v in G, denoted by $N_G[v]$, is defined as $N_G(v) \cup \{v\}$.

For a graph $G = (V, E)$, a *subgraph* of G is a graph $G' = (V', E')$ such that both $V' \subseteq V$ and $E' \subseteq E$ hold. In particular, it is a *spanning subgraph* of G if $V' = V$ holds. For a graph $G = (V, E)$ and a vertex subset $S \subseteq V$, a subgraph *induced by* S is the subgraph $G' = (V', E')$ where $V' = S$ and $E' = \{uv \in E \mid u, v \in S\}$; we say that S *induces* G'. On the other hand, for an edge subset $U \subseteq E$, a subgraph

induced by U is the subgraph $G' = (V', E')$ where $V' = \bigcup_{uv \in U} \{u, v\}$ and $E' = U$; we say that U *induces* G'. For a vertex subset $S \subseteq V$ (resp. an edge subset $U \subseteq E$), we denote by $G[S]$ (resp. $G[U]$) the subgraph of G induced by S (resp. induced by U).

2.1.2 Path, cycle, and connectivity

Let $G = (V, E)$ be a graph. For two vertices $u, v \in V$, a *walk from* u *to* v is a sequence of vertices $\mathcal{W} = \langle u = v_0, v_1, \ldots, v_\ell = v \rangle$ such that there is an edge $v_i v_{i+1}$ for each $i \in \{0, 1, \ldots, \ell - 1\}$. ℓ is the *length* of the walk $\mathcal{W}$. A walk $\mathcal{W}$ is called *path* if all vertices $v_0, v_1, \ldots, v_\ell$ in $\mathcal{W}$ are distinct. On the other hand, a walk $\mathcal{W}$ of length at least three is called *cycle* if $v_0 = v_\ell$ and vertices $v_0, v_1, \ldots, v_{\ell-1}$ are distinct. We sometimes consider a walk (similarly, path and cycle) to be a graph whose vertex set is $\{v_0, v_1, \ldots, v_\ell\}$ and edge set is $\{v_i v_{i+1} \mid i \in \{0, 1, \ldots, \ell - 1\}\}$.

For a graph G and two vertices $u, v \in V$, the *distance between* u *and* v is the minimum ℓ such that there exists a path between u and v of length ℓ; we denote by $\mathsf{dist}_G(u, v)$ the distance between u and v on G. If there is no path between u and v, we consider that the distance between u and v is infinite. G is *connected* if there is a path between every pair of two vertices in G, and is *disconnected* otherwise. For a connected graph G, a vertex $v \in V(G)$ is called a *cut-vertex* if $G[V(G) \setminus \{v\}]$ is disconnected. For a connected graph G, the *diameter* of G is the longest distance among all pairs of two vertices in G, that is, $\max_{u,v \in V(G)} \mathsf{dist}_G(u, v)$.

2.1.3 Independent set and clique

For a graph $G = (V, E)$, a vertex subset $I \subseteq V$ is an *independent set* if no two vertices are adjacent. On the other hand, a vertex subset $C \subseteq V$ is a *clique* if any two vertices are adjacent.

2.1.4 Graph classes and parameters

In this subsection, we give the definitions of graph classes and graph parameters which we deal with in this thesis. We first list the definitions of graph classes in the following.

- **Tree**: A graph is *tree* if it is connected and contains no cycle as a subgraph.
- **Cactus**: A graph is *cactus* if every edge is part of at most one cycle.
- **Planar**: A graph is *planar* if it can be embedded in the plane without crossing edges.
- **Complete**: A graph is *complete* if any two vertices in the graph are adjacent.
- **Split**: A graph G is *split* if its vertex set can be partitioned into an independent set $I \subseteq V(G)$ and a clique $C = V(G) \setminus I$.
- **Bipartite**: A graph G is *bipartite* if its vertex set can be partitioned into two independent sets $I_1 \subseteq V(G)$ and $I_2 = V(G) \setminus I_1$. In particular, it is *complete bipartite* or *biclique* if any pair of vertices $v_1 \in I_1$ and $v_2 \in I_2$ are adjacent.
- **Cograph**: A graph G is *cograph* if it contains no path of four vertices as an induced subgraph.
- **Chordal**: A graph is *chordal* if it contains no cycle of more than four vertices as an induced subgraph.
- **Interval**: A graph G with $V(G) = \{v_1, v_2, \ldots, v_n\}$ is *interval* if there exists a set $\mathcal{I}$ of (closed) intervals $I_1, I_2, \ldots, I_n$ such that $v_i v_j \in E(G)$ if and only if $I_i \cap I_j = \emptyset$ for each $i, j \in \{1, 2, \ldots, n\}$. We call the set $\mathcal{I}$ of intervals an *interval representation* of G.

We then list the definitions of graph parameters.

- **Treewidth**: For a graph G, a *tree-decomposition* of G is a tree $T = (\mathcal{B}, F)$ such that $\mathcal{B} = \{B_1, B_2, \ldots, B_k\}$ is the family of vertex subsets of G (each vertex subset in $\mathcal{B}$ is called a *bag*) and T satisfies the following three conditions:

(a) each vertex $v \in V(G)$ belongs to at least one bag, that is $\bigcup_{B \in \mathcal{B}} B = V(G)$;

(b) for each edge $uv \in E(G)$, there exists at least one bag which contains both u and v; and

(c) for each vertex $v \in V(G)$, all bags containing v forms a subtree of T.

The *width* of a tree-decomposition is defined as $\max_i |B_i| + 1$. The *treewidth* of G is the minimum t such that G has a tree-decomposition of width t.

- **Pathwidth**: For a graph G, a tree-decomposition T is also called a *path-decomposition* if T is a path. The *pathwidth* of G is the minimum p such that G has a path-decomposition of width p.
- **Bandwidth**: For a graph G, the *bandwidth* of G is the minimum b such that there is an injection $i : V(G) \to \{1, 2, \ldots, |V|\}$ such that $\max_{uv \in E(G)} |i(u) - i(v)| \leq b$.
- **Degeneracy**: For an integer $d \geq 0$, a graph G is called a *d-degenerate* graph if every induced subgraphs of G has a vertex of degree at most d [23]. The *degeneracy* of G is the minimum integer d such that G is a d-degenerate graph.

2.2 Algorithm-theoretical terminologies

In this section, we give the basic algorithm-theoretical terminologies [10, 28].

2.2.1 Problems and reductions

An *abstract problem* is defined to be a binary relation on a set $\mathcal{I}$ of problem *instances* and a set $\mathcal{S}$ of problem *solutions*. Note that an instance may have more than one solution. An abstract problem is called a *decision problem* if each instance has a solution either "yes" or "no" solution, that is, problem solutions $\mathcal{S}$ consists of $\{0, 1\}$. Many abstract problems are not decision problems, but rather *optimization problems*, which require some value to be minimized or maximized. For example, the reachability variant of a reconfiguration problem is a decision problem, while the optimization variant is not a decision problem but an optimization problem. To

solve an abstract problem by computers, we must represent the problem as a binary string. Thus we sometimes assume that an abstract problem is represented as a binary string. Then we say that an algorithm *solves* an abstract problem in time $O(T(n))$ if it can output a solution for any instance of length n in time $O(T(n))$. An abstract problem is *polynomial-time solvable* if there exists an algorithm solving the problem in time $O(n^k)$ for some constant k; such an algorithm is called a *polynomial-time* algorithm.

Let $\mathcal{P}, \mathcal{P}'$ be decision problems represented as binary strings. A *polynomial-time reduction* from $\mathcal{P}'$ to $\mathcal{P}$ is an algorithm which computes an instance I of $\mathcal{P}$ for any given instance I' of $\mathcal{P}'$ such that;

- the solution of I in $\mathcal{P}$ is "yes" if and only if that of I' in $\mathcal{P}'$ is "yes;" and
- the algorithm runs in time $O(n^k)$, where $n = |I'|$ is the length of the instance I and k is some constant.

An abstract problem *parameterized by* p is defined to be a binary relation on a set $\mathcal{I}_p$ of problem instances and a set $\mathcal{S}$ of problem solutions, where $\mathcal{I}_p$ is a set of pairs consist of an instance of an abstract problem and a natural number, called *parameter*; we call an abstract problem parameterized by some parameter a *parameterized problem*. Then an abstract problem parameterized by p is *fixed-parameter tractable* if there exists an algorithm which can be output a solution for any instance (I, p) in time $O(f(p) \cdot n^k)$, where f is some computable function depending only on p, $n = |(I, p)|$ is the length of the instance (I, p), and k is some constant; we call such an algorithm a *fixed-parameter* algorithm or an *FPT algorithm*. On the other hand, we call an algorithm an *XP algorithm* if it runs in time $O(n^{f(k)})$ where f is an arbitrary function depending only on k.

Let $\mathcal{P}$ and $\mathcal{P}'$ be decision problems parameterized by p and p', respectively. Then, a *parameterized reduction* (or an *FPT reduction*) is an algorithm which computes an instance (I, p) of $\mathcal{P}$ for any given instance (I', p') of $\mathcal{P}'$ such that;

- the solution of (I, p) in $\mathcal{P}$ is "yes" if and only if that of (I', p') in $\mathcal{P}'$ is "yes;"
- $p \leq g(p')$ holds for some computable function g depending only on p'; and
- the algorithm runs in time $O(f(p') \cdot n^k)$, where f is some computable function depending only on p', $n = |(I', p')|$ is the length of the instance (I', p'), and k is some constant.

2.2.2 Class of computational complexity

In this subsection, we describe some classes of problems with respect to computational complexity.

NP is the class of all decision problems that can be verified by a polynomial-time algorithm. An abstract problem $\mathcal{P}$ is *NP-hard* if for any problem $\mathcal{P}'$ in NP, there is a polynomial-time reduction from $\mathcal{P}'$ to $\mathcal{P}$. A decision problem $\mathcal{P}$ is *NP-complete* if it is NP-hard and in NP. Note that there is no polynomial-time algorithm solving an NP-hard problem, unless P = NP.

PSPACE is the class of all abstract problems that can be solved in polynomial space. An abstract problem $\mathcal{P}$ is *PSPACE-hard* if for any problem $\mathcal{P}'$ in PSPACE, there is a polynomial-time reduction from $\mathcal{P}'$ to $\mathcal{P}$. An abstract problem $\mathcal{P}$ is *PSPACE-complete* if it is PSPACE-hard and in PSPACE. Note that there is no polynomial-time algorithm solving a PSPACE-hard problem, unless P = PSPACE.

XP and *FPT* is the class of all parameterized problems that can be solved by an XP algorithm and an FPT algorithm, respectively. Note that FPT is a subclass of XP. *W[1]* is the class of all parameterized problems such that for any problem $\mathcal{P}$ in W[1], there is a parameterized reduction from it to WEIGHTED 2-CNF-SATISFIABILITY. A parameterized problem $\mathcal{P}$ is *W[1]-hard* if for any problem $\mathcal{P}'$ in W[1], there is a parameterized reduction from $\mathcal{P}'$ to $\mathcal{P}$. Note that there is no FPT algorithm solving a W[1]-hard problem, unless FPT = W[1].

Chapter 3 Reconfiguration of fundamental graphs

In this chapter, we study the complexity of the reachability variant of reconfiguration problems with several graph properties.

3.1 Definition of problems and preliminaries

We formally define the reconfiguration problems of subgraphs satisfying a fundamental property; the property subgraphs must satisfy is specified as property Π. Let G be an input graph. Feasible solutions are defined as subgraphs satisfying Π. More specifically, we consider three versions according to how to represent subgraphs; in each version, a feasible solution is defined, as follows:

- **Edge version**: an edge subset $U \subseteq E(G)$ such that $G[U]$ satisfies Π.
- **Induced version**: a vertex subset $S \subseteq V(G)$ such that $G[S]$ satisfies Π.
- **Spanning version**: a vertex subset $S \subseteq V(G)$ such that $G[S]$ contains at least one spanning subgraph satisfying Π.

Then, we consider the following two adjacency relations:

- **Token-jumping** (TJ, for short): Tow feasible solutions S, S' (edge subsets in the edge version, and vertex subsets in the induced and spanning version) are *adjacent under* TJ if $|S \setminus S'| = |S' \setminus S| = 1$ holds.
- **Token-sliding** (TJ, for short): In the edge version, two feasible solutions $U, U' \subseteq E(G)$ are *adjacent under* TS if $|U \setminus U'| = |U' \setminus U| = 1$ holds and the unique edges $e \in U \setminus U'$ and $e' \in U' \setminus U$ have a common end point in G. In the induced and

spanning versions, two feasible solutions $S, S' \subseteq V(G)$ are *adjacent under* TJ if $|S \setminus S'| = |S' \setminus S| = 1$ holds and the unique vertices $v \in S \setminus S'$ and $v' \in S' \setminus S$ are adjacent in G.

Then, we deal with the reachability variant of reconfiguration problems whose solution spaces are defined above. More specifically, for a property Π and an adjacency relation $\mathcal{R} \in \{\mathsf{TJ}, \mathsf{TS}\}$, we consider the following problem: We are given a graph G and two feasible solutions S_0 and S^r in the solution space, then asked to determine whether or not there exists a reconfiguration sequence between S and S' in the solution space.

We denote by (G, V_0, V_r) an instance of a spanning version or an induced version whose input graph is G and source and target solutions are vertex subsets V_0 and V_r of G. Similarly, we denote by (G, E_0, E_r) an instance of the edge version. We may assume without loss of generality that $|V_0| = |V_r|$ holds for the spanning and induced versions, and $|E_0| = |E_r|$ holds for the edge versions; otherwise, the answer is clearly no since under both the TJ and TS rules, all solutions must be of the same size.

3.2 General XP algorithm

In this section, we give a general XP algorithm when the size of a solution (that is, the size of a vertex or edge subset that represents a subgraph) is taken as the parameter. For notational convenience, we simply use *element* to represent a vertex (or an edge) for the spanning and induced versions (resp., the edge version), and *candidate* to represent a set of elements (which does not necessarily satisfy the property Π). Furthermore, we define the *size* of a given graph as the number of elements in the graph.

Theorem 3.1 *Let* Π *be any graph structure property such that we can check if a candidate of size* k *satisfies* Π *in* $f(k)$ *time, where* $f(k)$ *is a computable function*

depending only on k. Then, all of the spanning, induced, and edge versions under TJ *or* TS *can be solved in time $O(n^{2k}k + n^k f(k))$, where n is the size of a given graph and k is the size of a source (and target) solution. Furthermore, a shortest reconfiguration sequence between source and target solutions can be found in the same time bound, if it exists.*

Proof. Our claim is that the reconfiguration graph can be constructed in the stated time. Since a given source solution is of size k, it suffices to deal only with candidates of size exactly k. For a given graph, the total number of possible candidates of size k is $O(n^k)$. For each candidate, we can check in time $f(k)$ whether it satisfies Π. Therefore, we can construct the node set of the reconfiguration graph in time $O(n^k f(k))$. We then obtain the edge set of the reconfiguration graph. Since there are $O(n^k)$ nodes in the reconfiguration graph, the number of pairs of nodes is $O(n^{2k})$. Since each node corresponds to a set of k elements, we can check if two nodes are adjacent or not in $O(k)$ time. Therefore, we can find all pairs of adjacent nodes in time $O(n^{2k}k)$.

In this way, we can construct the reconfiguration graph in time $O(n^{2k}k+n^k f(k))$ in total. The reconfiguration graph consists of $O(n^k)$ nodes and $O(n^{2k})$ edges. Therefore, by breadth-first search starting from the node representing a given source solution, we can determine in time $O(n^{2k})$ whether or not there exists a reconfiguration sequence between two nodes representing the source and target solutions. Notice that if a desired reconfiguration sequence exists, then breadth-first search finds a shortest one. □

3.3 Edge versions

In this section, we study the edge version for five different properties, namely those specifying that the graph be a path, a cycle, a clique, a biclique, or a tree.

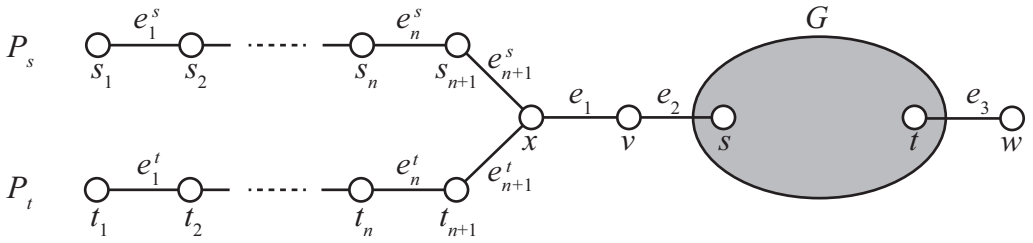


Figure 3.1: Reduction to the edge version under TJ for the property "a graph is a path."

We first consider the property "a graph is a path" under TJ.

Theorem 3.2 *The edge version under* TJ *is NP-hard for the property "a graph is a path."*

Proof. We give a polynomial-time reduction from the Hamiltonian path problem. Recall that a *Hamiltonian path* in a graph G is a path that visits each vertex of G exactly once. Given a graph G and two vertices $s, t \in V(G)$ of G, the NP-complete problem Hamiltonian path is to determine whether or not G has a Hamiltonian path which starts from s and ends in t [14].

For an instance (G, s, t) of Hamiltonian path, we construct a corresponding instance (G', E_s, E_t) of our problem, as follows. (See also Figure 3.1.) Let $n = |V(G)|$. We first add two new vertices v and x to G with two new edges $e_1 = xv$ and $e_2 = vs$. We then add two paths $P_s = \langle s_1, s_2, \ldots, s_{n+1}, x\rangle$ and $P_t = \langle t_1, t_2, \ldots, t_{n+1}, x\rangle$, where $s_1, s_2, \ldots, s_{n+1}$ and $t_1, t_2, \ldots, t_{n+1}$ are distinct new vertices. Each of P_s and P_t consists of $n+1$ edges; we denote by $e_1^s, e_2^s, \ldots, e_{n+1}^s$ the edges $s_1s_2,\ s_2s_3,\ \ldots, s_{n+1}x$ in P_s, respectively, and by $e_1^t, e_2^t, \ldots, e_{n+1}^t$ the edges $t_1t_2,\ t_2t_3,\ \ldots, t_{n+1}x$ in P_t, respectively. We finally add a new vertex w with an edge $e_3 = tw$, completing the construction of G'. We then set $E_s = \{e_1^s, e_2^s, \ldots, e_{n+1}^s, e_1, e_2\}$ and $E_t = \{e_1^t, e_2^t, \ldots, e_{n+1}^t, e_1, e_2\}$; these edge subsets clearly form paths in G'. We have thus constructed our corresponding instance (G', E_s, E_t) in polynomial time.

We now prove that an instance (G, s, t) of Hamiltonian path is a yes-instance

if and only if the corresponding instance (G', E_s, E_t) is a yes-instance.

To prove the only-if direction, we first suppose that G has a Hamiltonian path P starting from s and ending in t. Then, we construct an actual reconfiguration sequence from E_s to E_t using the edges in P. Notice that P consists of $n-1$ edges. Thus, we first move the $n-1$ edges $e_1^s, e_2^s, \ldots, e_{n-1}^s$ in E_s to the edges in P one by one, and then move e_n^s to e_3. Next, we move e_{n+1}^s to e_{n+1}^t, and then move the edges in $E(P) \cup \{e_3\}$ to $e_n^t, e_{n-1}^t, \ldots, e_1^t$ one by one. By the construction of G', we know that each of the intermediate edge subsets forms a path in G', as required.

We now prove the if direction by supposing that there exists a reconfiguration sequence $\langle E_s = E_0, E_1, \ldots, E_\ell = E_t \rangle$. Let E_q be the first edge subset in the sequence such that $E(P_t) \cap E_q \neq \emptyset$; we claim that E_q contains a Hamiltonian path in G. First, notice that the edge in $E(P_t) \cap E_q$ is e_{n+1}^t; otherwise the subgraph formed by E_q is disconnected. Since $|E_q| = |E_s| = n+3$ and $|E(P_s)| = n+1$, we can observe that E_q contains no edge in P_s; otherwise the degree of x would be three, or E_q would form a disconnected subgraph. Therefore, the $n+2$ edges in $E_q \setminus \{e_{n+1}^t\}$ must be chosen from $E(G) \cup \{e_1, e_2, e_3\}$. Since $|V(G)| = n$ and E_q must form a path in G', we know that $E_q \setminus \{e_{n+1}^t\}$ consists of e_1, e_2, e_3 and $n-1$ edges in G. Thus, $E_q \setminus \{e_{n+1}^t, e_1, e_2, e_3\}$ forms a Hamiltonian path in G starting from s and ending in t, as required. □

We now consider the property "a graph is a cycle," as follows.

Theorem 3.3 *The edge version under each of* TJ *and* TS *can be solved in linear time for the property "a graph is a cycle."*

Proof. Let (G, E_s, E_t) be a given instance. We claim that the reconfiguration graph is edgeless, in other words, no feasible solution can be transformed at all. Then, the answer is yes if and only if $E_s = E_t$ holds; this condition can be checked in linear time.

Let E' be any feasible solution of G, and consider a replacement of an edge $e^- \in E'$ with an edge e^+ other than e^-. Let u, v be the endpoints of e^-. When we remove

e^- from E', the resulting edge subset $E' \setminus \{e\}$ forms a path whose ends are u and v. Then, to ensure that the candidate forms a cycle, we can choose only $e^- = uv$ as e^+. This contradicts the assumption that $e^+ \neq e^-$. □

The same arguments partially hold for the property "a graph is a clique," and we obtain the following theorem. We note that, for this property, both induced and spanning versions (i.e., when solutions are represented by vertex subsets) are PSPACE-complete under any adjacency relation [19].

Theorem 3.4 *The edge version under each of* TJ *and* TS *can be solved in linear time for the property "a graph is a clique."*

Proof. Suppose that (G, E_s, E_t) be a given instance. If the size of the clique (the number of vertices of the subgraph) formed by E_s (and E_t) is at least three, then the same arguments in the proof of Theorem 3.3 hold, and hence the instance can be solved in linear time. We thus consider the other case where E_s and E_t form 2-cliques. In this case, we know $|E_s| = |E_t| = 1$. Then, under TJ, we observe that it is always a yes-instance, since we can move the unique edge in E_s into that in E_t directly. On the other hand, under TS, we observe that it is a yes-instance if and only if E_s and E_t are contained in the same connected component of G. Therefore, we can conclude that this case is also solvable in linear time. □

We next consider the property "a graph is an (i, j)-biclique," as follows.

Theorem 3.5 *The edge version under each of* TJ *and* TS *can be solved in polynomial time for the property "a graph is an (i, j)-biclique" for any pair of positive integers i and j.*

Proof. We may assume without loss of generality that $i \leq j$ holds. We prove the theorem in the following three cases:

- **Case 1:** $i = 1$ and $j \leq 2$;
- **Case 2:** $i, j \geq 2$; and

- Case 3: $i = 1$ and $j \geq 3$.

We first consider **Case 1**, which is the easiest case. In this case, any $(1, j)$-biclique has at most two edges. Therefore, by Theorem 3.1 we can conclude that this case is solvable in polynomial time.

We then consider **Case 2**. We show that (G, E_s, E_t) is a yes-instance if and only if $E_s = E_t$ holds. To do so, we claim that the reconfiguration graph is edgeless, in other words, no feasible solution can be transformed at all. To see this, because $i, j \geq 2$, notice that the removal of any edge e in an (i, j)-biclique results in a bipartite graph with the same bipartition of i vertices and j vertices. Therefore, to obtain an (i, j)-biclique by adding a single edge, we must add back the same edge e.

We finally deal with **Case 3**. Notice that a $(1, j)$-biclique is a star with j leaves, and its center vertex is of degree $j \geq 3$. Then, we claim that (G, E_s, E_t) is a yes-instance if and only if the center vertices of stars represented by E_s and E_t are the same. The if direction clearly holds, because we can always move edges in $E_s \setminus E_t$ into ones in $E_t \setminus E_s$ one by one. We thus prove the only-if direction; indeed, we prove the contrapositive, that is, the answer is no if the center vertices of stars represented by E_s and E_t are different. Consider such a star T_s formed by E_s. Since T_s has j (≥ 3) leaves, the removal of any edge in E_s results in a star having $j - 1$ (≥ 2) leaves. Therefore, to ensure that each intermediate solution is a star with j leaves, we can add only an edge of G which is incident to the center of T_s. Thus, we cannot change the center vertex. □

In this way, we have proved Theorem 3.5. Although **Case 1** takes non-linear time, our algorithm can be easily improved under TJ so that it runs in linear time; in **Case 1**, a subgraph forms a $(1, j)$-biclique if and only if it forms a tree, and hence we can solve in linear time by Theorem 3.6 (presented later).

We finally consider the property "a graph is a tree" under TJ. As we have mentioned in the introduction, for this property, there are several choices of trees even

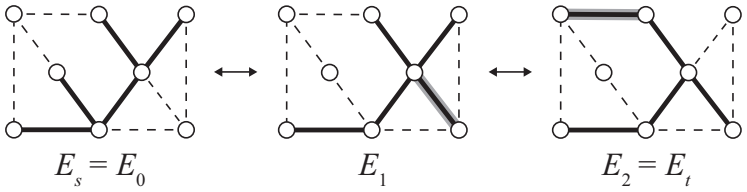


Figure 3.2: Reconfiguration sequence $\langle E_0, E_1, E_2 \rangle$ in the edge version under TJ with the property "a graph is a tree."

of a particular size, and a reconfiguration sequence does not necessarily consist of isomorphic trees (see Figure 3.2). This "flexibility" of subgraphs may yield the contrast between Theorem 3.2 for the path property and the following theorem for the tree property.

Theorem 3.6 *The edge version under* TJ *can be solved in linear time for the property "a graph is a tree."*

Proof. Suppose that (G, E_s, E_t) is a given instance. We may assume without loss of generality that $|E_s| = |E_t| \geq 2$; otherwise $|E_s| = |E_t| \leq 1$ holds, and hence the instance is trivially a yes-instance. We will prove below that any instance with $|E_s| = |E_t| \geq 2$ is a yes-instance if and only if all the edges in E_s and E_t are contained in the same connected component of G. Note that this condition can be checked in linear time.

We first prove the only-if direction of our claim. Since $|E_s| = |E_t| \geq 2$ and subgraphs always must retain a tree structure (more specifically, they must be connected graphs), observe that we can exchange edges only in the same connected component of G. Thus, the only-if direction follows.

To complete the proof, it suffices to prove the if direction of our claim. For notational convenience, for any feasible solution E_i we denote by T_i the tree represented by E_i, and by V_i the vertex set of T_i. In this direction, we consider the following two cases: (a) $V_s \cap V_t = \emptyset$, and (b) $V_s \cap V_t \neq \emptyset$.

We consider Case (a), that is, $V_s \cap V_t = \emptyset$. Since T_s and T_t are contained in one connected component of G, there exists a path $\langle v_0, v_1, \ldots, v_\ell \rangle$ in G such that $v_0 \in V_s$,

$v_\ell \in V_t$ and $v_i \notin V_s \cup V_t$ for all $i \in \{1, 2, \ldots, \ell - 1\}$. Since T_s is a tree, it has at least two degree-one vertices. Let v_s be any degree-one vertex in $V_s \setminus \{v_0\}$, and let e_s be the leaf edge of T_s incident to v_s. Then, we can exchange e_s with v_0v_1, and obtain another tree represented by the resulting edge subset $(E_s \cup \{v_0v_1\}) \setminus \{e_s\}$. By repeatedly applying this operation along the path $\langle v_1, v_2, \ldots, v_\ell \rangle$, we can obtain a solution E_k such that $V_k \cap V_t = \{v_\ell\} \neq \emptyset$; this case will be considered below.

We finally consider Case (b), that is, $V_s \cap V_t \neq \emptyset$. Consider the graph $(V_s \cap V_t, E_s \cap E_t)$. Then, $(V_s \cap V_t, E_s \cap E_t)$ is a forest, and let $G' = (V', E')$ be a connected component (i.e., a tree) of $(V_s \cap V_t, E_s \cap E_t)$ whose edge set is of maximum size. We now prove that there is a reconfiguration sequence between E_s and E_t by induction on $k = |E_s \setminus E'| = |E_t \setminus E'|$. If $k = 0$, then $E_s = E' = E_t$ and hence the claim holds. We thus consider the case where $k > 0$ holds. Since G' is a proper subtree of T_t, there exists at least one edge e_t in $E_t \setminus E'$ such that one endpoint of e_t is contained in V' and the other is not. We claim that there exists an edge e_s in $E_s \setminus E'$ which can be moved into e_t, that is, the subgraph represented by the resulting edge subset $(E_s \cup \{e_t\}) \setminus \{e_s\}$ forms a tree. If both endpoints of e_t are contained in V_s (not just V'), $E_s \cup \{e_t\}$ contains a cycle; let $C \subseteq E_s \cup \{e_t\}$ be the edge set of the cycle. Since the subgraph T_t has no cycle, there exists at least one edge in $C \setminus E_t$, and we choose one of them as e_s. On the other hand, if just one endpoint of e_t is contained in V_s, then we choose a leaf edge of T_s in $E_s \setminus E'$ as e_s. Note that there exists such a leaf edge since G' is a proper subtree of T_s. From the choice of e_s and e_t, we know the subgraph represented by the resulting edge subset $(E_s \cup \{e_t\}) \setminus \{e_s\}$ forms a tree; let $E_k = (E_s \cup \{e_t\}) \setminus \{e_s\}$. Furthermore, since $E_k \cap E_t$ includes $E' \cup \{e_t\}$ and the subgraph formed by $E' \cup \{e_t\}$ is connected, the subgraph formed by $E_k \cap E_t$ has a connected component whose edge set has size at least $|E'| + 1$. Therefore, we can conclude that E_k is reconfigurable into E_t by the induction hypothesis. □

3.4 Induced and spanning versions

In this section, we deal with the induced and spanning versions where subgraphs are represented as vertex subsets. Most of our results for these versions are hardness results, except for Theorems 3.11 and 3.14.

3.4.1 Path and cycle

In this subsection, we show that both induced and spanning versions under TJ or TS are PSPACE-complete for the properties "a graph is a path" and "a graph is a cycle." All proofs in this subsection make use of reductions that employ almost identical constructions. Therefore, we describe the detailed proof for only one case, and give proof sketches for the other cases.

We give polynomial-time reductions from the SHORTEST PATH RECONFIGURATION problem [4, 5, 21]. For a simple, unweighted, and undirected graph G and two distinct vertices s, t of G, SHORTEST PATH RECONFIGURATION is defined as the induced (or spanning) version under TJ for the property "a graph is a shortest st-path." As we mentioned in the introduction, there is no difference between the induced version and the spanning version for this property, because any shortest path in a simple graph forms an induced subgraph. This problem is known to be PSPACE-complete [4].

Let d be the (shortest) distance from s to t in G. For each $i \in \{0, 1, \ldots, d\}$, we denote by $L_i \subseteq V(G)$ the set of vertices that lie on a shortest st-path at distance i from s. Therefore, we have $L_0 = \{s\}$ and $L_d = \{t\}$. We call each L_i a *layer*. Observe that any shortest st-path contains exactly one vertex from each layer, and we can assume without loss of generality that G has no vertex which does not belong to any layer.

We first give the following theorem.

Theorem 3.7 *For the property "a graph is a path," the induced and spanning versions under* TJ *are both PSPACE-complete on bipartite graphs.*

Proof. Observe that these versions are in PSPACE. Therefore, we construct a polynomial-time reduction from SHORTEST PATH RECONFIGURATION.

Let (G, V_s, V_t) be an instance of SHORTEST PATH RECONFIGURATION. Since any shortest st-path contains exactly one vertex from each layer, we can assume without loss of generality that G has no edge joining two vertices in the same layer, that is, each layer L_i forms an independent set in G. Then, G is a bipartite graph. From (G, V_s, V_t), we construct a corresponding instance (G', V_s', V_t') for the induced and spanning versions; note that we use the same reduction for both versions. Let G' be the graph obtained from G by adding four new vertices s_1, s_2, t_1, t_2 which are connected with four new edges s_2s_1, s_1s, tt_1, t_1t_2. Note that G' is also bipartite. We then set $V_s' = V_s \cup \{s_1, s_2, t_1, t_2\}$ and $V_t' = V_t \cup \{s_1, s_2, t_1, t_2\}$. Since each of V_s and V_t induces a shortest st-path in G, each of V_s' and V_t' is a feasible solution to both versions. This completes the polynomial-time construction of the corresponding instance.

Consider any vertex subset $V' \subseteq V(G')$ that satisfies the following conditions (a) and (b);

(a) $s_2, s_1, s, t, t_1, t_2 \in V'$; and

(b) V' contains exactly one vertex from each layer of G.

Note that V' is not necessarily a feasible solution. Then, these conditions ensure that $V' \setminus \{s_2, s_1, t_1, t_2\}$ forms a shortest st-path in G if and only if the subgraph represented by V' induces a path in G'. Thus, we finally prove the following claim, which ensures that an instance (G, V_s, V_t) of SHORTEST PATH RECONFIGURATION is a yes-instance if and only if the corresponding instance (G', V_s', V_t') of the induced or spanning version is a yes-instance.

Claim *Let $V' \subseteq V(G')$ be any solution for the induced or spanning version which is*

reachable by a reconfiguration sequence from V'_s (or V'_t) under TJ*. Then, V' satisfies Conditions (a) and (b).*

We first prove that Condition (a) is satisfied. Notice that both V'_s and V'_t satisfy this condition. Because V' is reconfigurable from V'_s or V'_t, it suffices to show that we cannot move any of s_2, s_1, s, t, t_1, t_2 in a reconfiguration sequence starting from V'_s or V'_t. Suppose for sake of contradiction that we can move at least one of s_2, s_1, s, t, t_1, t_2. Then, the first removed vertex $v \in \{s_2, s_1, s, t, t_1, t_2\}$ must be either s_2 or t_2; otherwise the resulting subgraph would be disconnected. Let w be the vertex with which we exchanged v. Then, $w \in V(G') \setminus \{s_2, s_1, s, t, t_1, t_2\} = V(G) \setminus \{s, t\}$. Therefore, the resulting vertex subset cannot induce a path, and hence it cannot be a solution for the induced version. Similarly, the induced subgraph cannot contain a spanning path, and hence it cannot be a solution for the spanning version.

We next show that Condition (b) is satisfied. Recall that d denotes the number of edges in a shortest st-path in G. Then, we have $|V'| = |V'_s| = |V'_t| = d + 5$. By Condition (a), we know that V' contains s_2, s_1, t_1, t_2, and hence in both induced and spanning versions, s and t must be connected by a path formed by $d + 1$ vertices in $V' \setminus \{s_2, s_1, t_1, t_2\}$. Since the length of this st-path is d, this path is shortest and hence V' must contain exactly one vertex from each of $d + 1$ layers of G. □

Similar arguments give the following theorem.

Theorem 3.8 *Both the induced and spanning versions under* TJ *are PSPACE-complete for the property "a graph is a cycle."*

Proof. Our reduction is the same as in the proof of Theorem 3.7 except for the following point: instead of adding four new vertices, we connect s and t by a path of length three with two new vertices s_1 and t_1. Then, the same arguments hold as in the proof of Theorem 3.7. □

We now consider TS. Notice that, in the proofs of Theorems 3.7 and 3.8, we

exchange only vertices contained in the same layer. Since any shortest st-path in a graph G contains exactly one vertex from each layer, we can assume without loss of generality that each layer L_i of G forms a clique. Then, the same reductions work for TS, and we obtain the following theorem.

Theorem 3.9 *Both the induced and spanning versions of under* TS *are PSPACE-complete for the properties "a graph is a path" and "a graph is a cycle."*

3.4.2 Tree

Wasa et al. [34] showed that the induced version under TJ and TS is PSPACE-complete for the property "a graph is a tree." In this subsection, we show that the spanning version for this property is also PSPACE-complete under TS, while it is linear-time solvable under TJ.

We first note that our proof of Theorem 3.9 yields the following theorem.

Theorem 3.10 *The spanning version of under* TS *is PSPACE-complete for the property "a graph is a tree."*

Proof. We claim that the same reduction as in Theorem 3.9 applies. Let $V' \subseteq V(G')$ be any solution which is reachable by a reconfiguration sequence from V'_s (or V'_t) under TS, where (G', V'_s, V'_t) is the corresponding instance for the spanning version, as in the reduction. Then, TS ensures that $s_2, s_1, s, t, t_1, t_2 \in V'$ holds, and V' contains exactly one vertex from each layer of G. Therefore, any solution forms a path even for the property "a graph is a tree," and hence the theorem follows. □

In contrast to Theorem 3.10, the spanning version under TJ is solvable in linear time. We note that the reduction in Theorem 3.10 does not work under TJ, because the tokens on s_2 and t_2 can move (jump) and hence there is no guarantee that a solution forms a path for the property "a graph is a tree."

Theorem 3.11 *The spanning version of under* TJ *can be solved in linear time for the property "a graph is a tree."*

Proof. Suppose that (G, V_s, V_t) is a given instance. We assume that $|V_s| = |V_t| \geq 2$ holds; otherwise it is a trivial instance. Then, Theorem 3.11 can be obtained from the following claim.

Claim (G, V_s, V_t) *with* $|V_s| = |V_t| \geq 2$ *is a* yes*-instance if and only if* V_s *and* V_t *are contained in the same connected component of* G*.*

We first prove the only-if direction of the claim. Notice that the property requires subgraphs to be connected. Because $|V_s| = |V_t| \geq 2$ and we exchange only one vertex at a time, we can exchange a vertex only with another vertex in the same connected component. Therefore, V_s and V_t are contained in the same connected component of G if (G, V_s, V_t) is a yes-instance.

Next, we prove the if direction of the claim for the case where $|V_s| = |V_t| = 2$. In this case, $G[V_s]$ and $G[V_t]$ consist of single edges, say e_s and e_t, respectively. Since V_s and V_t are contained in the same connected component of G, there is a path in G between e_s and e_t. Thus, we can exchange vertices along the path, and obtain a reconfiguration sequence from V_s to V_t. In this way, the if direction holds for this case.

Finally, we prove the if direction of the claim for the remaining case, that is, $|V_s| = |V_t| \geq 3$. Consider any spanning trees T_s of $G[V_s]$ and T_t of $G[V_t]$. Since $|V_s| = |V_t| \geq 3$, each of T_s and T_t has at least two edges. Then, if we regard $(G, E(T_s), E(T_t))$ as an instance of the edge version under TJ for the property "a graph is a tree," we know from the proof of Theorem 3.6 that it is a yes-instance. Thus, there exists a reconfiguration sequence $\mathcal{E} = \langle E(T_s) = E_0, E_1, \ldots, E_\ell = E(T_t) \rangle$ of edge subsets under TJ. We below show that, based on $\mathcal{E}$, we can construct a reconfiguration sequence between V_s and V_t for the spanning version under TJ.

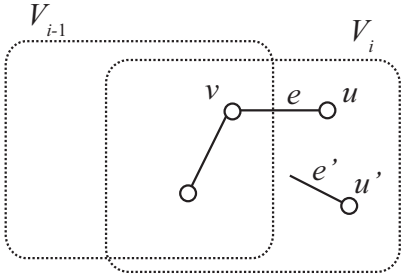


Figure 3.3: Illustration for the proof of Theorem 3.11.

For each E_i in $\mathcal{E}$, let V_i be the vertex set of the tree represented by E_i. Notice that V_i is a feasible solution for the spanning version, and that $V_0 = V_s$ and $V_\ell = V_t$ hold. We claim that the sequence $\langle V_s = V_0, V_1, \ldots, V_\ell = V_t \rangle$ of vertex subsets is a reconfiguration sequence for the spanning version under TJ (after removing redundant vertex subsets if needed). To show this, it suffices to prove $|V_i \setminus V_{i-1}| = |V_{i-1} \setminus V_i| \leq 1$ for all $i \in \{1, 2, \ldots, \ell\}$. Suppose for the sake of contradiction that there exists V_i such that $|V_i \setminus V_{i-1}| \geq 2$ holds. (See also Figure 3.3.) Since $|E_i| \geq 2$ and $|E_i \setminus E_{i-1}| = 1$ hold, we have $E_i \cap E_{i-1} \neq \emptyset$ and hence $V_i \cap V_{i-1} \neq \emptyset$. Then there is at least one edge $e = uv$ in $E_i \setminus E_{i-1}$ joining a vertex $u \in V_i \setminus V_{i-1}$ and $v \in V_i \cap V_{i-1}$, because E_i must form a connected subgraph. Since $|V_i \setminus V_{i-1}| \geq 2$, there is another vertex $u' \neq u$ in $V_i \setminus V_{i-1}$, and there is an edge e' incident to u'. Note that $e \neq e'$. Furthermore, we know that $e' \in E_i \setminus E_{i-1}$ because $u' \in V_i \setminus V_{i-1}$. Therefore, we have $e, e' \in E_i \setminus E_{i-1}$, which contradicts the fact that $|E_i \setminus E_{i-1}| = 1$ holds. □

3.4.3 Biclique

For the property "a graph is an (i, j)-biclique," we show that the induced version under TJ is PSPACE-complete even if $i = j$ holds, or i is fixed. On the other hand, the spanning version under TJ is NP-hard even if $i = j$ holds, while it is polynomial-time solvable when i is fixed.

We first give the following theorem for a fixed $i \geq 1$.

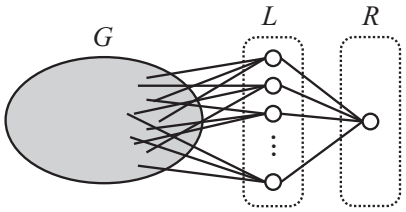


Figure 3.4: Reduction for the property "a graph is an (i, j)-biclique" for any fixed $i \geq 1$.

Theorem 3.12 *For the property "a graph is an (i, j)-biclique," the induced version under* TJ *is PSPACE-complete even for any fixed integer $i \geq 1$.*

Proof. We give a polynomial-time reduction from the MAXIMUM INDEPENDENT SET RECONFIGURATION problem [35]. The MAXIMUM INDEPENDENT SET RECONFIGURATION problem is defined as the induced version for the property "a graph is edgeless" such that two given independent sets are maximum. This problem is known to be PSPACE-complete under TJ and TS [35].

Suppose that (G, V_s, V_t) is an instance of MAXIMUM INDEPENDENT SET RECONFIGURATION. We now construct a corresponding instance (G', V_s', V_t') of the induced version under TJ for the property "a graph is an (i, j)-biclique," where i is any fixed positive integer. (See also Figure 3.4.) Let L and R be distinct sets of new vertices such that $|L| = i$ and $|R| = 1$. The vertex set of G' is defined as $V(G') = V(G) \cup L \cup R$, and the edge set of G' as $E(G') = E(G) \cup \{uv \mid u \in V(G), v \in L\} \cup \{vw \mid v \in L, w \in R\}$, that is, new edges are added so that there are edges between each vertex of L and each vertex of $V(G) \cup R$. Let $V_s' = V_s \cup L \cup R$ and $V_t' = V_t \cup L \cup R$. Since L, R, V_s and V_t are all independent sets in G', both V_s' and V_t' form (i, j)-bicliques, where $i = |L|$ and $j = |V_s \cup R| = |V_t \cup R|$. We have now completed the construction of our corresponding instance, which can be accomplished in polynomial time.

We first show the claim that for any feasible solution $V' \subseteq V(G')$ which is reconfigurable from V_s' or V_t' under TJ, the vertex subset $V' \cap V(G)$ forms a maximum

independent set of G. Because V' induces a bipartite graph (more specifically, an (i, j)-biclique) and each vertex in L is connected to all vertices in $V(G)$, at least one of the following conditions holds; (a) $V' \cap L = \emptyset$; and (b) $V' \cap V(G)$ forms an independent set. However, we can see that (a) does not hold, as follows. We know that all vertices in $L \cup R$ are initially contained in V'_s and V'_t. Consider exchanging $u \in L \cup R$ with $v \in V(G)$; if $|L| = 1$ and $u \in L$ then the resulting subgraph would be disconnected, and otherwise both Conditions (a) and (b) would be violated since $V_s \cap V(G)$ and $V_t \cap V(G)$ are maximum independent sets of G. Therefore, we cannot exchange any vertex in $L \cup R$ with a vertex in $V(G)$, and hence the claim holds.

On the other hand, we observe by the construction of G' that if a vertex subset $V'' \subseteq V(G)$ forms a (maximum) independent set of G, then $V'' \cup L \cup R$ induces an (i, j)-biclique in G'.

By the above discussion, we can conclude that an instance (G, V_s, V_t) of MAXIMUM INDEPENDENT SET RECONFIGURATION is a yes-instance if and only if our corresponding instance (G', V'_s, V'_t) is a yes-instance. □

The corresponding instance (G', V'_s, V'_t) constructed in the proof of Theorem 3.12 satisfies $i = j$ if we set $i = |V_s| + 1 = |V_t| + 1$. Therefore, we can obtain the following corollary.

Corollary 3.1 *For the property "a graph is an (i, j)-biclique," the induced version under* TJ *is PSPACE-complete even if $i = j$ holds.*

We next give the following theorem.

Theorem 3.13 *For the property "a graph is an (i, j)-biclique," the spanning version under* TJ *is NP-hard even if $i = j$ holds.*

Proof. We give a polynomial-time reduction from the BALANCED COMPLETE BIPARTITE SUBGRAPH problem, defined as follows [14]. Given a bipartite graph G and a positive integer k, the BALANCED COMPLETE BIPARTITE SUBGRAPH problem

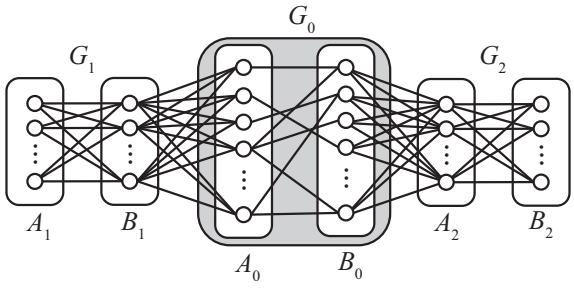


Figure 3.5: Reduction for the property "a graph is a (k, k)-biclique."

is to determine whether or not G contains a (k, k)-biclique as a subgraph; this problem is known to be NP-hard [14].

Suppose that (G, k) is an instance of BALANCED COMPLETE BIPARTITE SUBGRAPH, where G is a bipartite graph. Then, we construct a corresponding instance (G', V_s, V_t) of the spanning version under TJ for the property "a graph is a (k, k)-biclique." We first construct a graph G'. (See Figure 3.5.) Let G_0 be a graph which is isomorphic to G and (A_0, B_0) be the bipartition of G_0. We add to G_0 two new (k, k)-bicliques G_1 and G_2; let (A_1, B_1) be the bipartition of G_1, and (A_2, B_2) be that of G_2. We then add edges between any two vertices $x \in B_1$ and $y \in A_0$, and between any two vertices $x \in B_0$ and $y \in A_2$. Therefore, $G'[B_1 \cup A_0]$ and $G'[B_0 \cup A_2]$ are bicliques in G'. This completes the construction of G'. We then set $V_s = V(G_1)$ and $V_t = V(G_2)$. Then, V_s and V_t are solutions, since $G'[V_s]$ and $G'[V_t]$ contain (k, k)-bicliques G_1 and G_2, respectively. In this way, the corresponding instance can be constructed in polynomial time.

By the construction of G', any reconfiguration sequence between $V_s = V(G_1)$ and $V_t = V(G_2)$ must pass through a (k, k)-biclique of G_0. Therefore, the theorem follows. □

We now give a polynomial-time algorithm solving the spanning version for a fixed constant $i \geq 1$.

Theorem 3.14 *For the property "a graph is an (i, j)-biclique," the spanning version*

under TJ *is solvable in polynomial time when* $i \geq 1$ *is a fixed constant.*

We give such an algorithm as a proof of Theorem 3.14. We will refer to the i vertices in the bounded-size part of the biclique as *hubs*, and the j vertices in the other part as *terminals*. Let $H \subseteq V(G)$ be an arbitrary vertex subset such that $|H| = i$. We denote by $C(H) \subseteq V(G)$ the set of all common neighbors of H in G, i.e., $C(H) = \bigcap_{v \in H}\{u \in V(G) \mid uv \in E(G)\}$. We write $C[H] = C(H) \cup H$. We denote by $\mathcal{S}(H)$ the set of all solutions that contain (i, j)-bicliques with the hub set H. We know that $\mathcal{S}(H) \neq \emptyset$ if and only if $|C(H)| \geq j$ holds; if $|C(H)| \geq j$, then $H \cup T$ is in $\mathcal{S}(H)$ for any subset $T \subseteq C(H)$ such that $|T| = j$. We also observe that $W \subseteq C[H]$ holds for any solution $W \in \mathcal{S}(H)$. It should be noted that a solution in the spanning version is simply a vertex subset V' of $V(G)$, and there is no restriction on how to choose a hub set from V'. (For example, if a solution V' induces a clique of size five, then there are ten ways to choose a hub set from V' for $(2, 3)$-bicliques.) Therefore, $\mathcal{S}(H) \cap \mathcal{S}(H') \neq \emptyset$ may hold for distinct hub sets H, H'.

We describe two key observations in the following. The first one is that for a hub set H, any two solutions $W, W' \in \mathcal{S}(H)$ are reconfigurable because we can always move vertices in $W \setminus W'$ into ones in $W' \setminus W$ one by one. The second one is that for any two distinct hub sets H_a and H_b, if there exist $V_a \in \mathcal{S}(H_a)$ and $V_b \in \mathcal{S}(H_b)$ such that $|V_a \setminus V_b| = |V_b \setminus V_a| \leq 1$ (this means that V_a and V_b are reconfigurable by one reconfiguration step, or $V_a = V_b$), then all pairs of solutions in $\mathcal{S}(H_a) \cup \mathcal{S}(H_b)$ are reconfigurable.

Based on these observations, we construct an *auxiliary graph* A for a given instance (G, V_s, V_t), as follows. Each node in A corresponds to a set H of i vertices (hubs) in the input graph G such that $|C(H)| \geq j$; we represent a node in A simply by the corresponding hub set H. Two nodes H_a and H_b are adjacent in A if there exist $V_a \in \mathcal{S}(H_a)$ and $V_b \in \mathcal{S}(H_b)$ such that $|V_a \setminus V_b| = |V_b \setminus V_a| \leq 1$. We first prove the following lemma.

Lemma 3.1 *Let H_s and H_t be any two nodes in A such that $V_s \in \mathcal{S}(H_s)$ and $V_t \in \mathcal{S}(H_t)$, respectively. Then, there is a reconfiguration sequence between V_s and V_t if and only if there is a path in A between H_s and H_t.*

Proof. We first suppose that there is a path $\mathcal{P} = \langle H_s = H_0, H_1, \ldots, H_{\ell'} = H_t \rangle$ in A between H_s and H_t. We know that any two consecutive nodes H_i and H_{i+1} in $\mathcal{P}$ are adjacent in A. Then, as we mentioned above, all pairs of solutions in $\mathcal{S}(H_i) \cup \mathcal{S}(H_{i+1})$ are reconfigurable. Since $V_s \in \mathcal{S}(H_s)$ and $V_t \in \mathcal{S}(H_t)$, we conclude that there is a reconfiguration sequence between V_s and V_t.

We now suppose that there exists a reconfiguration sequence $\mathcal{R} = \langle V_s = V_0, V_1, \ldots, V_\ell = V_t \rangle$ between V_s and V_t. For each solution V_i in $\mathcal{R}$ except for V_s and V_t, we choose an arbitrary node H_i in A which satisfies $V_i \in \mathcal{S}(H_i)$. Consider any two consecutive solutions V_i and V_{i+1} in $\mathcal{R}$. Then, by the construction of A, the chosen nodes H_i and H_{i+1} are adjacent in A (or sometimes $H_i = H_{i+1}$) because $|V_i \setminus V_{i+1}| = |V_{i+1} \setminus V_i| = 1$. In this way, we can ensure the existence of a desired path in A. □

Our algorithm first constructs an auxiliary graph A, and checks whether or not there is a path between H_s and H_t; the correctness of the algorithm follows directly from Lemma 3.1. However, it is not so obvious how to construct the auxiliary graph A in the desired running time. To this end, we give the following lemma.

Lemma 3.2 *Any two nodes H_a and H_b in A are joined by an edge in A if and only if all the following four conditions hold*:

(a) $|C[H_a] \cap C[H_b]| \geq i + j - 1$;

(b) $|H_a \setminus C[H_b]| \leq 1$;

(c) $|H_b \setminus C[H_a]| \leq 1$; *and*

(d) $|H_a \cup H_b| \leq i + j + 1$.

Proof. Suppose that two nodes H_a and H_b in A are joined by an edge in A.

Then, there exist two solutions $V_a \in \mathcal{S}(H_a)$ and $V_b \in \mathcal{S}(H_b)$ such that $|V_a \setminus V_b| = |V_b \setminus V_a| \leq 1$. Let $V' = V_a \cap V_b$. Notice that we have $V' \subseteq C[H_a] \cap C[H_b]$, because both $V_a \subseteq C[H_a]$ and $V_b \subseteq C[H_b]$ hold. Then, Condition (a) holds because $|V'| \geq |V_a| - 1 = i + j - 1$. Furthermore, we observe that $|H_a \setminus C[H_b]| \leq |H_a \setminus V'| \leq |V_a \setminus V'| \leq 1$, and hence we obtain Condition (b). Similarly, Condition (c) holds, too. Finally, Condition (d) holds, because we observe that $H_a \cup H_b \subseteq V_a \cup V_b$ and $|V_a \cup V_b| \leq i + j + 1$. In this way, we have all the four conditions.

Conversely, suppose that all four conditions hold. We first prove the following claim, and then use it to complete the proof of the if direction.

Claim *There exists a vertex subset $V' \subseteq C[H_a] \cap C[H_b]$ of size $i + j - 1$ such that $|H_a \setminus V'| \leq 1$ and $|H_b \setminus V'| \leq 1$.*

Proof of the claim. By Condition (b) we know that at most one vertex in H_a is not contained in $C[H_b]$. Let $v_a \in H_a \setminus C[H_b]$ if $|H_a \setminus C[H_b]| = 1$, and otherwise v_a is any vertex in $H_a \setminus H_b$; recall that $H_a \neq H_b$ and $|H_a| = |H_b| = i$, and hence $H_a \setminus H_b \neq \emptyset$. Let $H'_a = H_a \setminus \{v_a\}$. Then, $H'_a \subseteq C[H_b]$. Furthermore, since $H'_a \subseteq H_a \subseteq C[H_a]$, we have $H'_a \subseteq C[H_a] \cap C[H_b]$. Similarly, let $v_b \in H_b \setminus C[H_a]$ if $|H_b \setminus C[H_a]| = 1$, and otherwise v_b is any vertex in $H_b \setminus H_a$. Let $H'_b = H_b \setminus \{v_b\}$; then we have $H'_b \subseteq C[H_a] \cap C[H_b]$. Therefore, $H'_a \cup H'_b \subseteq C[H_a] \cap C[H_b]$, and by Condition (d) we have $|H'_a \cup H'_b| \leq i + j - 1$. Thus, by Condition (a) we can choose a vertex subset $V' \subseteq C[H_a] \cap C[H_b]$ of size exactly $i + j - 1$ which contains all vertices in $H'_a \cup H'_b$. (Note that V' may contain v_a and v_b, due to the cardinality constraint.) Since V' contains all vertices in $H'_a \cup H'_b$, we have $|H_a \setminus V'| \leq 1$ and $|H_b \setminus V'| \leq 1$ as claimed. □

We now prove the if direction of the lemma. It suffices to prove that there exist $V_a \in \mathcal{S}(H_a)$ and $V_b \in \mathcal{S}(H_b)$ such that $|V_a \setminus V_b| = |V_b \setminus V_a| \leq 1$; then two nodes H_a and H_b in A are joined by an edge in A. Let v'_a be the unique vertex in $H_a \setminus V'$ if $|H_a \setminus V'| = 1$, and otherwise v'_a is any vertex in $C(H_a) \setminus V'$; recall that $|H_a \cup C(H_a)| \geq$

$i + j$ and hence $C(H_a) \setminus V' \neq \emptyset$. We define $V_a = V' \cup \{v'_a\}$. Then, V_a contains all hubs in H_a. Furthermore, since $V_a \subseteq C[H_a]$, every vertex in $V_a \setminus H_a$ is adjacent to all hubs in H_a. Therefore, V_a contains an (i, j)-biclique with the hub set H_a, and hence we have $V_a \in \mathcal{S}(H_a)$. Similarly, let v'_b be the unique vertex in $H_b \setminus V'$ if $|H_b \setminus V'| = 1$, and otherwise v'_b is any vertex in $C(H_b) \setminus V'$. We define $V_b = V' \cup \{v_b\}$, and have $V_b \in \mathcal{S}(H_b)$. Then, $|V_a \setminus V_b| = |V_b \setminus V_a| \leq 1$ holds, as required. □

Finally, we give the following lemma, which completes the proof of Theorem 3.14.

Lemma 3.3 *The algorithm runs in $O(n^{2i+1})$ time.*

Proof. We first construct an auxiliary graph A. Let n be the number of vertices in G. The number of choices (subsets) of i vertices is $O(n^i)$. For each choice H, we can check in $O(i \cdot n)$ time whether $|C(H)| \geq j$ or not. Thus we can construct the vertex set of A in $O(n^{i+1})$ time. For any two nodes H_a and H_b in A, we can determine in $O(i \cdot n)$ time whether there is an edge between them using Lemma 3.2. In this way, we can construct the edge set of A in $O(n^{2i+1})$ time, since the size of the vertex set of A is in $O(n^i)$.

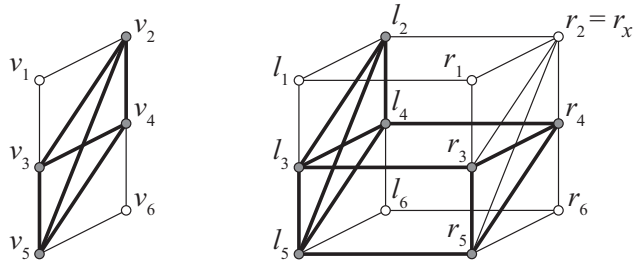
We finally check in $O(V(A) + E(A)) = O(n^{2i})$ time whether there is a path between H_s and H_t by breadth-first search on A. In this way, we can conclude that our algorithm runs in $O(n^{2i+1})$ time. □

3.4.4 Diameter-two graph

In this subsection, we consider the property "a graph has diameter at most two." Note that the induced and spanning versions are the same for this property.

Theorem 3.15 *Both induced and spanning versions under* TS *are PSPACE-complete for the property "a graph has diameter at most two."*

Proof. Since the induced version and the spanning version are the same for this property, it suffices to show PSPACE-hardness only for the induced version. We give



(a) G (b) G'

Figure 3.6: Reduction for the property "a graph has diameter at most two." The vertices of V_s in G and of V'_s in G' are depicted by gray vertices, where $r_x = r_2$.

a polynomial-time reduction from the CLIQUE RECONFIGURATION problem, which is the induced version (also the spanning version) of SUBGRAPH RECONFIGURATION for the property "a graph is a clique." This problem is known to be PSPACE-complete under both TJ and TS [19]; we give a reduction from the problem under TS.

Suppose that (G, V_s, V_t) is an instance of CLIQUE RECONFIGURATION under TS such that $|V_s| = |V_t| \geq 2$; otherwise it is a trivial instance. Then, we construct a corresponding instance (G', V'_s, V'_t) of the induced version under TS. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$, where $n = |V(G)|$. We form G' by making two copies of G and adding edges between corresponding vertices of the two graphs. (See Figure 3.6.) More formally, the vertex set $V(G')$ is defined as $V(G') = L \cup R$, where $L = \{l_i \mid v_i \in V(G)\}$ and $R = \{r_i \mid v_i \in V(G)\}$, and the edge set $E(G')$ is defined as $E(G') = E_l \cup E_r \cup E_c$, where $E_l = \{l_i l_j \mid v_i v_j \in E(G)\}$, $E_r = \{r_i r_j \mid v_i v_j \in E(G)\}$ and $E_c = \{l_i r_i \mid v_i \in V(G)\}$. For each $i \in \{1, 2, \ldots, n\}$, we call l_i and r_i *corresponding vertices*, and $l_i r_i \in E_c$ a *connecting edge*. For a vertex subset $V' \subseteq V(G')$, we say that a vertex $l_i \in V' \cap L$ (resp. $r_j \in V' \cap R$) is *exposed* in V' if the corresponding vertex $r_i \in R$ (resp. $l_j \in L$) does not belong to V'. We construct V'_s and V'_t so that each of them has exactly one exposed vertex, as follows. Let x and y be any indices such that $v_x \in V_s$ and $v_y \in V_t$, respectively. Then, we let $V'_s = \{l_i, r_i \mid v_i \in V_s\} \setminus \{r_x\}$ and $V'_t = \{l_i, r_i \mid v_i \in V_t\} \setminus \{r_y\}$. Note that l_x and l_y are the unique exposed vertices

in V'_s and V'_t, respectively. Since V_s and V_t form cliques in G, both $G'[V'_s]$ and $G'[V'_t]$ have diameter at most two. We have thus constructed our corresponding instance in polynomial time. The following claim completes the proof of the theorem.

Claim *(G, V_s, V_t) of* CLIQUE RECONFIGURATION *is a* yes*-instance if and only if the corresponding instance (G', V'_s, V'_t) is a* yes*-instance.*

We first prove the only-if direction of the claim. Suppose that there exists a reconfiguration sequence $\langle V_s = V_0, V_1, \ldots, V_\ell = V_t \rangle$ of cliques in G. Then, we show that V'_s is reconfigurable into V'_t by induction on ℓ. If $\ell = 0$ and hence $V_s = V_t$, then we can obtain V'_t from V'_s by exchanging r_y in V'_s with r_x (or $V'_s = V'_t$ already holds). We then consider the case where $\ell \geq 1$. Let V'_1 be the solution obtained from V_1 in the same way that we obtained V'_s from V_s, i.e., for any index z such that $v_z \in V_1$, let $V'_1 = \{l_i, r_i \mid v_i \in V_1\} \setminus \{r_z\}$. Then, by the induction hypothesis, we know that there exists a reconfiguration sequence between V'_1 and V'_t. Thus it suffices to show that we can reconfigure $V'_s = V'_0$ into V'_1. Let p and q be indices such that $\{v_p\} = V_0 \setminus V_1$ and $\{v_q\} = V_1 \setminus V_0$, respectively. Note that v_p and v_q are adjacent in G, because two cliques V_0 and V_1 appear consecutively in the reconfiguration sequence under TS. Then we consider two vertex subsets:

$$\begin{aligned} V'_a &= \{l_i, r_i \mid v_i \in V_0\} \setminus \{r_p\} = V'_0 \cup \{r_x\} \setminus \{r_p\}; \text{and} \\ V'_b &= \{l_i, r_i \mid v_i \in V_1\} \setminus \{r_q\} = V'_1 \cup \{r_z\} \setminus \{r_q\}. \end{aligned}$$

We observe that r_x and r_p are adjacent in G' if $r_x \neq r_p$, because both v_x and v_p are contained in the clique V_0. Thus, we can reconfigure V'_0 into V'_a (or already $V'_0 = V'_a$ if $r_x = r_p$). Similarly, we observe that r_z and r_q are adjacent in G', and we can reconfigure V'_1 into V'_b (or already $V'_1 = V'_b$ if $r_z = r_q$); since reconfiguration is

reversible, we can also reconfigure V'_b into V'_1. Furthermore, it holds that

$$\begin{aligned} V'_b &= \{l_i, r_i \mid v_i \in V_1\} \setminus \{r_q\} \\ &= \{l_i, r_i \mid v_i \in V_0\} \cup \{l_q, r_q\} \setminus \{l_p, r_p, r_q\} \\ &= V'_a \cup \{l_q\} \setminus \{l_p\}. \end{aligned}$$

Since v_p and v_q are adjacent in G, l_p and l_q are adjacent in G'. Thus, we can reconfigure V'_a into V'_b. In this way, we obtain from V'_0, V'_a, V'_b, V'_1 (by removing redundant ones if needed) the reconfiguration sequence from V'_0 and V'_1.

We now prove the if direction of the claim. Suppose that there exists a reconfiguration sequence $\mathcal{V}' = \langle V'_s = V'_0, V'_1, \ldots, V'_\ell = V'_t \rangle$ of solutions (vertex subsets) whose induced subgraphs are of diameter at most two. We show that any solution V'_i in $\mathcal{V}$ satisfies the following two conditions: (a) V'_i contains exactly one exposed vertex; and (b) either $V'_i \cap L$ or $V'_i \cap R$ forms a clique of size $|V_s| = |V_t|$ in G'. To see this, it suffices to show that if V'_{i-1} satisfies both Conditions (a) and (b), then V'_i also satisfies them; recall that $V'_s = V'_0$ initially satisfies the two conditions. Consider the case where V'_{i-1} has the unique exposed vertex in the side L, say $l_p \in V'_{i-1}$, and hence $r_p \notin V'_{i-1}$; the other case is symmetric. Because $G'[V'_i]$ must have diameter at most two, we know that V'_i is obtained from V'_{i-1} by one of the following three moves: (1) a token on a vertex $r \in V'_{i-1} \cap R$ is moved to r_p; (2) the token on l_p is moved to its corresponding vertex r_p; or (3) the token on l_p is moved to a vertex in $L \setminus V'_{i-1}$ which is adjacent to all vertices $V'_{i-1} \cap L$. Notice that any other move makes the resulting graph have diameter more than two. For each of the three moves, we observe that the resulting subgraph V'_i satisfies Conditions (a) and (b). Then, by Condition (b), each $V'_i \in \mathcal{V}'$ induces a clique of size $|V_s| = |V_t|$ in either L or R, and we can obtain a desired sequence of cliques between V_s and V_t. □

Chapter 4 Reconfiguration of Steiner trees

In this chapter, we study the complexity of reconfiguration problems of Steiner trees under several adjacency relations. In Section 4.1, we give formal definitions for the reconfiguration problems of Steiner trees which we deal with in this chapter. In Section 4.2, we introduce another reconfiguration problem with the concept "Steiner sets;" we call the problem the *auxiliary problem*. This problem play important roles in this chapter. Specifically, the auxiliary problem can be reduced from/to the reconfiguration problem of Steiner trees under several adjacency relations without changing an input graph. We thus study in Section 4.2 the complexity of the auxiliary problem. In Section 4.3 to 4.6, we study the reconfiguration problem under each adjacency relations.

4.1 Definition of problems and preliminaries

We formally define the reconfiguration problems of Steiner trees. Recall that for an unweighted graph G and a vertex subset $S \subseteq V(G)$, called a *terminal set*, a *Steiner tree* of G for S is a subtree of G which contains all vertices in S. A Steiner tree of G for S is *minimum* if it has the minimum number of edges among all Steiner trees of G for S. Note that minimum Steiner trees can be seen as a generalization of shortest paths, because any shortest path in G between two vertices s and t forms a minimum Steiner tree of G for $S = \{s, t\}$. We use the terms *node* for Steiner trees and *vertex* for input graphs.

Let G be an input graph and S be an input terminal set. Then, feasible solu-

tions are defined as all Steiner trees of G for S, and we consider the following four adjacency relations:

- **Vertex exchange** (VE, for short): We say that two Steiner trees T and T' of G for S are *adjacent under* VE if there exist two vertices $v \in V(T)$ and $v' \in V(T')$ such that;
  - $V(T) \setminus \{v\} = V(T') \setminus \{v'\}$.
- **Local vertex exchange** (LVE, for short): We say that two Steiner trees T and T' of G for S are *adjacent under* LVE if there exist two vertices $v \in V(T)$ and $v' \in V(T')$ such that;
  - $T[V(T) \setminus \{v\}] = T'[V(T') \setminus \{v'\}]$.
- **Local vertex exchange without changing neighbors** (LVE-N for short): We say that two Steiner trees T and T' of G for S are *adjacent under* LVE-N if there exist two vertices $v \in V(T)$ and $v' \in V(T')$ such that;
  - $T[V(T) \setminus \{v\}] = T'[V(T') \setminus \{v'\}]$; and
  - $N_T(v) = N_{T'}(v')$.
- **Edge exchange** (EE, for short): We say that two Steiner trees T and T' of G for S are *adjacent under* EE if there exist two edges $e \in E(T)$ and $e' \in E(T')$ such that;
  - $E(T) \setminus \{e\} = E(T') \setminus \{e'\}$.

Consider each adjacency relation $\mathcal{R}$ in VE, LVE, LVE-N and EE and two Steiner trees T and T'. Then we write $T \overset{\mathcal{R}}{\leftrightarrow} T'$ if they are adjacent in the solution space under $\mathcal{R}$, and $T \overset{\mathcal{R}}{\leftrightsquigarrow} T'$ if there exists a reconfiguration sequence in the solution space under $\mathcal{R}$. $T \overset{\text{LVE-N}}{\leftrightsquigarrow} T'$.

Then, we deal with the reachability variant of reconfiguration problems whose solution spaces are defined above; we use the terms Steiner tree reconfiguration for the reconfiguration problem with the solution space, and denote by Reach-STR the reachability variant of Steiner tree reconfiguration. More

specifically, for an adjacency relation $\mathcal{R} \in \{\mathsf{VE}, \mathsf{LVE}, \mathsf{LVE\text{-}N}, \mathsf{EE}\}$, we consider the following problem: We are given a graph G, a terminal set $S \subseteq V(G)$, and two Steiner trees T_0 and T_r, then asked to determine whether $T_0 \overset{\mathcal{R}}{\leftrightsquigarrow} T_r$ or not.

We denote by a 4-tuple (G, S, T_0, T_r) an instance of the problems. In this chapter, we assume without loss of generality that $|V(T_0)| = |V(T_r)|$ (and hence $|E(T_0)| = |E(T_r)|$) holds; otherwise it is clearly a no-instance.

4.2 Auxiliary problem: reconfiguration of Steiner sets

In this section,we first introduce the concept of "Steiner sets" and their reconfiguration. Then, we study the computational complexity of the problem that determine the existence of the reconfiguration sequence of Steiner sets. We again note that Reach-STR under several adjacency relations can be reduced from/to this problem with Steiner sets without changing an input graph.

4.2.1 Steiner sets and their reachability problem

For a graph G and a terminal set S, a *Steiner set* of G for S is a vertex subset $F \subseteq V(G)$ such that $S \subseteq F$ and $G[F]$ is connected. A Steiner set F of G for S is *minimum* if the cardinality of F is minimum among all Steiner sets of G for S. On the other hand, a Steiner set F of G for S is *minimal* if F contains no Steiner set other than F itself. Notice that if a subtree T of G is a Steiner tree for S, then $V(T)$ is a Steiner set of G for S. Conversely, if F is a Steiner set of G for S, then any spanning tree of $G[F]$ is a Steiner tree for S. In particular, a Steiner tree T is minimum if and only if a Steiner set $V(T)$ is minimum.

For two Steiner sets F and F' of G for S, a sequence $\langle F = F_0, F_1, \ldots, F_\ell = F' \rangle$ of Steiner sets of G for S is called a *Steiner set sequence* between F and F' if $|F_i \setminus F_{i+1}| = |F_{i+1} \setminus F_i| = 1$ holds for each $i \in \{0, 1, \ldots, \ell - 1\}$. We write $F \leftrightarrow F'$ if

$|F \setminus F'| = |F' \setminus F| = 1$, and $F \leftrightsquigarrow F'$ if there exists a Steiner set sequence between F and F'. We say that two Steiner sets F and F' are *reachable* if $F \leftrightsquigarrow F'$. Note that all Steiner sets in the sequence have the same cardinality.

Then we study in this section the following problem; we call the problem the *auxiliary problem*: We are given a graph G, a terminal set $S \subseteq V(G)$, and two Steiner sets F_0 and F_r, then asked to determine whether there exists a Steiner set sequence between them. We denote by a triple (G, S, F_0, F_r) an instance of the auxiliary problem.

4.2.2 PSPACE-hardness for planar graphs.

In this subsection, we consider planar graphs, and give the following theorem.

Theorem 4.1 *The auxiliary problem is PSPACE-hard for planar graphs even if two given Steiner sets are minimum.*

To prove the theorem, we construct a polynomial-time reduction from the MINIMUM VERTEX COVER RECONFIGURATION (MVCR, for short) problem.

Recall that a *vertex cover* C of a graph G is a vertex subset of G which contains at least one of the two endpoints of every edge in G. A vertex cover C of G is *minimum* if the cardinality of C is minimum among all vertex covers of G. Then a solution space of MVCR is defined as follows: Feasible solutions are defined as all minimum vertex covers of an input graph, and there is an edge between two minimum vertex covers C and C' if they are adjacent under TJ (i.e. $|C \setminus C'| = |C' \setminus C| = 1$ holds). We here consider the reachability variant of MVCR, that is, we are given a graph G and two minimum vertex covers C_0 and C_r of G, then asked to determine whether there exists a reconfiguration sequence between them in the solution space. We denote by a triple (G, C_0, C_r) an instance of MVCR. This problem is known to be PSPACE-complete for planar graphs [16].[1]

[1]Precisely, Hearn and Demaine [16] showed the PSPACE-completeness for the recon-

Reduction. Let (G', C'_0, C'_r) be a given instance of MVCR such that G' is a planar graph. We fix a planar embedding of G' arbitrarily, and denote by $\mathsf{face}(G')$ the set of all faces (including the outer face) of G'. We construct the corresponding instance (G, S, F_0, F_r) of the auxiliary problem, as follows. (See Figure 4.1 as an example.)

We first construct the corresponding graph G from G'. For each face $f \in \mathsf{face}(G')$, we add a new vertex w_f, and join w_f and all vertices on the boundary of f by adding new edges. Then, we subdivide each original edge $e = uv \in E(G')$ by adding a new vertex w_e. Let G be the resulting graph. Then, G is a planar graph.

We then define the corresponding terminal set S as the set of all newly added vertices, that is, $S = \{w_f \mid f \in \mathsf{face}(G')\} \cup \{w_e \mid e \in E(G')\}$; each w_f is called a *face-terminal*, while each w_e is called an *edge-terminal*.

We finally define the corresponding minimum Steiner sets F_0 and F_r as the set $C'_0 \cup S$ and $C'_r \cup S$, respectively; we will prove later in Lemma 4.1 that both F' and F' form minimum Steiner sets of G for S.

This completes the construction of (G, S, F_0, F_r). The construction can be done in polynomial time.

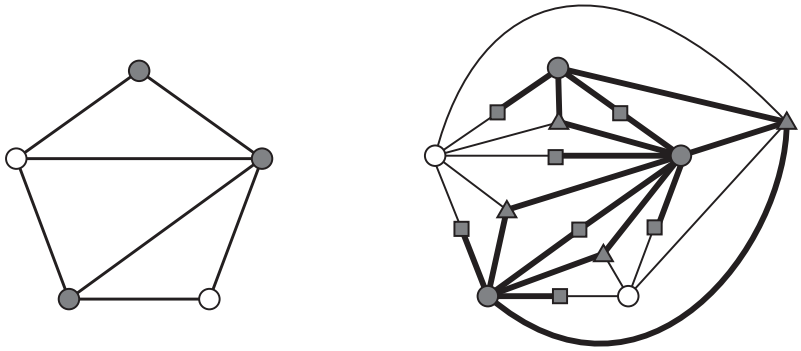
Correctness. We start with showing that both F and F' form minimum Steiner sets of G for S.

Lemma 4.1 *Let C' be a vertex subset of $V(G')$. Then, C' is a minimum vertex cover of G' if and only if $C' \cup S$ is a minimum Steiner set of G for S.*

Proof. It suffices to show that C' is a (not necessary minimum) vertex cover of G' if and only if $C' \cup S$ is a (not necessary minimum) Steiner set of G for S.

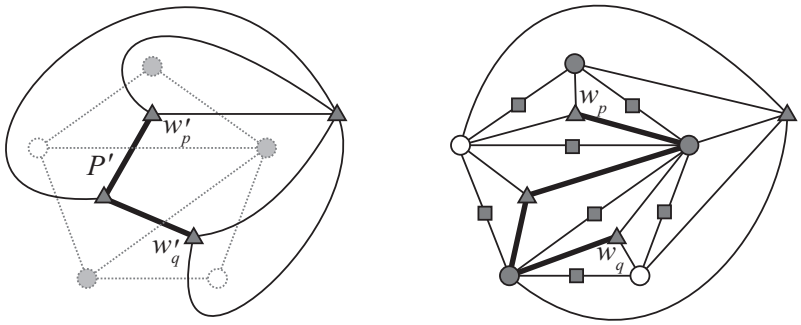
We first prove the if direction. Suppose that $F = C' \cup S$ is a Steiner set of G for S. For each edge $e \in E(G')$, the corresponding edge-terminal $w_e \in S$ is contained in F. Then, we know that at least one of the two adjacent vertices of w_e is also contained

figuration of maximum independent sets. However, it immediately yields the PSPACE-completeness of MVCR.



(a) G' (b) G

Figure 4.1: (a) An input graph G' of MVCR with a minimum vertex cover C', where the vertices in C' are depicted by gray vertices, and (b) its corresponding planar graph G of the auxiliary problem together with the minimum Steiner set to C', where face-terminals are depicted by triangles, edge-terminals by squares, and connected subgraph corresponding to the minimum Steiner set by thick lines.



(a) H' (b) G

Figure 4.2: (a) The dual graph of the graph G in Figure 4.1(a) with a path P' between w'_p and w'_q depicted by thick lines, and (b) a walk between w_p and w_q corresponding to P' depicted by thick lines.

in F, since $|F \setminus \{w_e\}| > 0$ and $G[F]$ is connected. Furthermore, it is observed that such a vertex is in C'. Therefore, we can conclude that for each edge $e \in E(G')$, C' contains at least one of the two endpoints of e, and hence C' is a vertex cover of G'.

We thus prove the only-if direction. Suppose that C' is a vertex cover of G'. Then, it suffices to show that $G[C' \cup S]$ is connected, since S is clearly contained in the vertex subset $C' \cup S$. To this end, we show the following three claims:

(A) each edge-terminal is adjacent to at least one vertex in C' on G;

(B) each vertex in C' is adjacent to at least one face-terminal on G; and

(C) for any pair of two face-terminals, there exists a walk between them in $G[C' \cup S]$.

We first consider the claim (A). Since C' is a vertex cover of G', C' contains at least one of the two endpoints of every edge in G'. We thus know that C' contains at least one of the two adjacent vertices of each edge-terminal in G, and hence the claim (A) follows. The claim (B) can be easily obtained, since any vertex in C' (more specifically, $V(G')$) belong to at least one face of G'. We finally consider the claim (C). Let w_p, w_q be any face-terminals in G. Then we show that there exists a walk between w_p and w_q on G. (See Figure 4.2 as an example.) Suppose that H' be the dual graph of G'. We notice that each face-terminal in G corresponds to a vertex in H'. Let w'_p and w'_q be the vertices in H' which correspond to w_p and w_q, respectively. Since the dual graph H' is connected, there is a path between w'_p and w'_q on H'; we denote such a path by the sequence $P' = \langle w'_p = w'_0, w'_1, \ldots, w'_{\ell'} = w'_q \rangle$ of vertices in H'. Then we show in the following that for any consecutive vertices w'_i and w'_{i+1} in P', there exists a path on G between the face-terminals w_i and w_{i+1} corresponding to w'_i and w'_{i+1}. Since H' is the dual graph of G', there exists an one-to-one correspondence between $E(H')$ and $E(G')$. Therefore, there exists an edge e' in G' which corresponds to the edge $w'_i w'_{i+1}$ in H'. Then at least one of the two endpoints of e' is contained in C'; let $u' \in C'$ be such a endpoint of e'. We then know from the construction of G that there exist two edges $w_i u'$ and $u' w_{i+1}$ in G, and hence there exists a path between w_i and w_{i+1} on G. In this way, we can conclude that there exists a walk between w_p and w_q on G. Therefore, the claim (C) follows. □ Lemma 4.1 ensures the existence of two minimum Steiner trees T_0 and T_r in our reduction.

The following lemma completes the proof of Theorem 4.1.

Lemma 4.2 *(G', C'_0, C'_r) is a* yes*-instance of MVCR if and only if (G, S, F_0, F_r) is a* yes*-instance of the auxiliary problem.*

Proof. First, suppose that there exists a Steiner set sequence $\langle F_0 = F_0, F_1, \ldots, F_\ell = F_r$ between F_0 and F_r. Then, Lemma 4.1 implies that the sequence $\langle C'_0 = F_0 \setminus S, F_1 \setminus S, \ldots, F_\ell \setminus S = C'_r\rangle$ is a vertex cover sequence on G' between C'_0 and C'_r.

Second, suppose that there exists a vertex cover sequence $\langle C'_0, C'_1, \ldots, C'_{\ell'} = C'_r\rangle$ between C'_0 and C'_r. Then, Lemma 4.1 implies that the sequence $\langle F_0 = C'_0 \cup S, C'_1 \cup S, \ldots, C'_{\ell'} \cup S = F_r\rangle$ is a Steiner set sequence on G between F_0 and F_r. □

4.2.3 PSPACE-hardness for split graphs.

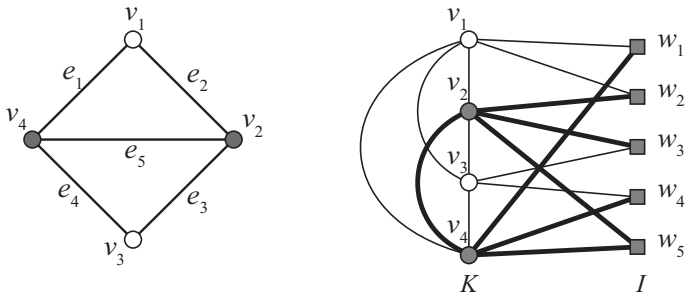
In this subsection, we consider split graphs, and give the following theorem.

Theorem 4.2 *The auxiliary problem is PSPACE-hard for split graphs even if two given Steiner sets are minimum.*

To prove the lemma, we again construct a polynomial-time reduction from MVCR.

Reduction. Let (G', C'_0, C'_r) be an instance of MVCR. We assume without loss of generality that $|E(G)| \geq 2$; otherwise, the instance is a trivial instance. Then we construct the corresponding instance (G, S, F_0, F_r) of the auxiliary problem, as follows. (See Figure 4.3 as an example.) We first construct a graph G. Let K be a complete graph with $V(K) = V(G')$. For the edge set $E(G') = \{e_1, e_2, \ldots, e_{|E(G')|}\}$, we consider an edge less graph I with $V(I) = \{w_1, w_2, \ldots, w_{|E(G')|}\}$; each vertex w_i in $V(I)$ corresponds to an edge e_i in $E(G')$. For each edge $e_i = v_p v_q \in E(G')$, we add two edges joining $w_i \in V(I)$ and $v_p, v_q \in V(K)$. This completes the construction of G; notice that G is a split graph. We then define S as $V(I)$. We finally define F_0 as $C'_0 \cup S$, and F_r as $C'_r \cup S$; we will prove later in Lemma 4.3 that both F' and F' form minimum Steiner sets of G for S. This completes the construction of (G, S, T_0, T_r).

Correctness. The proof of the correctness is almost same as that for planar graphs in Section 4.2.2. We start with showing the following lemma, which corresponds to



(a) G' (b) G

Figure 4.3: (a) An example of an input graph G' of MVCR with a vertex cover C' represented by colored vertices, and (b) its corresponding split graph G of the auxiliary problem with the Steiner set corresponding to C', where terminals are depicted by squares, and the connected subgraph corresponding to the Steiner set by thick lines.

Lemma 4.1.

Lemma 4.3 *Let G' be a graph with at least two edges and C' be any vertex subset of $V(G')$. Then C' is a minimum vertex cover of G' if and only if $C' \cup S = C' \cup V(I)$ is a minimum Steiner set of G for S.*

Proof. It suffices to show that C' is a (not necessary minimum) vertex cover of G' if and only if $C' \cup S$ is a (not necessary minimum) Steiner set of G for S.

We first prove the only-if direction. Suppose that $C' \subseteq V(G')$ is a vertex cover of G'. To prove $C \cup V(I)$ is a Steiner set for $S = V(I)$, it suffices to show that $G[C \cup V(I)]$ is connected. For each edge $e_i = v_j v_k \in E(G')$, we have $v_j \in C$ or $v_k \in C$. Since $G[C]$ forms a clique and G has two edges $w_i v_j$ and $w_i v_k$ for the vertex $w_i \in V(G)$ corresponding to $e_i \in E(G')$, we thus obtain that $G[C \cup V(I)]$ is connected.

We then prove the if direction. For a vertex subset $C \subseteq V(G')$, suppose that $C \cup V(I)$ is a Steiner set for $S = V(I)$. Since $V(I)$ forms an independent set of G, every vertex $w_i \in V(I)$ must be adjacent to at least one vertex $v_j \in C$ in F. By the construction of the graph G, the edge e_i in G' (corresponding to $w_i \in V(G)$) is incident to $v_j \in V(G')$. Thus, C forms a vertex cover of G'. □

Lemma 4.3 ensures the existence of two minimum Steiner trees T_0 and T_r in our reduction. The following lemma completes the proof of Theorem 4.2; the proof is omitted, since it is almost same as that for Lemma 4.2.

Lemma 4.4 *(G', C'_0, C'_r) is a* yes*-instance of MVCR if and only if (G, S, F_0, F_r) is a* yes*-instance of the auxiliary problem.*

4.2.4 Linear-time algorithm for cographs

In this subsection, we consider cographs, and give the following theorem.

Theorem 4.3 *Suppose that G is a cograph and $S \subseteq V(G)$ is a terminal set of G. Then, for any two Steiner sets of G for S with the same size, there is a Steiner set sequence between them on G. Thus, the auxiliary problem is linear-time solvable for cographs.*

To prove the theorem, we use the following proposition that holds for any cograph.

Proposition 1 *Any connected cograph G has at most one cut-vertex.*

Proof. We first claim that any cut-vertex v_c in G is adjacent to all vertices in $V(G) \setminus \{v_c\}$. Suppose for a contradiction that there exists a vertex $w \in V(G) \setminus \{v_c\}$ which is not adjacent to v_c. We choose such a non-adjacent vertex w which is closest to v_c. (See Figure 4.4.) Then, G has a path $ww'v_c$, where w' is adjacent to both w and v_c. Since v_c is a cut-vertex of G, the induced subgraph $G \setminus \{v_c\}$ consists of at least two connected components; we denote by G_1 the connected component containing both w and w', and by G_2 an arbitrary connected component in $G \setminus \{v_c\}$ other than G_1. Let u be a vertex in $V(G_2)$ which is adjacent to v_c. Then, u is adjacent to neither w nor w', and hence $G[\{w, w', v_c, u\}]$ is a path of four vertices. This contradicts the assumption that G is a cograph.

We now prove the proposition. Suppose for a contradiction that a connected cograph G has more than one cut-vertices, say $u_c, v_c \in V(G)$. However, $G \setminus \{u_c\}$

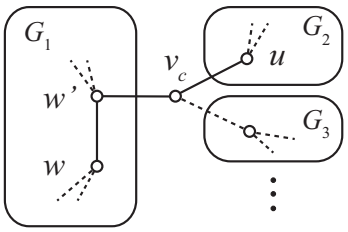


Figure 4.4: Illustration for the proof of Proposition 1.

is connected, because v_c is adjacent to all vertices in $V(G) \setminus \{v_c\}$ as we mentioned above; a contradiction. □

We then prove the theorem 4.3.

Proof of Theorem 4.3. Let F_p and F_q be any two Steiner sets for S such that $|F_p| = |F_q|$ and $k = |F_p \setminus F_q| = |F_q \setminus F_p|$. We prove the theorem by induction on k.

If $k \leq 1$, then we have either $F_p = F_q$ or $F_p \leftrightarrow F_q$. Therefore, $F_p \leftrightsquigarrow F_q$ holds for $k \leq 1$.

Consider the case where $k \geq 2$. Since $G[F_q]$ is connected, there exists at least one vertex u in $F_q \setminus F_p$ which is adjacent to a vertex in $F_p \cap F_q$. Notice that $F_p \cap F_q \neq \emptyset$ holds, because $S \subseteq F_p \cap F_q$ and $S \neq \emptyset$. Since $|F_p \setminus F_q| = k \geq 2$, Proposition 1 implies that $F_p \setminus F_q$ contains at least one vertex w which is not a cut-vertex of $G[F_p]$. Note that $w \notin S$ holds, because $S \subseteq F_p \cap F_q$. Let $F'_p = (F_p \setminus \{w\}) \cup \{u\}$, then F'_p is a Steiner set for S. Therefore, we have $F_p \leftrightarrow F'_p$. Since $|F'_p \setminus F_q| = |F_q \setminus F'_p| = k - 1$, by the induction hypothesis $F'_p \leftrightsquigarrow F_q$ holds. Therefore, we can conclude that $F_p \leftrightarrow F'_p \leftrightsquigarrow F_q$. □

4.2.5 Linear-time algorithm for interval graphs

In this subsection, we consider interval graphs, and give the following theorem.

Theorem 4.4 *Suppose that G is an interval graph and $S \subseteq V(G)$ is a terminal set of G. Then, for any two Steiner sets of G for S with the same size, there is a Steiner set sequence between them on G. Thus, the auxiliary problem is linear-time*

solvable for interval graphs.

We prove the theorem by giving two lemmas. We first give the following lemma, which holds for not only interval graphs but also any graph.

Lemma 4.5 *For a graph G and a terminal set $S \subseteq V(G)$, let F_p and F_q be any two Steiner sets for S such that $|F_p| = |F_q|$ and $F_p \cap F_q \neq \emptyset$. Then, $F_p \leftrightsquigarrow F_q$ if there exists a Steiner set F such that $F \subseteq F_p \cap F_q$.*

Proof. Suppose that there exists a Steiner set F such that $F \subseteq F_p \cap F_q$; we take the maximal one, that is, $F \cup \{w\}$ is not a Steiner set for any vertex $w \in (F_p \cap F_q) \setminus F$. Note that, since $F \subseteq F_p \cap F_q$, we have $|F_p \setminus F| = |F_q \setminus F|$. We prove the lemma by induction on $k = |F_p \setminus F| = |F_q \setminus F|$. If $k = 0$, then we have $F_p = F = F_q$ and hence $F_p \leftrightsquigarrow F_q$.

Consider the case where $k \geq 1$. (See Figure 4.5(a).) We consider any vertex ordering $u_1, u_2, \ldots, u_k$ of $F_p \setminus F$ such that $G[F \cup \{u_1, u_2, \ldots, u_i\}]$ is connected for each $i \in \{1, 2, \ldots, k\}$; such a vertex ordering always exists because $G[F_p] = G[F \cup \{u_1, u_2, \ldots, u_k\}]$ is connected. Then, since F is maximal, we know that $u_1 \in F_p \setminus (F_p \cap F_q) = F_p \setminus F_q$. Similarly, consider any vertex ordering $w_1, w_2, \ldots, w_k$ of $F_q \setminus F$ such that $G[F \cup \{w_1, w_2, \ldots, w_i\}]$ is connected for each $i \in \{1, 2, \ldots, k\}$; we know that $w_1 \in F_q \setminus F_p$. Then, let $F'_p = (F_p \cup \{w_1\}) \setminus \{u_k\}$, and let $F' = F \cup \{w_1\}$, as illustrated in Figure 4.5(b). To apply the induction hypothesis, we now claim that

(a) F'_p is a Steiner set for S such that $|F'_p| = |F_q|$; and

(b) F' is a Steiner set for S such that $F' \subseteq F'_p \cap F_q$ and $|F'_p \setminus F'| = |F_q \setminus F'| = k-1$.

Then, by applying the induction hypothesis, we have $F'_p \leftrightsquigarrow F_q$ and hence $F_p \leftrightarrow F'_p \leftrightsquigarrow F_q$ holds.

We first prove Claim (a). Since F is a Steiner set for S and $u_k \in F_p \setminus F$, we know that $u_k \notin S$. Since $S \subseteq F_p$, we thus have $S \subseteq (F_p \cup \{w_1\}) \setminus \{u_k\} = F'_p$. In addition, since $G[F_p]$ is connected and $F \subseteq F_p$, the choice of u_k and w_1 implies that $G[F'_p]$ is connected. Furthermore, $|F'_p| = |F_p| = |F_q|$. We thus verified that Claim (a) holds.

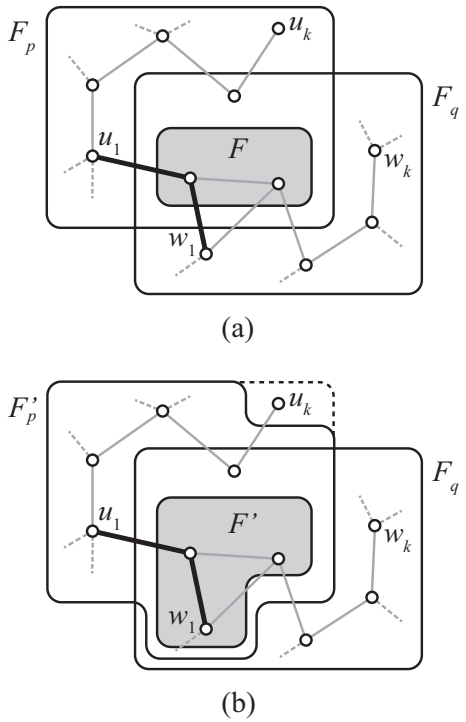


Figure 4.5: Illustration for the proof of Lemma 4.5.

We then prove Claim (b). Since F is a Steiner set for S, by the choice of w_1 we obtain that $F' = F \cup \{w_1\}$ ($\supseteq S$) is a Steiner set for S, too. Since $u_k \notin F$ and $F \subseteq F_p \cap F_q$, we have $F \subseteq (F_p \setminus \{u_k\}) \cap F_q \subset F'_p \cap F_q$. In addition, since w_1 is contained in both F'_p and F_q, we can conclude that $F' = F \cup \{w_1\} \subseteq F'_p \cap F_q$. Then, since $w_1 \in F_q \setminus F_p$, we have $|F'_p \setminus F'| = |F_q \setminus F'| = k - 1$. □

The following lemma completes Theorem 4.4 by combining it with Lemmas 4.5.

Lemma 4.6 *For an interval graph G and a terminal set $S \subseteq V(G)$, let F_p and F_q be any two Steiner sets for S such that $|F_p| = |F_q|$. Then, there always exist two Steiner sets F'_p and F'_q such that $F_p \leftrightsquigarrow F'_p$, $F_q \leftrightsquigarrow F'_q$, and there exists a Steiner set F' for S such that $F' \subseteq F'_p \cap F'_q$.*

Proof. Let G be an interval graph with $V(G) = \{v_1, v_2, \ldots, v_n\}$, and let $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ be an interval representation of G such that I_i corresponds to $v_i \in V(G)$ for each $i \in \{1, 2, \ldots, n\}$. For an interval $I_i \in \mathcal{I}$, we denote by $l(I_i)$ and $r(I_i)$ the left and right coordinates of I_i, respectively; we sometimes call the values $l(I_i)$

and $r(I_i)$ the *l-value* and *r-value* of I_i, respectively. We may assume without loss of generality that all l-values and r-values are distinct. For notational convenience, we sometimes identify a vertex $v_i \in V$ with its corresponding interval $I_i \in \mathcal{I}$, and simply write $l(v_i) = l(I_i)$ and $r(v_i) = r(I_i)$. Let s_{left} be the terminal in S which has the minimum l-value, that is, $l(s_{\text{left}}) = \min\{l(s) \mid s \in S\}$, while let s_{right} be the terminal in S which has the maximum r-value, that is, $r(s_{\text{right}}) = \max\{r(s) \mid s \in S\}$. Note that $s_{\text{left}} = s_{\text{right}}$ may hold.

We prove the lemma by constructing two Steiner sets F'_p and F'_q such that $F_p \leftrightsquigarrow F'_p$, $F_q \leftrightsquigarrow F'_q$, and $[l(s_{\text{left}}), r(s_{\text{right}})]$ is covered by the intervals in $F'_p \cap F'_q$. Then, we can obtain a Steiner set F' from $F'_p \cap F'_q$, because all intervals in S are contained in $[l(s_{\text{left}}), r(s_{\text{right}})]$.

We sweep the interval representation $\mathcal{I}$ from left to right, starting from $l(s_{\text{left}})$ and ending at $r(s_{\text{right}})$. Let x be the first coordinate such that $F_p \cap F_q$ contains no interval (vertex) covering x. If such an x does not exist in $[l(s_{\text{left}}), r(s_{\text{right}})]$, then F_p and F_q are desired Steiner sets already.

We thus consider the case where such an x exists in $[l(s_{\text{left}}), r(s_{\text{right}})]$. (See Figure 4.6.) For notational convenience, for a vertex subset V' and a coordinate x, we denote by $(V')_x$ the set of all vertices in V' that cover x, that is, $(V')_x = \{v \in V' \mid l(v) \leq x \leq r(v)\}$. Then, $(F_p \cap F_q)_x = \emptyset$. Since $G[F_p]$ and $G[F_q]$ are connected, we have $(F_p \setminus F_q)_x \neq \emptyset$ and $(F_q \setminus F_p)_x \neq \emptyset$. Let u be the vertex in $(\triangle F_p F_q)_x$ whose r-value is maximum. Note that $u \notin S$, because $S \subseteq F_p \cap F_q$ holds. We may assume that $u \in F_q \setminus F_p$; the other case is symmetric. Then, we will prove that there is a vertex w in $F_p \setminus F_q$ such that $(F_p \cup \{u\}) \setminus \{w\}$ remains a Steiner set for S. Notice that, since $w \in F_p \setminus F_q$ and hence $w \notin S$, it suffices to show that $G[(F_p \cup \{u\}) \setminus \{w\}]$ is connected. Therefore, we are done if $(F_p \setminus F_q)_x$ contains a vertex w such that $G[(F_p \cup \{u\}) \setminus \{w\}]$ is connected. (See Figure 4.6(a).)

We thus consider the remaining case: $G[(F_p \cup \{u\}) \setminus \{w'\}]$ is not connected for

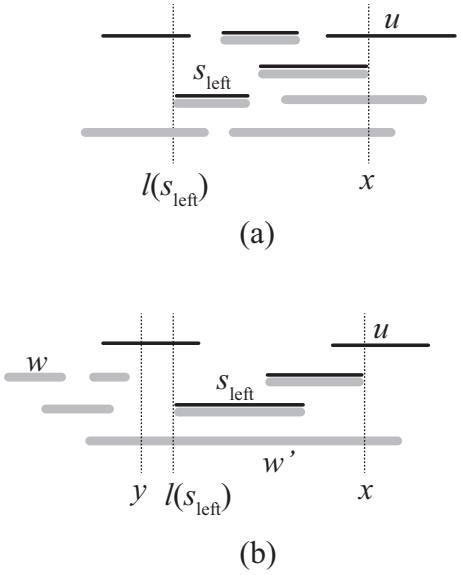


Figure 4.6: Illustration for the proof of Lemma 4.6, where the intervals in F_p are depicted by thick gray lines and those in F_q by thin black lines.

any vertex w' in $(F_p \setminus F_q)_x$. (See Figure 4.6(b).) Then, there is a coordinate y such that $(F_p)_y = w'$. Since $w' \notin F_q$, by the definition of x we know that $y < l(s_{\text{left}})$. Therefore, F_p has a vertex w'' such that $r(w'') < y < l(s_{\text{left}})$. Among such vertices in F_p, we choose the one w having the minimum r-value. Then, w is not a cut-vertex of $G[F_p]$, and hence $G[(F_p \cup \{u\}) \setminus \{w\}]$ is connected.

In this way, we can continue sweeping the interval representation $\mathcal{I}$ until we reach the coordinate $r(s_{\text{right}})$. Then, the resulting Steiner sets F'_p and F'_q are desired ones, and hence the lemma follows. □

4.2.6 Polynomial-time algorithm for cactus graphs

In this subsection, we consider cactus graphs, and give the following theorem.

Theorem 4.5 *The auxiliary problem is polynomial-time solvable for cactus graphs.*

In our algorithm, we greedily reconfigure given two Steiner sets into other Steiner sets, called "canonical" canonical Steiner sets. Then we show that the instance is yes-instance if and only if the obtained canonical Steiner sets have the same

"characteristic."

Preliminaries for algorithm.

We first introduce some definitions and concepts used in our algorithm. Let (G, S, F_0, F_r) be an instance of the auxiliary problem, where G is a cactus graph. We denote by n and n_{s} the number of vertices in G and F_0 (the same with F_r), respectively. We say that $v \in V(G)$ is *critical* if $G[V(G) \setminus \{v\}]$ has no connected component G' containing all terminals, that is, $S \subseteq G'$; notice that every terminals are critical. We observe that any Steiner set must contain all critical vertices. Let $K \subseteq V(G)$ be a set of all critical vertices of G.

Consider a set of cycles $\mathcal{C} = \{C_1, C_2, \ldots, C_{|\mathcal{C}|}\}$ where $\mathcal{C}$ has all cycles of G which contains at least two and at most $|C_i| - 1$ critical vertices. We consider each C_i in $\mathcal{C}$ as a vertex subset. Then, $2 \leq |C_i \cap K| \leq |C_i| - 1$ for each $C_i \in \mathcal{C}$. We subdivide each cycle C_i in $\mathcal{C}$ into subpaths $P_{i,1}, P_{i,2}, \ldots$ by removing all critical vertices from C_i, that is, each $P_{i,j}$ is a connected component of $G[C_i \setminus K]$. We again consider each subpath $P_{i,j}$ as a vertex subset. Let $\mathcal{P}_i = \{P_{i,1}, P_{i,2}, \ldots, P_{i,|\mathcal{P}_i|}\}$ be a set of all subpaths in C_i constructed by the above operation. In the set of subpaths, we assume without loss of generality that $|P_{i,j}| \leq |P_{i,k}|$ for $j < k$. Note that we have $|\mathcal{P}_i| \geq 1$ since any cycle in $\mathcal{C}$ contains at least one non-critical vertex.

Suppose that F is any Steiner set of G for S which has the cardinality n_{s}. For each cycle C_i in $\mathcal{C}$, $F \cap C_i$ induces a path or a cycle (specifically, a connected subgraph of G), since G is a cactus graph and F induces a connected subgraph of G. We know that if $F \cap C_i$ induces a cycle, then all subpaths in $\mathcal{P}_i$ are contained in the induced cycle. On the other hand, if $F \cap C_i$ induces a path, then exactly $|\mathcal{P}_i| - 1$ subpaths in $\mathcal{P}_i$ are contained in the induced path; in other words, there is the unique path $P_{i,j}$ in $\mathcal{P}_i$ which satisfies $P_{i,j} \nsubseteq F \cap C_i$ (actually, $P_{i,j} \nsubseteq F$). Then, we define the *lack path* of F for C_i, as follows. If $F \cap C_i$ induces a path, the lack path of F for C_i is the unique

subpath in $\mathcal{P}_i$ which is not contained in $F \cap C_i$, while if $F \cap C_i$ induces a cycle, the lack path of F for C_i is the subpath $P_{i,|\mathcal{P}_i|}$ in $\mathcal{P}_i$ (i.e. the subpath whose second index is maximized in $\mathcal{P}_i$). Let $L(F) \in \{1, \ldots, |\mathcal{P}_1|\} \times \{1, \ldots, |\mathcal{P}_2|\} \times \ldots \times \{1, \ldots, |\mathcal{P}_{|\mathcal{C}|}|\}$ be a vector of $|\mathcal{C}|$ elements such that the ith element indicates the second index of the lack path of F for C_i; we call $L(F)$ the *lack-path vector* of F. For a lack-path vector $L(F)$, we denote by $L(F, i)$ the number of ith element in $L(F)$. We sometimes consider just a lack-path vector without the corresponding Steiner set. In this case, we use a capital letter like X without specifying a Steiner set, and we denote by $X(i)$ the number of ith element in X. For two lack-path vectors X and Y, let $\delta(X, Y) = |\{i \mid X(i) \neq Y(i)\}|$.

For any Steiner set F of size n_{s}, consider a subset F' of F which consists of all critical vertices and all subpaths for each cycle in $\mathcal{C}$ except for the lack path, that is;

$$F' = K \cup \bigcup_{i=1}^{|\mathcal{C}|} (\bigcup_{j=1}^{|\mathcal{P}_i|} (P_{i,j}) \setminus P_{i,L(F,i)}) \subseteq F.$$

Then, F' forms a minimal Steiner set of G for S; the size of F' may not be n_{s}. Furthermore, for any Steiner sets with the same lack-path vector, the corresponding minimal Steiner sets defined above are identical. We denote by $F^{\mathsf{m}}(X)$ the minimal Steiner set defined above which contained in Steiner sets having the lack-path vector X.

Then, we have the following sufficient condition for reachability.

Lemma 4.7 *Let F_p and F_q be any Steiner sets of size n_{s}. If $L(F_p) = L(F_q)$, then $F_p \leftrightsquigarrow F_q$.*

Proof. We know that there exists the minimal Steiner tree $F^{\mathsf{m}}(L(F_p)) = F^{\mathsf{m}}(L(F_p))$ which is contained in both F_p and F_q. Then we can conclude by Lemma 4.5 that $F_p \leftrightsquigarrow F_q$. □

Canonical Steiner set.

We now give the definition of canonical Steiner sets. To this end, we first introduce a binary relation "$\rightarrow$" and a partial order "$\preceq$" over a set of lack-path vectors. Let X and Y be two lack-path vectors such that $\delta(X, Y) = 1$ where exactly one index $i \in \{1, \ldots, |\mathcal{C}|\}$ satisfies $X(i) < Y(i)$ and every $j \neq i$ satisfy $X(j) = Y(j)$. Then we define a binary relation $\rightarrow$, as follows: $X \rightarrow Y$ if and only if there exist two Steiner sets F_x with $L(F_x) = X$ and F_y with $L(F_y) = Y$ such that $F_x \leftrightarrow F_y$. By using the concept of the binary relation, we introduce a partial order $\preceq$, as follows: For two lack-path vectors X and Y, $X \preceq Y$ if and only if there exists a sequence of $\mathcal{L} = \langle X = X_0, X_1, \ldots, X_\ell = Y \rangle$ of lack-path vectors such that $X_i \rightarrow X_{i+1}$ holds for each $i \in \{0, 1, \ldots, \ell - 1\}$. It is observed from Lemma 4.7 that $\preceq$ is a partial order. For two lack-path vectors X and Y, we say that X and Y are *comparable* if $X \preceq Y$ or $Y \preceq X$ holds, and *incomparable* otherwise. Notice that two Steiner sets F_x and F_y satisfy $F_x \leftrightsquigarrow F_y$ if $L(F_x) \preceq L(F_y)$. We say that a Steiner set F of size n_{s} is *canonical* if its lack-path vector is maximal on the partial order $\preceq$.

In the following, we show that for any Steiner set F, any canonical Steiner sets reachable from F have the same lack-path vector. This implies that F_0 and F_r are reachable if and only if a canonical Steiner set reachable from F_0 and that from F_r have the same lack-path vector. To this end, we first give two lemmas about the binary relation $\rightarrow$.

For any Steiner set F of size n_{s}, we call a vertex which belongs to F but not to $F^{\mathsf{m}}(L(X))$ a *free* vertex of F. It is observed that any two Steiner sets F_x and F_y of size n_{s} satisfying $L(F_x) = L(F_y)$ have the same number of free vertices. We denote by $f(X)$ the number of free vertices of a Steiner set of size n_{s} whose lack-path vector is X. Notice that $f(L(F_x)) \leq f(L(F_y))$ holds for any two Steiner set F_x and F_y such that $F_x \preceq F_y$, since for any $i \in \{1, \ldots, |\mathcal{C}|\}$, we have $X(i) \leq Y(i)$, and hence

$|P_{i,X(i)}| \leq |P_{i,Y(i)}|$. Then, we give the following two lemmas.

Lemma 4.8 *Let X and Y be two lack-path vectors such that $\delta(X,Y) = 1$, where $X(i) < Y(i)$ for the unique index i and $X(j) = Y(j)$ for every $j \neq i$. Then, $X \to Y$ if and only if $f(X) \geq |P_{i,X(i)}| - 1$.*

Proof. We first prove the only-if direction. Suppose that $X \to Y$ holds, that is, there exist two Steiner sets F_x and F_y such that $L(F_x) = X$, $L(F_y) = Y$, and $F_x \leftrightarrow F_y$. Since $X(i) < Y(i)$, we know that $X(i)$ does not take the maximum number (i.e. $X(i) \neq |\mathcal{P}_i|$), and hence the lack path $P_{i,X(i)}$ satisfies $P_{i,X(i)} \nsubseteq F_x$. On the other hand, F_y contains $P_{i,X(i)}$ because the lack path is changed between F_x and F_y. It implies that F_x contains at least $|P_{i,X(i)}| - 1$ vertices in $P_{i,X(i)}$ since $|F_x \setminus F_y| = |F_y \setminus F_x| = 1$. Then such vertices in $P_{i,X(i)}$ are free vertices of F_x, and hence we can conclude that $f(X) \geq |P_{i,X(i)}| - 1$.

We now prove the if direction. Suppose that $f(X) \geq |P_{i,X(i)}| - 1$ holds. Let v_x and v_y be two vertices in $P_{i,X(i)}$ and $P_{i,Y(i)}$, respectively. We then consider any Steiner set F_x of size n_{s} such that $L(F_x) = X$ and F_x contains all vertices in $F^{\mathsf{m}}(X) \cup (P_{i,X(i)} \setminus \{v_x\})$; since $f(X) \geq |P_{i,X(i)}| - 1$, such a Steiner set F_x always exists. We now consider another Steiner set $F_y = F_x \setminus \{v_y\} \cup \{v_x\}$. Then F_y satisfies $L(F_y) = Y$ and $F_x \leftrightarrow F_y$, and hence we have $X \to Y$. □

Lemma 4.9 *Let F_x be a Steiner set. If there exists a Steiner set F_y such that $F_x \leftrightsquigarrow F_y$ and $L(F_x, i) < L(F_y, i)$ for some $i \in \{1, \ldots, |\mathcal{C}|\}$, then there exist a lack-path vector Z such that $L(F_x) \to Z$.*

Proof. Suppose that such a Steiner set F_y exists. Since $F_x \leftrightsquigarrow F_y$, there exists a Steiner set sequence $\mathcal{F} = \langle F_x = F_0, F_1, \ldots, F_\ell = F_y \rangle$ between F_x and F_y. Let F_j be the first Steiner set in $\mathcal{F}$ such that there exists an index $i \in \{1, \ldots, |\mathcal{C}|\}$ satisfying $L(F_x, i) < L(F_j, i)$, that is, for any $j' \in \{1, \ldots, j-1\}$, $L(F_x, i) \geq L(F_{j'}, i)$ holds for every index i; by the definition of F_x and F_y, such a Steiner set F_j always

exists. Since $L(F_x, i) \geq L(F_{j-1}, i)$ holds for every index i, we have $f(L(F_x)) \geq f(L(F_{j-1}))$. We have $L(F_{j-1}) \to L(F_j)$ from the following three facts: (a) $F_{j-1} \leftrightarrow F_j$, (b) $L(F_{j-1}, k) < L(F_j, k)$ for exactly one index k, and (c) $\delta(L(F_j), L(F_{j-1})) = 1$. Then, combining it with Lemma 4.8, we have $f(L(F_{j-1})) \geq |P_{k,L(F_{j-1},k)}| - 1$. Since $L(F_{j-1}, k) \leq L(F_x, k)$, we have $|P_{k,L(F_{j-1},k)}| \geq |P_{k,L(F_x,k)}|$. We then finally obtained that;

$$\begin{aligned} f(L(F_x)) &\geq f(L(F_{j-1})) \\ &\geq |P_{k,L(F_{j-1},k)}| - 1 \\ &\geq |P_{k,L(F_x,k)}| - 1 \end{aligned}$$

Then we can conclude that there exist a desired lack-path vector by Lemma 4.8. □

From the above two lemmas, we have the following desired lemma.

Lemma 4.10 *Let F be any Steiner set of size n_{s}. Then every canonical Steiner sets reachable from F have the same lack-path vectors.*

Proof. Assume for a contradiction that there exist two canonical Steiner sets F_x and F_y such that $F \leftsquigarrow F_x$, $F \leftsquigarrow F_y$, and $L(F_x) \neq L(F_y)$. Then, we know $F_x \leftsquigarrow F_y$ via F. Since $L(F_x) \neq L(F_y)$, we can assume without loss of generality that $L(F_x, i) < L(F_y, i)$ holds for at least one index $i \in \{1, \ldots, |\mathcal{C}|\}$; otherwise, the proof is symmetric. Then, by Lemma 4.9, there exists a lack-path vector Z such that $L(F_x) \to Z$. This contradicts the fact that F_x is canonical. □

Let $L_{\mathsf{c}}(F)$ be the unique lack-path vector of canonical Steiner sets that reachable from F. Then, we have the following corollary.

Corollary 4.1 *Let F and F' be two Steiner sets of size n_{s}. Then $F \leftsquigarrow F'$ if and only if $L_{\mathsf{c}}(F) = L_{\mathsf{c}}(F')$.*

Finding a canonical Steiner set.

We finally show that the auxiliary problem on cactus graphs is polynomial-time solvable. By Corollary 4.1, if we know $L_{\mathsf{c}}(F_0)$ and $L_{\mathsf{c}}(F_r)$, then we can determine whether $F_0 \leftrightsquigarrow F_r$ or not by checking if $L_{\mathsf{c}}(F_0) = L_{\mathsf{c}}(F_r)$ or not. Therefore, the following lemma completes the proof of Theorem 4.5.

Lemma 4.11 *For any Steiner set F, we can compute $L_{\mathsf{c}}(F)$ in polynomial time.*

Proof. If F is canonical, then we have already completed the computation. We now assume that F is not canonical. Considering any canonical Steiner set F_c that reachable from F, the lack-path vector $L(F_c) = L_{\mathsf{c}}(F)$ satisfies $L(F, i) < L(F_c, i)$ for some $i \in \{1, \ldots, |\mathcal{C}|\}$. Therefore there exists another Steiner sets F' such that $L(F) \to L(F')$ by Lemma 4.9. Such a Steiner set F' can be obtained in polynomial time by checking if the condition in Lemma 4.9 for each cycles in $\mathcal{C}$. Thus by greedily applying the above operation finding another Steiner set at most n times, we finally obtain a canonical Steiner set and its lack-path vector $L_{\mathsf{c}}(F)$. □

4.3 Vertex exchange

In this section, we study the complexity of Reach-STR under VE. It is observed that two Steiner trees T and T' are adjacent under VE if and only if the corresponding Steiner sets $V(T)$ and $V(T')$ satisfy $|V(T) \setminus V(T')| = |V(T') \setminus V(T)| \leq 1$. Therefore we obtain the following observation:

Observation 1 *Let T and T' be Steiner trees of a graph G for a terminal set S. Then, $T \overset{\mathsf{VE}}{\leftrightsquigarrow} T'$ holds if and only if there exists a Steiner set sequence between two Steiner sets $V(T)$ and $V(T')$ of G for S.*

We observe that Reach-STR under VE is in PSACE. Therefore, combining Observation 1 and theorems in Section 4.2, we have the following corollary.

Corollary 4.2 Reach-STR *under* VE *is PSPACE-complete for split graphs and planar graphs, even if given two Steiner trees are minimum. On the other hand,* Reach-STR *under* VE *is linear-time solvable for cographs and interval graphs, and polynomial-time solvable for cactus graphs.*

4.4 Local vertex exchange

In this section, we study the complexity of Reach-STR under LVE. The following is the main theorem in this section.

Theorem 4.6 *Let T and T' be Steiner trees of a graph G for a terminal set S. Then, $T \overset{\mathsf{LVE}}{\leftrightsquigarrow} T'$ holds if and only if there exists a Steiner set sequence between two Steiner sets $V(T)$ and $V(T')$ of G for S.*

We observe that Reach-STR under LVE is in PSPACE. Therefore, combining Theorem 4.6 and theorems in Section 4.2, we have the following corollary.

Corollary 4.3 Reach-STR *under* LVE *is PSPACE-complete for split graphs and planar graphs, even if given two Steiner trees are minimum. On the other hand,* Reach-STR *under* VE *is linear-time solvable for cographs and interval graphs, and polynomial-time solvable for cactus graphs.*

In the remainder of this section, we give the proof for Theorem 4.6. We have the only-if direction, as follows. Suppose that there exists a reconfiguration sequence $\langle T = T_0, T_1, \ldots, T_\ell = T' \rangle$ between T and T' under LVE. Then we know that $|V(T_i) \setminus V(T_{i+1})| = |V(T_{i+1}) \setminus V(T_i)| \leq 1$ for each $i \in \{0, 1, \ldots, \ell - 1\}$. Therefore, we can obtain a Steiner set sequence from $V(T)$ and $V(T')$ by removing redundant ones from the sequence $\langle V(T) = V(T_0), V(T_1), \ldots, V(T_\ell) = V(T') \rangle$ of Steiner sets.

We thus show the if direction in the following. Suppose that there exists a Steiner set sequence $\mathcal{F} = \langle V(T) = F_0, F_1, \ldots, F_\ell = V(T') \rangle$. To prove the direction, we show

that for each Steiner set F_i in $\mathcal{F}$, there are two Steiner trees T_i^- with $V(T_i^-) = F_i$ and T_i^+ with $V(T_i^+) = F_i$ which satisfy the following three conditions;

(a) $T = T_0^-$ and $T' = T_\ell^+$;

(b) $T_i^- \overset{\mathsf{LVE}}{\leftrightsquigarrow} T_i^+$ for each $i \in \{0, 1, \ldots, \ell\}$; and

(c) $T_i^+ \overset{\mathsf{LVE}}{\leftrightsquigarrow} T_{i+1}^-$ for each $i \in \{0, 1, \ldots, \ell - 1\}$.

We start with showing the existence of two Steiner trees satisfying the condition (c).

Lemma 4.12 *For any consecutive Steiner sets F_i and F_{i+1} in $\mathcal{F}$, there exists two Steiner trees T_i^+ with $V(T_i^+) = F_i$ and T_{i+1}^- with $V(T_{i+1}^-) = F_{i+1}$ such that $T_i^+ \overset{\mathsf{LVE}}{\leftrightsquigarrow} T_{i+1}^-$.*

Proof. Since F_i and F_{i+1} are consecutive in $\mathcal{F}$, we know $|F_i \setminus F_{i+1}| = |F_{i+1} \setminus F_i| = 1$; let $\{v_i\} = F_i \setminus F_{i+1}$ and $\{v_{i+1}\} = F_{i+1} \setminus F_i$. Notice that $F_i \cap F_{i+1} = F_i \setminus \{v_i\} = F_{i+1} \setminus \{v_{i+1}\}$. Let U be any spanning forest of $G[F_i \cap F_{i+1}]$ such that each connected component is maximal. Since $G[F_i] = G[(F_i \cap F_{i+1}) \cup \{v_i\}]$ is connected, there exists at least one vertex in each (maximal) connected component in U which is adjacent to v_i in $G[F_i]$. Therefore, by adding v_i and edges connecting all connected components with v_i to U, we can obtain a Steiner tree T_i^+ such that $V(T_i^+) = F_i$. Similarly, we can obtain a Steiner tree T_{i+1}^- such that $V(T_{i+1}^-) = F_{i+1}$, by adding v_{i+1} and edges connecting all (maximal) connected components with v_{i+1}. Then, we have $T_i^+[V(T_i^+) \setminus \{v_i\}] = T_{i+1}^-[V(T_{i+1}^-) \setminus \{v_{i+1}\}] = U$. We thus conclude that T_i^+ and T_{i+1}^- are adjacent under LVE, and hence $T_i^+ \overset{\mathsf{LVE}}{\leftrightsquigarrow} T_{i+1}^-$. □

We now notice that T_0^- and T_ℓ^+ are not defined in Lemma 4.12. We then exceptionally define T_0^- as T, and T_ℓ^+ as T'; then the condition (a) is satisfied. We finally show that Steiner trees defined above also satisfy the condition (b); this completes the proof of Theorem 4.6.

Lemma 4.13 *For each $i \in \{0, 1, \ldots, \ell\}$, $T_i^- \overset{\mathsf{LVE}}{\leftrightsquigarrow} T_i^+$ holds.*

Proof. Let i be any index in $\{0, 1, \ldots, \ell\}$. By the definition, we know that $V(T_i^-) =$

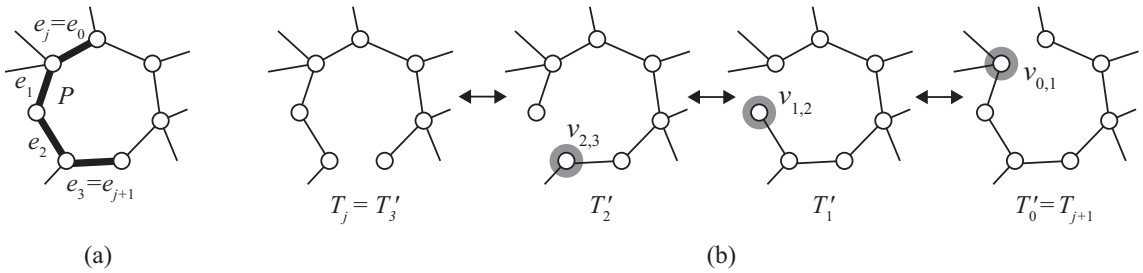


Figure 4.7: (a) An example of a cycle in H with P depicted by thick lines, and (b) the corresponding reconfiguration sequence $\langle T_j = T'_3, T'_2, T'_1, T'_0 = T_{j+1} \rangle$ under LVE between T_j and T_{j+1}.

$V(T_i^+) = F_i$ holds. Then, we know by the known result [18, Proposition 1] that there is a sequence $\langle T_i^- = T_0, T_1, \ldots, T_{\ell'} = T_i^+ \rangle$ between T_i^- and T_i^+ such that $V(T_j) = V(T_{j+1})$ and $|E(T_j) \setminus E(T_{j+1})| = |E(T_{j+1}) \setminus E(T_j)| = 1$ hold for each $j \in \{0, 1, \ldots, \ell' - 1\}$.

Our claim is that $T_j \overset{\mathsf{LVE}}{\leftrightsquigarrow} T_{j+1}$ for each $j \in \{0, 1, \ldots, \ell' - 1\}$. Let e_j be the unique edge in $E(T_j) \setminus E(T_{j+1})$ and e_{j+1} be the unique edge in $E(T_{j+1}) \setminus E(T_j)$. Consider the subgraph H obtained by adding e_{j+1} to T_j, that is, $V(H) = V(T_j)$ and $E(H) = E(T_j) \cup \{e_{j+1}\} = E(T_{j+1}) \cup \{e_j\}$. Then H has exactly one cycle which contains both e_j and e_{j+1}, since T_j and T_{j+1} are trees. (See Figure 4.7(a) as an example of such a cycle.) Therefore H contains exactly two subpaths which start from e_j and end in e_{j+1}; we denote one of such subpaths by the sequence of edges $P = \langle e_j = e_0, e_1, \ldots, e_p = e_{j+1} \rangle$. We can obtain a Steiner tree by removing any single edge in P from H; we denote by T'_q the Steiner tree obtained by removing e_q in P from H. Consider the sequence $\langle T_j = T'_p, T'_{p-1}, \ldots, T'_0 = T_{j+1} \rangle$ of Steiner trees. (See Figure 4.7(b) as an example of such a sequence corresponding to the path P in (a).) Then, any consecutive Steiner trees T'_q, T'_{q-1} in the sequence are adjacent under LVE, since $T'_q[V(T'_q) \setminus \{v_{q,q-1}\}] = T'_{q-1}[V(T'_{q-1}) \setminus \{v_{q,q-1}\}]$ holds where $v_{q,q-1}$ is the common endpoint of e_q and e_{q-1}. Therefore, we can conclude that $T_j \overset{\mathsf{LVE}}{\leftrightsquigarrow} T_{j+1}$ for each $j \in \{0, 1, \ldots, \ell' - 1\}$; this completes the proof of Lemma 4.13. □

4.5 Local vertex exchange without changing neighbors

In this section, we show that Reach-STR under LVE-N is solvable in polynomial time for cographs, chordal graphs, and for planar graphs if given two Stiner trees are minimum. To this ends, we introduce the concept of Steiner tree embeddings and their reconfiguration, which gives a necessary condition for the existence of a reconfiguration sequence under LVE-N.

4.5.1 Steiner tree embeddings and their reconfiguration

We first introduce the concept of Steiner tree embeddings. Let T be a Steiner tree of a graph G for a terminal set S. An injection $\varphi\colon V(T) \to V(G)$ is called a *T-embedding* into G if the following two conditions hold:

- $\varphi(x)\varphi(y) \in E(G)$ if $xy \in E(T)$; and
- $\varphi(s) = s$ holds for each $s \in S \subseteq V(T)$.

Thus, a T-embedding φ *defines* a Steiner tree T_φ of G for S. Observe that no two distinct T-embeddings define the same Steiner tree. A Steiner tree T' is said to be *T-embeddable* if there exists a T-embedding φ which defines T'. Note that T itself is T-embeddable. We write $\varphi(V') = \{\varphi(x) \mid x \in V'\}$ for any node subset $V' \subseteq V(T)$. We now give the following lemma.

Lemma 4.14 *Let T_a and T_b be any two Steiner trees of a graph G for a terminal set S. If $T_a \overset{\text{LVE-N}}{\leftrightsquigarrow} T_b$, then T_b is T_a-embeddable.*

Proof. Let $\mathcal{T} = \langle T_a = T_0, T_1, \ldots, T_\ell = T_b \rangle$ be a reconfiguration sequence between T_a and T_b under LVE-N. Recall again that $T_a = T_0$ is T_a-embeddable. Then, we observe from the definition of LVE-N that T_1 is also T_a-embeddable. Considering the same observation for each consecutive Steiner trees in $\mathcal{T}$, we know that T_b is T_a-embeddable. □

By taking a contrapositive of Lemma 4.14, we can conclude that a given instance (G, S, T_0, T_r) is a no-instance if T_r is not T_0-embeddable; this can be checked in polynomial time. Thus, in the remainder of this section, we assume without loss of generality that T_r is T_0-embeddable.

We then introduce the reconfiguration of Steiner tree embeddings. Let T be a Steiner tree of a graph G for a terminal set S. We say that two T-embeddings φ and φ' are *adjacent* if exactly one node in T is mapped into different vertices between φ and φ', that is, $|\{x \in V(T) \mid \varphi(x) \neq \varphi'(x)\}| = 1$ holds. For two T-embeddings φ and φ', an *embedding sequence* between φ and φ' is a sequence $\langle \varphi = \varphi_0, \varphi_1, \ldots, \varphi_\ell = \varphi' \rangle$ of T-embeddings such that φ_i and φ_{i+1} are adjacent for each $i \in \{0, 1, \ldots, \ell - 1\}$. We write $\varphi \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi'$ if there exists an embedding sequence between φ and φ'. Then, we have the following lemma.

Lemma 4.15 *Suppose that T_a and T_b are any two Steiner trees of a graph G for a terminal set S such that T_b is T_a-embeddable. Let φ_a and φ_b be T_a-embeddings which define T_a and T_b, respectively. Then, $T_a \overset{\mathsf{LVE\text{-}N}}{\leftrightsquigarrow} T_b$ if and only if $\varphi_a \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi_b$.*

Proof. Suppose that there exists a embedding sequence $\langle \varphi_a = \varphi_0, \varphi_1, \ldots, \varphi_\ell = \varphi_b \rangle$ between φ_a and φ_b. Then we can construct the reconfiguration sequence $\langle T_a = T_0, T_1, \ldots, T_\ell = T_b \rangle$ between T_a and T_b, where for each $i \in \{0, 1, \ldots, \ell\}$, T_i is the Steiner tree that φ_i defines. On the other hand, suppose that there exists a reconfiguration sequence $\langle T_a = T_0, T_1, \ldots, T_\ell = T_b \rangle$ between T_a and T_b. Since each T_i in the sequence is reachable from T_0, we know that T_i is T_0-embeddable by Lemma 4.14, and hence there is a T_0-embedding φ_i. Then the sequence $\langle \varphi_a = \varphi_0, \varphi_1, \ldots, \varphi_\ell = \varphi_b \rangle$ is a desired embedding sequence. □

By Lemmas 4.14 and 4.15, Reach-STR under LVE-N can be rephrased to the following problem: Given a graph G, a terminal set S, a Steiner tree T (actually T_0), and two T-embeddings φ_0 and φ_r into G, we are asked to determine whether or not there exists an embedding sequence between φ_0 and φ_r. Therefore, we also

denote by $(G, S, T, \varphi_0, \varphi_r)$ an instance of Reach-STR under LVE-N.

4.5.2 Layers for Steiner trees

We here introduce one more important concept, called layers, which was originally introduced by Bonsma [4] for shortest path reconfiguration. Bonsma [4] introduced the natural concept of layers for a shortest path. We generalize the concept to Steiner trees.

Let T be a Steiner tree of a graph G for a terminal set S. For each $x \in V(T)$, let $L_T(x) = \{\varphi(x) \in V(G) \mid \varphi \text{ is a } T\text{-embedding into } G\}$; we call $L_T(x)$ the *layer* of x. Notice that $L_T(s) = \{s\}$ holds for each $s \in S$. We write $L_T(V') = \bigcup_{x \in V'} L_T(x)$ for any node subset $V' \subseteq V(T)$. Then, we have the following property, which says that the layers are disjoint.

Lemma 4.16 *Let T be any Steiner tree of a graph G for a terminal set S. If T is minimum, then $L_T(x) \cap L_T(y) = \emptyset$ holds for any two distinct nodes $x, y \in V(T)$.*

Proof. Suppose that T is minimum. Assume for a contradiction that $L_T(x) \cap L_T(y) \neq \emptyset$ holds for some two distinct nodes $x, y \in V(T)$, that is, there exists a vertex $u \in L_T(x) \cap L_T(y)$. Then there exist two T-embeddings φ_1 and φ_2 such that $\varphi_1(x) = u$ and $\varphi_2(y) = u$, respectively.

Consider the unique subpath between x and y in T. By removing any edge in the subpath from T, we obtain two subtrees of T; note that one of them contains x and the other one contains y. Let T_x and T_y be such subtrees of T containing x and y, respectively. Suppose that G' is the subgraph of G induced by $\varphi_1(V(T_x)) \cup \varphi_2(V(T_y))$. Then we have $S \subseteq V(G')$, since $S \subseteq V(T_x) \cup V(T_y)$. Furthermore, we know that G' is connected, since both $G[\varphi_1(V(T_x))]$ and $G[\varphi_2(V(T_y))]$ are connected and contain the same vertex u. Therefore, any spanning tree T' of G' is a Steiner tree of G for S. Then, since $u \in \varphi_1(V(T_x)) \cap \varphi_2(V(T_y))$, we have $|V(T')| = |V(G')| <$

$|\varphi_1(V(T_x))| + |\varphi_2(V(T_y))| = |V(T_x)| + |V(T_y)| = |V(T)|$; this contradicts the fact that T is a minimum Steiner tree of G for S. □

4.5.3 Polynomial-time algorithm for cographs

In this subsection, we show that Reach-STR under LVE-N is polynomial-time solvable for cographs. We first prove the following lemma.

Lemma 4.17 *Suppose that G is a cograph and $S \subseteq V(G)$ is a terminal set. Then any minimum Steiner tree T of G for S contains at most one non-terminal node.*

Proof. We can assume without loss of generality that G is connected cograph. It is known that for any connected cograph G, the vertex subset can be partitioned into two non-empty vertex subsets $A \subset V(G)$ and $B = V(G) \setminus A$ such that any two vertices $a \in A$ and $b \in B$ are adjacent.

We first consider the case where either $S \subseteq A$ or $S \subseteq B$ holds; we here consider only the case where $S \subseteq A$, since the proof is symmetric otherwise. In this case, we can construct a Steiner tree whose vertex set is $S \cup \{b\}$ for any vertex in B, because b is adjacent to all vertices in S (actually, A); such b always exists since $B \neq \emptyset$. Thus the lemma follows.

We then consider the remaining case where both A and B contains at least one terminal. In this case, a terminal in A is adjacent to all terminals in B, and a terminal in B is adjacent to all terminals in A. Thus, there exists a (minimum) Steiner tree such that all nodes are terminals. Therefore, the lemma follows. □

The following is the main theorem in this subsection.

Theorem 4.7 Reach-STR *under* LVE-N *is polynomial-time solvable for cographs.*

Proof. From Lemma 4.14, we can assume without loss of generality that given two minimum Steiner trees T_0 and T_r are T_0 embeddable; otherwise, it is a no-instance and it can be checked in polynomial time. Furthermore, we know from Lemma 4.17

that these two minimum Steiner trees contain at most one non-terminal vertices. Then we observe that $T_0 \overset{\mathsf{LVE\text{-}N}}{\leftrightarrow} T_r$ holds since they are T_0-embeddable. Thus the theorem follows. □

4.5.4 Polynomial-time algorithm for chordal graphs

In this subsection, we show the following theorem.

Theorem 4.8 Reach-STR *under* LVE-N *is polynomial-time solvable for chordal graphs if given two Steiner trees are minimum.*

In our proof, we deal with the well-known characterization of chordal graphs in terms of perfect elimination orderings [8]. For a graph G of n vertices, an ordering of vertices $\langle v_1, v_2, \ldots, v_n \rangle$ is called a *perfect elimination ordering* if for each $i \in \{1, 2, \ldots, n\}$, the closed neighbor of v_i in $G_i = G[\{v_i, v_{i+1}, \ldots, v_n\}]$ induces a clique in G_i. It is known that a graph is chordal if and only if it has a perfect elimination ordering.

Let $(G, S, T, \varphi_0, \varphi_r)$ be an instance of Reach-STR under LVE-N, where G is chordal and T is a minimum Steiner tree of G for S. Then, Theorem 4.8 is obtained by the following lemma.

Lemma 4.18 *There always exists an embedding sequence between φ_0 and φ_r.*

Proof. Let $V_0 = \varphi_0(V(T))$ and $V_r = \varphi_r(V(T))$. Suppose that k is the size of the symmetric difference of V_0 and V_r, that is, $k = |V_0 \Delta V_r| = |(V_0 \setminus V_r) \cup (V_r \setminus V_0)|$. We show the lemma by induction of k.

If $k = 0$, then we have $V_0 = V_r$. Recall that any T-embedding contains exactly one vertex from each layer and any two layers are disjoint from Lemma 4.16. Therefore, we know that $\varphi_0 = \varphi_r$, and hence the lemma follows.

We now consider the case where $k \geq 1$. Suppose that G' is the subgraph of G induced by $V_0 \cup V_r$; since G is chordal, G' is also chordal. Let $\mathcal{P} = \langle v_1, v_2, \ldots, v_{n'} \rangle$ be

a perfect elimination ordering of G', where n' is the number of vertices in G'. Then we focus on the first vertex v_p in $\mathcal{P}$ which contained in $V_0 \Delta V_r$, that is, $v_p \notin V_0 \cap V_r$ and $v_i \in V_0 \cap V_r$ for any $i < p$. We assume without loss of generality that $v_p \in V_0 \setminus V_r$; otherwise, our proof is symmetric. Let $x \in V(T)$ be the node such that $\varphi_0(x) = v_p$. Notice that $\varphi_r(x) \neq \varphi_0(x) = v_p$ by the definition of v_p; let $v_q = \varphi_r(x)$. Then we know $p < q$. Roughly speaking, our claim is that we can exchange the mapping of x in φ_0 from v_p to v_q. Formally, our claim is the following (the correctness will be presented later):

Claim *There exists a T-embedding φ'_0 such that $\varphi'_0(y) = \varphi_0(y)$ for each $y \neq x$ and $\varphi'_0(x) = \varphi_r(x) = v_q$.*

We observe that φ_0 and φ'_0 are adjacent. Furthermore, for $V'_0 = \varphi'_0(V(T))$, we have $|V'_0 \Delta V_r| = k - 2$ holds. Therefore, we know by the induction hypothesis that φ'_0 and φ_r are reconfigurable; then the lemma follows.

We finally give the correctness of the claim. It suffices to show that for each neighbor $y \in N_T(x)$ of x in T, $\varphi_0(y)$ is adjacent to $\varphi_r(x) = v_q$ in G' (then also in G). Let $\varphi_0(y) = v_i$. We then consider separately in the following two cases.

- If $i < p$, then we know $v_i \in V_0 \cap V_r$, that is, $\varphi_0(y) = \varphi_r(y) = v_i$. Thus, we know that v_i is adjacent to v_q in G'.
- We know consider the case where $i > p$. In this case, we again focus on the vertices faster than v_i in $\mathcal{P}$. We first show that there exists at least one neighbor z of x in T such that $\varphi_0(z)$ appears faster than v_p in $\mathcal{P}$. Assume for a contradiction that such a neighbor does not exist. Then for all neighbors $z \in N_T(x)$, $\varphi_0(z)$ appears after v_p in $\mathcal{P}$, and hence $\varphi_0(z)$ is contained in $G'_p = G'[\{v_p, v_{p+1}, \ldots, v_n\}]$. Therefore, we know by the definition of a perfect elimination ordering that the vertex subset $\varphi_0(N_T(x))$ forms a clique in G'_p (and hence also in G'). Since we know v_p is not a terminal (because $v_p \notin V_t$), we can construct a Steiner tree in G' (and hence in G) whose vertex set is $V_s \setminus \{v_p\}$; this contradicts the fact that the

Steiner tree defined by φ_0 is minimum. We thus know that there is a neighbor $z \in N_T(x)$ such that $\varphi_0(z) = v_j$ with $j < p$. Then, since both v_p and v_q are adjacent to v_j and they are contained in $G'_j = G'[\{v_j, v_{j+1}, \ldots, v_{n'}\}]$, there is an edge between v_p and v_q in G'_j (and hence also in G'). Furthermore, since v_q and v_i are adjacent to v_p and they are contained in $G'_p = G'[\{v_p, v_{p+1}, \ldots, v_{n'}\}]$, we can conclude that there is an edge between v_q and v_i. This completes the proof for the claim.

In this way, we obtain the correctness of the claim. □

4.5.5 Polynomial-time algorithm for planar graphs

In this subsection, we give the following theorem.

Theorem 4.9 Reach-STR *under* LVE *is polynomial-time solvable for planar graphs if given two Steiner trees are minimum.*

As a proof of the Theorem 4.9, we construct a polynomial-time algorithm to solve the problem. Roughly speaking, our idea is to decompose a given instance of Reach-STR under LVE-N into several shortest path reconfiguration (SPR, for short) instances for planar graphs. (See Subsection 3.4.1 for the definition of SPR). Then, we can solve each SPR instance by using the polynomial-time algorithm for SPR on planar graphs [5]. Finally, we combine the answers to SPR instances, and output the answer to the original Reach-STR instance under LVE-N.

Computing actual layers.

To decompose a given instance of Reach-STR under LVE-N to instances, we need to compute actual layers for a give instance. Then we need some additional properties of layers.

We first give the following property of layer; note that the property holds for general graphs.

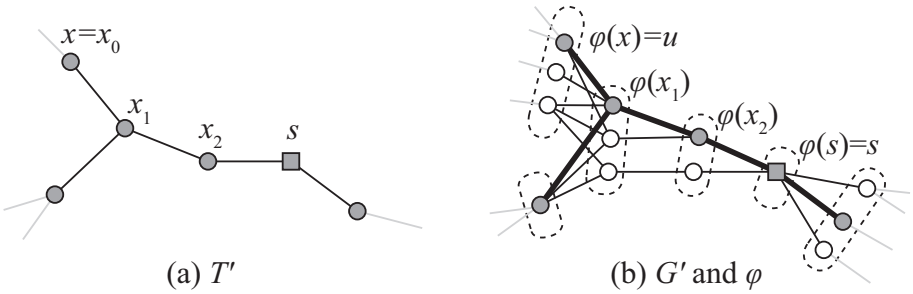


Figure 4.8: An example of Lemma 4.19, where (a) a subtree T', and (b) G' corresponding to T' and an embedding φ such that $\varphi(x) = u$.

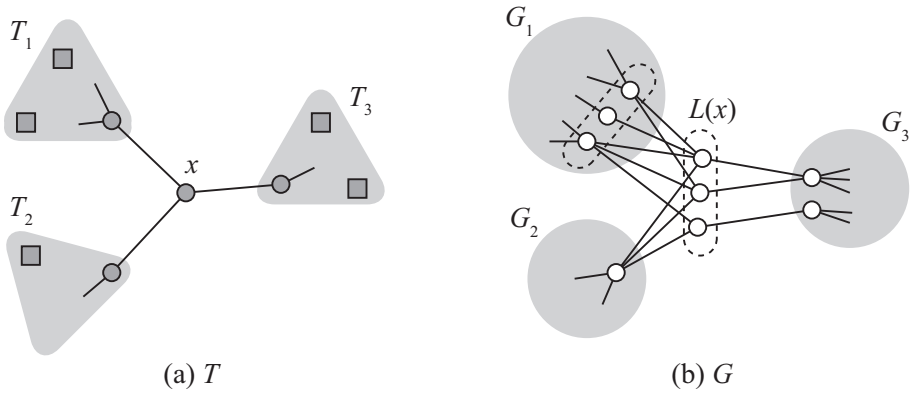
Lemma 4.19 *Let T be a Steiner tree of a graph G for a terminal set S, and T' be any subtree of T containing at least one terminal. Then the subgraph of G induced by $L_T(V(T'))$ is connected.*

Proof. Let G' be the subgraph of G induced by $L_T(V(T'))$, and let $s \in S$ be any terminal contained in T'. Then it suffices to show that for any vertex $u \in V(G')$, there exists a path between s and u in G'. Let $u \in V(G')$ be any vertex in G', and let $x \in V(T')$ be the node such that $u \in L_T(x)$. (See Figure 4.8.) Since $x, s \in V(T')$, T' includes the unique path $\langle x = x_0, x_1, \ldots, x_\ell = s \rangle$ between x and s. Then, by the definition of G', $L_T(x_i) \subseteq V(G')$ holds for each $i \in \{0, 1, \ldots, \ell\}$. Consider any T-embedding φ such that $\varphi(x) = u$. Then we know that $\varphi(x_i) \in L_T(x_i) \subseteq V(G')$ for each $i \in \{0, 1, \ldots, \ell\}$, and $\varphi(x_i)\varphi(x_{i+1}) \in E(G')$ for each $i \in \{0, 1, \ldots, \ell - 1\}$. Note that $\varphi(s) = s$. In this way, we can conclude that there exists a path between s and u in G'. □

For a Steiner tree T, we call a node $x \in V(T)$ a *branching node* of T if $|N_T(x)| \geq 3$. Let $B(T)$ be the set of all branching nodes of T. Then, we show that a layer of each node in $B(T)$ contains at most two vertices if a given graph is planar.

Lemma 4.20 *Let T be any minimum Steiner tree of a graph G for a terminal set S. If G is planar, then $|L_T(x)| \leq 2$ holds for every branching node $x \in B(T)$.*

Proof. Suppose for a contradiction that $|L_T(x)| \geq 3$ holds for some node $x \in B(T)$. Since x has degree at least three in T, we obtain at least three subgtrees by removing



(a) T (b) G

Figure 4.9: An example of Lemma 4.20, where (a) T with a branching node x, and (b) G with subgraphs G_1, G_2, and G_3.

x from T; let T_1, T_2, T_3 be any three among such subtrees. (See Figure 4.9.) By the minimality of T, each of T_1,T_2,T_3 contains at least one terminal. Therefore, considering the subgraphs G_1, G_2, G_3 of G induced by $L_T(V(T_1))$,$L_T(V(T_2))$,$L_T(V(T_3))$, respectively, then each of them is connected by Lemma 4.19. Since the vertices of G_1, G_2, G_3 are disjoint by Lemma 4.16, we obtain three distinct vertices u_1, u_2, u_3 by contracting all edges in G_1, G_2, G_3, respectively. Then, since u_1, u_2, u_3 are adjacent to all vertices in $L_T(x)$ and $|L_T(x)| \geq 3$ holds, we know that G has a complete bipartite graph $K_{3,3}$ as a minor; this contradicts that G is planar. □

We now explain how to compute the layers for a Steiner tree. In SPR [4], we can easily find the layers for a shortest path by computing the distances from the two terminals to each vertex in the underlying graph. This is because the subpath between each node and each terminal is always a shortest path in the underlying graph. On the other hand, this property does not always hold if $|S| \geq 3$, and hence it is difficult to find the layers simply by computing the distances. (For example, see Figure 4.10.)

Our idea is to compute the "refined" layers for a Steiner tree, instead of computing the layers completely. Let $(G, S, T, \varphi_0, \varphi_r)$ be a given instance of Reach-STR under LVE-N. Then, for all nodes $x \in V(T)$, it suffices to find vertex subsets $L'_T(x)$ such that

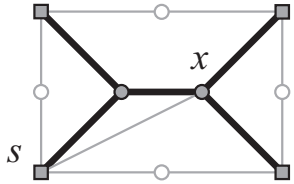


Figure 4.10: The subtree T depicted by thick lines is a minimum Steiner tree. However, the subpath on T between s and x is not a shortest path in the underlying graph.

(a) $L'_T(x) \subseteq L_T(x)$; and

(b) $\varphi(x) \in L'_T(x)$ holds for any T-embedding φ satisfying $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi$ or $\varphi_r \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi$.

To avoid a confusion, we call such a vertex subset $L'_T(x)$ the *refined-layer* of x, while call the (original) layer $L_T(x)$ the *complete-layer* of x. We know that the vertices in $L_T(x) \setminus L'_T(x)$ are useless when we want to check if $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi_r$ or not. The following lemma says that the refined-layers can be found in polynomial time.

Lemma 4.21 *Let $(G, S, T, \varphi_0, \varphi_r)$ be a given instance of* Reach-STR *under* LVE-N *such that G is a planar graph. Then, there exists a polynomial-time algorithm to compute the refined-layers for all nodes in T.*

We prove the lemma by giving an actual algorithm which computes the refined-layers. We say that two nodes $x, y \in B(T) \cup S$ are *close* if the unique path on T between x and y contains no vertex in $(B(T) \cup S) \setminus \{x, y\}$. To avoid the duplication of $\{x, y\}$ and $\{y, x\}$, we choose one of the ordered pairs (x, y) and (y, x) arbitrarily for each pair of close nodes, and define the set $C(T)$ of all ordered pairs (x, y) of close nodes x, y in $B(T) \cup S$; we call each pair in $C(T)$ a *close pair*. Then, the following is an algorithm finding the refined-layer for each node in T; its correctness will be proved later.

1: Compute $\mathsf{dist}_G(u, v)$ for all pairs of vertices $u, v \in V(G)$.

(Initializing each node in $V(T)$ and finding a refined-layer for each node in S)

2: Initialize $L'_T(x) \leftarrow \{\varphi_0(x), \varphi_r(x)\}$ for each node $x \in V(T)$, and $B' \leftarrow \{x \in$

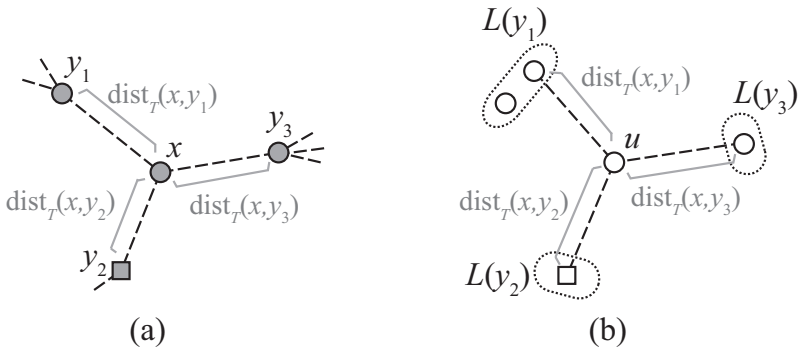


Figure 4.11: An example of Line 3 in the algorithm, where (a) a branching node x with its close nodes y_1, y_2, and y_3, and (b) a vertex u we look for in Line 3.

$B(T) \mid |L'_T(x)| < 2\}$.

(Finding a refined-layer for each node in $B(T)$)

3: For each node $x \in B'$, look for a vertex $u \in V(G) \setminus L'_T(x)$ such that there exists a vertex $v \in L'_T(y)$ for each close node y of x satisfying $\mathsf{dist}_G(u, v) = \mathsf{dist}_T(x, y)$. If such a vertex u exists, then update $L'_T(x) \leftarrow L'_T(x) \cup \{u\}$ and $B' \leftarrow B' \setminus \{x\}$.

4: Repeat Line **3** until $B' = \emptyset$ holds or updating of $L'_T(x)$ and B' does not occur for any node $v \in B'$.

(Finding a refined-layer for each node in $V(T) \setminus (B(T) \cup S)$)

5: For each node $x \in V(T) \setminus (B(T) \cup S)$, let $y, z \in B(T) \cup S$ be close nodes such that the unique path between y and z in T contains x. Then look for a vertex $u \in V(G) \setminus L'_T(x)$ such that there exist vertices $v \in L'_T(y)$ satisfying $\mathsf{dist}_G(u, v) = \mathsf{dist}_T(x, y)$ and $w \in L'_T(z)$ satisfying $\mathsf{dist}_G(u, w) = \mathsf{dist}_T(x, z)$. For all such vertices u, update $L'_T(x) \leftarrow L'_T(x) \cup \{u\}$.

6: Output $L'_T(x)$ for each node $x \in V(T)$.

Notice that we can compute the refined-layer $L'_T(s)$ (actually, the complete-layer $L_T(s)$) for each $s \in S$ in Line 2, since $L_T(s) = \{s\}$ and $\varphi_0(s) = \varphi_r(s) = s$. Figure 4.11 and 4.12 illustrate examples of a vertex u which we should find in Line 3 and 5, respectively.

Lemma 4.22 *The algorithm runs in polynomial time.*

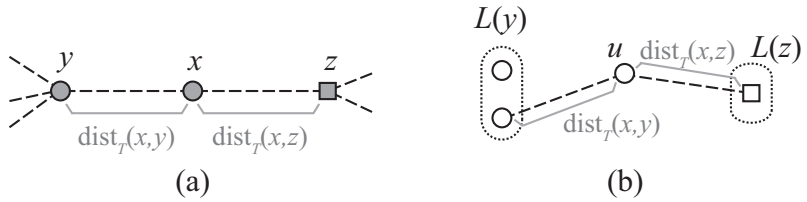


Figure 4.12: An example of Line 5 in the algorithm, where (a) a node x with close nodes y and z such that the unique path y and z contains x, and (b) a vertex u we look for in Line 5.

Proof. Let n be the number of vertices in G. It is observed that Line 1 can be done in $O(n^3)$ time, and Line 2 and 6 can be done in linear time. Thus we now refer to Line 3, 4, and 5. In Line 3, we have $|B'| \leq n$. For each node $x \in B'$, candidates which may be included in $L_T(x)$ by updating are at most n. We can check in time $O(n)$ whether each candidate satisfies the condition to belong to $L'_T(x)$. We thus conclude that Line 3 can be done in time $O(n^3)$. In Line 4, for each updating in Line 3, the number of vertices in B' decreases exactly one. Thus the total time for Line 3, 4 is $O(n^4)$. In Line 5, the number of vertices in $V(T) \setminus (B(T) \cup S)$ is at most n. For each node $x \in V(T) \setminus (B(T) \cup S)$, candidates which may be included in $L_T(x)$ by updating are at most n. We can check in time $O(n)$ whether each candidate satisfies the condition to belong to $L'_T(x)$. We thus conclude that Line 5 can be done in time $O(n^3)$. In this way, we can conclude that the running time of the algorithm is $O(n^4)$. □

The following ensures that each vertex subset obtained by the algorithm is a relevant refined-layer.

Lemma 4.23 *For any node $x \in V(T)$, the vertex subset $L'_T(x)$ obtained by the algorithm satisfies the following two conditions*:

(a) $L'_T(x) \subseteq L_T(x)$; *and*

(b) *for any T-embedding φ satisfying $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi$ or $\varphi_r \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi$, $\varphi(x) \in L'_T(x)$.*

Proof. We first prove the condition (a). In Line 2, we observe that $\{\varphi_0(x), \varphi_r(x)\} \subseteq L_T(x)$ holds for any node $x \in V(T)$. Thus $L'_T(x)$ for each $x \in V(T)$ obtained in

Line 2 satisfies the condition (a). Therefore, it suffices to show that for each updating $L'_T(x) \Leftarrow L'_T(x) \cup \{u\}$ in Line 3, 5, if all current layers satisfy (a), then the vertex u is also contained in $L_T(x)$. Suppose that updating $L'_T(x) \Leftarrow L'_T(x) \cup \{u\}$ occurs for some node x and vertex u in Line 3 or 5. Then we give an actual T-embedding φ such that $\varphi(x) = u$. Let $Y \subseteq V(T)$ be a set of all close nodes of x for Line 3, or a set of two close nodes such that the unique path between them in T contains x for Line 5. By the operation of the algorithm, for any node $y \in Y$, $L'_T(y)$ contains a node v such that there exists a path between u and v in G with length $\mathsf{dist}_T(x, y)$. We know that such paths are vertex-disjoint; otherwise we can construct a Steiner tree with edge set smaller than T, contradicting the minimality of T. Then, φ is constructed as follows. For each node $y \in Y$, we first set a path $\langle u = u_0, u_1, \ldots, u_{\mathsf{dist}_T(x,y)} = v \rangle$ between u and v in G as an embedding of a path $\langle x = x_0, x_1, \ldots, x_{\mathsf{dist}_G(u,v)} = y \rangle$ between x and y in T of φ, that is, $\varphi(x_i) = u_i$ for each $i \in \{0, 1, \ldots, \mathsf{dist}_T(x, y)\}$. Consider the unique path between x and y in T, and remove the edge incident to y in the path from T. Then we obtain two subtrees of T; let T_y be one of the subtrees containing y. Since $u \in L'_T(y)$, there exists a T-embedding φ' such that $\varphi'(y) = u$. We then set $\varphi(w) = \varphi'(w)$ for each node $w \in V(T_y)$. Applying these operation to each node $y \in Y$, we obtain a desired T-embedding φ.

We then prove the condition (b). We prove the claim only for the case of $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi$; the other case of $\varphi_r \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi$ can be proved by similar way. Suppose that the condition (b) does not hold. Let Φ be a set of all T-embeddings such that they are reconfigurable from φ_0 and contain a vertex which does not belong to $L'_T(x)$ for some node $x \in V(T)$. Let $\varphi \in \Phi$ be any T-embedding such that there exists a embedding sequence $\langle \varphi_0 = \varphi_0, \varphi_1, \ldots, \varphi_\ell = \varphi \rangle$ where $\varphi_i \notin \Phi$ holds for each $i < \ell$. Suppose that $x \in V(T)$ is a node such that $\varphi(x) \notin L'_T(x)$. Let $Y \subseteq V(T)$ be a set of all close nodes of x if $x \in B(T) \cup S$, and otherwise a set of two close nodes such that x is contained in the unique path between them in T. Then by the existence

of φ, we know that for each $y \in Y$, there exists a path between $\varphi(x)$ and $\varphi(y)$ in G of length $\mathsf{dist}_T(x, y)$. Furthermore, by the definition of φ, we have $\varphi(y) \in L'_T(y)$ for any node $y \in V(T) \setminus \{x\}$. Therefore, the algorithm can find $\varphi(x)$, contradicting $\varphi(x) \notin L'_T(x)$. □

Given an instance $(G, S, T, \varphi_0, \varphi_r)$ of Reach-STR under LVE-N, we can compute refined-layers in polynomial time by Lemma 4.21. Since the vertices in $L_T(x) \setminus L'_T(x)$, $x \in V(T)$, are useless, we can remove such useless vertices from G. In this way, we can assume without loss of generality that each vertex in G belongs to exactly one (complete-)layer, and we indeed know the layer $L_T(x)$ for each node $x \in V(T)$.

Decomposition of a given instance into SPR instances.

Suppose that $(G, S, T, \varphi_0, \varphi_r)$ is an instance of Reach-STR under LVE-N, and that we have the layer $L_T(x)$ for each node $x \in V(T)$. For each close pair (x, y) in $C(T)$, we now construct the corresponding instance $\mathrm{SPR}(x, y) = (G', S', T', \varphi'_0, \varphi'_r)$ such that $|S'| = 2$, as follows. Let P be the unique path on T between x and y. Note that by the definition of close pairs, P is a shortest path on G between x and y. Consider the subgraph of G induced by the vertices in $L_T(V(P))$. We add two new vertices s_x and t_y to the subgraph so that s_x is joined to all vertices in $L_T(x)$ and t_y is joined to all vertices in $L_T(y)$; note that each of s_x and t_y is indeed adjacent to one or two vertices. Let G' be the resulting graph, and let $S' = \{s_x, t_y\}$. We then define T' as the path on G' between s_x and t_y obtained by adding s_x and t_y to P. Note that T' is a shortest path on G' between s_x and t_y. We finally define φ'_0 as a T'-embedding into G' such that $\varphi'_0(s_x) = s_x$, $\varphi'_0(t_y) = t_y$, and $\varphi'_0(x) = \varphi_0(x)$ for each $x \in V(P)$. Similarly, we define φ'_r as a T'-embedding into G' such that $\varphi'_r(s_x) = s_x$, $\varphi'_r(t_y) = t_y$, and $\varphi'_r(x) = \varphi_r(x)$ for each $x \in V(P)$. This completes the construction of $\mathrm{SPR}(x, y)$. The corresponding instance $\mathrm{SPR}(x, y)$ can be obtained in polynomial time, and satisfies the following property.

Lemma 4.24 *If G is planar, then G' is also planar.*

Proof. We know that $G[L_T(V(P))]$ is planar because it is a subgraph of G. If $|L_T(x)| = 1$ (resp. $|L_T(y)| = 1$) holds, then added vertex s (resp. t) has degree exactly one. Therefore adding s (resp. t) does not destroy the planarity.

We thus suppose that $|L_T(x)| = 2$ (Notice that $|L_T(x)| > 2$ does not occur since x is a branching node); the case of $|L_T(y)| = 2$ can be proved by similar way. In this case, we show that s can be obtained by contracting some edges in G. Since $|L_T(x)| = 2$, we know that x is an internal node of T; if x is a leaf of T, x must be a terminal by minimality of T, and hence $|L_T(x)| = 1$. Therefore, by removing x from T, we obtain at least two subtrees of T. Let T'' be one of the subtrees which does not contain y. By the minimality of T, we know that T'' contains at least one terminal. Thus, by Lemma 4.19, $G[L_T(V(T''))]$ is connected, and hence we obtain a single vertex by contracting all edges in $G[L_T(V(T''))]$. Furthermore, the obtained vertex is adjacent to two vertices in $L_T(x)$. Therefore, we can conclude that s can be obtained by contracting some edges in G, and hence G' is planar. □

Determination of the reachability.

We finally determine the reachability between two given minimum Steiner trees. By Lemma 4.24 we can solve the instance $\mathrm{SPR}(x, y)$ for each close pair $(x, y) \in C(T)$ by the polynomial-time algorithm for SPR on planar graphs [5]. We can immediately conclude that the given instance $(G, S, T, \varphi_0, \varphi_r)$ of Reach-STR under LVE-N is a no-instance if there exists at least one instance $\mathrm{SPR}(x, y)$ whose answer is no. However, even if the answers are yes to all instances $\mathrm{SPR}(x, y)$, $(x, y) \in C(T)$, it is not always possible to extend their embedding sequences to a whole embedding sequence between φ_0 and φ_r for the original instance $(G, S, T, \varphi_0, \varphi_r)$. To check this, we introduce further notion.

Consider an embedding sequence $\mathcal{R} = \langle \varphi = \varphi_0, \varphi_1, \ldots, \varphi_\ell = \varphi' \rangle$ between two

T-embeddings φ and φ'. For each node $x \in V(T)$, we say that $\mathcal{R}$ is *x-touching* if the assignment of x is changed by $\mathcal{R}$ at least once; otherwise it is *x-untouching*. Note that if $\varphi(x) \neq \varphi'(x)$ for a node $x \in V(T)$, then any embedding sequence between φ and φ' must be x-touching. On the other hand, if $|L_T(x)| = 1$, then any embedding sequence must be x-untouching. For each close pair $(x, y) \in C(T)$ and its corresponding instance $\mathrm{SPR}(x, y) = (G', S', T', \varphi'_0, \varphi'_r)$, we define the set $\mathsf{Touch}(x, y) \subseteq \{(\mathsf{u}, \mathsf{u}), (\mathsf{u}, \mathsf{t}), (\mathsf{t}, \mathsf{u}), (\mathsf{t}, \mathsf{t})\}$, as follows:

- $(\mathsf{u}, \mathsf{u}) \in \mathsf{Touch}(x, y)$ if and only if there exists an embedding sequence between φ'_0 and φ'_r which is x-untouching and y-untouching;
- $(\mathsf{u}, \mathsf{t}) \in \mathsf{Touch}(x, y)$ if and only if there exists an embedding sequence between φ'_0 and φ'_r which is x-untouching and y-touching;
- $(\mathsf{t}, \mathsf{u}) \in \mathsf{Touch}(x, y)$ if and only if there exists an embedding sequence between φ'_0 and φ'_r which is x-touching and y-untouching; and
- $(\mathsf{t}, \mathsf{t}) \in \mathsf{Touch}(x, y)$ if and only if there exists an embedding sequence between φ'_0 and φ'_r which is x-touching and y-touching.

Note that $\mathsf{Touch}(x, y) = \emptyset$ if there is no embedding sequence between φ'_0 and φ'_r. Then, we have the following lemma.

Lemma 4.25 *For each close pair $(x, y) \in C(T)$, $\mathsf{Touch}(x, y)$ can be computed in polynomial time.*

To prove the lemma, we give necessary and sufficient conditions for the statements $(\mathsf{u}, \mathsf{u}) \in \mathsf{Touch}(x, y)$, $(\mathsf{u}, \mathsf{t}) \in \mathsf{Touch}(x, y)$, and $(\mathsf{t}, \mathsf{t}) \in \mathsf{Touch}(x, y)$, respectively; the condition for $(\mathsf{t}, \mathsf{u}) \in \mathsf{Touch}(x, y)$ is symmetric to the condition for $(\mathsf{u}, \mathsf{t}) \in \mathsf{Touch}(x, y)$.

Before giving the conditions, we introduce another probelm, called GENERALIZED SHORTEST PATH RECONFIGURATION (GSPR, for short) [5], defined as follows. In the problem, we are given a graph G, a terminal set S of size exactly two, a minimum Steiner tree T (actually, a shortest path), a T-embedding φ_0, and a vertex subset

$V_r \subseteq V(G)$. Then, we are asked to determine whether or not there exists a T-embedding φ_r such that $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi_r$ and $V_r \subseteq \varphi(V(T))$. It is known that GSTR is solvable in polynomial time if G is planar and $|V_r| = 1$ holds [5, Theorem 31]. We denote by a 5-tuple $(G', S, T, \varphi_0, V_r)$ an instance of GSPR.

Then we now give necessary and sufficient conditions for the statements $(\mathsf{u}, \mathsf{u}) \in \mathsf{Touch}(x, y)$, $(\mathsf{u}, \mathsf{t}) \in \mathsf{Touch}(x, y)$, and $(\mathsf{t}, \mathsf{t}) \in \mathsf{Touch}(x, y)$. To simplify notation, for a vertex subset $V' \subseteq V(G)$, we denote by $G - V'$ a subgraph of G induced by $V(G) \setminus V'$, that is, $G - V' = G[V(G) \setminus V']$.

Lemma 4.26 *Let* $\mathrm{SPR}(x, y) = (G, S, T, \varphi_s, \varphi_t)$. *Then,* $(\mathsf{u}, \mathsf{u}) \in \mathsf{Touch}(x, y)$ *if and only if the following three conditions hold*;

(a) $\varphi_0(x) = \varphi_r(x)$;

(b) $\varphi_0(y) = \varphi_r(y)$; *and*

(c) $(G', S, T, \varphi_0, \varphi_r)$ *of SPR is a* yes*-instance,*

where $G' = G - (L_T(x) \setminus \{\varphi_0(x)\}) - (L_T(y) \setminus \{\varphi_0(y)\})$.

Proof. We first prove the only-if direction. Suppose that $(\mathsf{u}, \mathsf{u}) \in \mathsf{Touch}(x, y)$ holds, that is, there exists a embedding sequence $\mathcal{R}$ between φ_0 and φ_r which is x-untouching and y-untouching. Then the conditions (a) and (b) clearly hold. Since $\mathcal{R}$ is x-untouching and y-untouching, we know that all T-embeddings in $\mathcal{R}$ do not contain a vertex in $L_T(x) \setminus \{\varphi_0(x)\}$ and in $L_T(y) \setminus \{\varphi_0(y)\}$. Thus the condition (c) hold.

We then prove the if direction. Suppose that all conditions in the lemma hold. Then by conditions (a) and (b), we have $\varphi_0(V(T)) \subseteq V(G')$ and $\varphi_r(V(T)) \subseteq V(G')$, and hence φ_0 and φ_r are suitable T-embedding into G'. Since condition (c) holds, there exists a embedding sequence $\mathcal{R}$ between φ_0 and φ_r on G'. By the definition of G', we observe that layers of x, y in G' satisfy $|L_T(x)| = 1$ and $|L_T(y)| = 1$, and hence $\mathcal{R}$ is x-untouching and y-untouching. Since G' is an induced subgraph of G, $\mathcal{R}$ is a embedding sequence also on G. Thus $\mathcal{R}$ is a desired embedding sequence

between φ_0 and φ_r on G which is x-untouching and y-untouching. □

Lemma 4.27 *Let* $\mathrm{SPR}(x, y) = (G, S, T, \varphi_s, \varphi_t)$. *Then,* $(\mathsf{u}, \mathsf{t}) \in \mathsf{Touch}(x, y)$ *if and only if the following four conditions hold*;

(a) $\varphi_0(x) = \varphi_r(x)$;

(b) $|L_T(y)| = 2$;

(c) $(G', S, T, \varphi_0, \varphi_r)$ *of SPR is a* yes*-instance*; *and*

(d) $(G', S, T, \varphi_0, L_T(y) \setminus \{\varphi_0(y)\})$ *of GSPR is a* yes*-instance,*

where $G' = G - (L_T(x) \setminus \{\varphi_0(x)\})$.

Proof. We first prove the only-if direction. Suppose that $(\mathsf{u}, \mathsf{t}) \in \mathsf{Touch}(x, y)$ holds, that is, there exists a embedding sequence $\mathcal{R}$ between φ_0 and φ_r which is x-untouching and y-touching. Then the conditions (a) and (b) clearly hold. Since $\mathcal{R}$ is x-untouching, we know that all T-embeddings in $\mathcal{R}$ do not contain a vertex in $L_T(x) \setminus \{\varphi_0(x)\}$. Thus the condition (c) hold. Furthermore $\mathcal{R}$ is y-touching, there exists a T-embedding in $\mathcal{R}$ containing the unique vertex in $L_T(y) \setminus \{\varphi_0(y)\}$. Thus the condition (d) hold.

We then prove the if direction. Suppose that all conditions in the lemma hold. Then by the condition (a), we have $\varphi_0(V(T)) \subseteq V(G')$ and $\varphi_r(V(T)) \subseteq V(G')$, and hence φ_0 and φ_r are suitable T-embedding into G'. By the condition (c), we know $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi_r$ on G'. Since the condition (b) hold, we have $|L_T(y) \setminus \{\varphi_0(y)\}| = 1$. Then by the condition (d), we know that there exists a T-embedding φ'_r such that $\varphi_0(y) \notin \varphi'_r(V(T))$ and $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi'_r$. Therefore we can construct a embedding sequence $\mathcal{R} = \langle \varphi_0, \ldots, \varphi'_r, \ldots, \varphi_0, \ldots, \varphi_r \rangle$ since $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi'_r$ and $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi_r$ hold; then $\mathcal{R}$ is y-touching. Furthermore by the definition of G', we observe that layers of x in G' satisfy $|L_T(x)| = 1$, and hence $\mathcal{R}$ is x-untouching. Since G' is induced subgraph of G, $\mathcal{R}$ is a embedding sequence also on G. Thus $\mathcal{R}$ is a desired embedding sequence between φ_0 and φ_r on G which is x-untouching and y-touching. □

Lemma 4.28 *Let* $\mathrm{SPR}(x,y) = (G,S,T,\varphi_s,\varphi_t)$. *Then,* $(\mathsf{t},\mathsf{t}) \in \mathsf{Touch}(x,y)$ *if and only if the following four conditions hold*;

(a) $|L_T(x)| = |L_T(y)| = 2$;

(b) $(G,S,T,\varphi_0,\varphi_r)$ *of SPR is a* yes*-instance*;

(c) $(G,S,T,\varphi_0,L_T(x)\setminus\{\varphi_0(x)\})$ *of GSPR is a* yes*-instance*; *and*

(d) $(G,S,T,\varphi_0,L_T(y)\setminus\{\varphi_0(y)\})$ *of GSPR is a* yes*-instance,*

where $G' = G - (L_T(x)\setminus\{\varphi_0(x)\})$.

Proof. The only-if direction is trivial. We thus prove the if direction. Suppose that all conditions in the lemma hold. By the condition (b), we know $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi_r$. By the condition (a), we have $|L_T(x)\setminus\{\varphi_0(x)\}| = 1$ and $L_T(y)\setminus\{\varphi_0(y)\}| = 1$. Then since the condition (c) holds, we know that there exist a T-embedding φ'_r such that $\varphi_0(x) \notin \varphi'_r(V(T))$ and $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi'_r$. Similarly, since the condition (d) holds, we know that there exist a T-embedding φ''_r such that $\varphi_0(y) \notin \varphi''_r(V(T))$ and $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi''_r$. Then we can construct a embedding sequence $\mathcal{R} = \langle \varphi_0,\ldots,\varphi'_r,\ldots,\varphi_0,\ldots,\varphi''_r,\ldots,\varphi_0,\ldots,\varphi_r\rangle$ since $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi'_r$, $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi''_r$, and $\varphi_0 \overset{\mathsf{emb}}{\leftrightsquigarrow} \varphi_r$ hold; then $\mathcal{R}$ is x-touching and y-touching. This is a desired embedding sequence. □

Since all conditions in Lemma 4.26, 4.27, and 4.28 can be checked in polynomial-time, we can compute $\mathsf{Touch}(x,y)$ in polynomial time.

We finally solve the given instance $(G,S,T,\varphi_0,\varphi_r)$ of Reach-STR under LVE-N. Assume that $\mathrm{SPR}(x,y)$ are yes-instances for all close pairs $(x,y) \in C(T)$, and hence $\mathsf{Touch}(x,y) \neq \emptyset$; otherwise $(G,S,T,\varphi_0,\varphi_r)$ is a no-instance. Consider an assignment $\alpha : B(T) \cup S \to \{\mathsf{u},\mathsf{t}\}$. Then, we say that α is *synchronizing* if $(\alpha(x),\alpha(y)) \in \mathsf{Touch}(x,y)$ holds for every close pair $(x,y) \in C(T)$.

Lemma 4.29 *We can determine in polynomial time whether or not there exists a synchronizing assignment* α.

Proof. We construct a 2CNF formula F which is satisfiable if and only if there exists a synchronizing assignment α; note that satisfiability of 2CNF can be checked

in polynomial time [32]. In the remainder of the proof, we replace the letter t, u by the boolean value $\mathsf{true}, \mathsf{false}$, respectively; for example, $(\mathsf{t}, \mathsf{u}) \in \mathsf{Touch}(x, y)$ for a close pair (x, y) is replaced by $(\mathsf{true}, \mathsf{false}) \in \mathsf{Touch}(x, y)$.

Let $k = |B(T) \cup S|$. For the node set $B(T) \cup S = \{x_1, x_2, \ldots, x_k\}$, we define a variable set $\mathcal{X} = \{X_1, X_2, \ldots, X_k\}$ in which each variable X_i corresponds to $x_i \in B(T) \cup S$. Let $R = \{\mathsf{true}, \mathsf{false}\}$. For a boolean value $a \in R$ and a variable $X_i \in \mathcal{X}$, let X_i^a be a literal X_i if $a = \mathsf{false}$, and $\neg X_i$ if $a = \mathsf{true}$. For each close pair $(x_i, x_j) \in C(T)$, we define the formula $F_{i,j}$ as follows:

$$F_{i,j} = \bigwedge_{(a,b) \in R^2 \setminus \mathsf{Touch}(x_i, x_j)} (X_i^a \vee X_j^b).$$

Then the formula F is defined as follows:

$$F = \bigwedge_{(x_i, x_j) \in C(T)} F_{i,j}.$$

We then show that F is satisfiable if and only if there exists a synchronizing assignment α. For any close pair $(x_i, x_j) \in C(T)$, let $P_{i,j}$ be a set of all pairs of assignment $(X_i, X_j) \in R^2$ satisfying $F_{i,j}$. Then it suffices to show that $P_{i,j}$ is equal to $\mathsf{Touch}(x_i, x_j)$. We observe that a set of all pairs of assignment satisfying a single clause $(X_i^a \vee X_j^b)$ is $R^2 \setminus \{(a, b)\}$. Therefore, we have

$$P_{i,j} = \bigcap_{(a,b) \in R^2 \setminus \mathsf{Touch}(x_i, x_j)} R^2 \setminus \{(a, b)\} = R^2 \setminus (R^2 \setminus \mathsf{Touch}(x_i, x_j)) = \mathsf{Touch}(x_i, x_j).$$

In this way, we can conclude that the lemma follows. □

The following two lemmas complete the proof of Theorem 4.9, which say that an given instance is a yes-instance if and only if there exists a synchronizing assignment.

Lemma 4.30 *Suppose that $(G, S, T, \varphi_0, \varphi_r)$ is an instance of* Reach-STR *under* LVE-N *such that G is a planar graph. If the instance is a* yes*-instance, there exists a synchronizing assignment α.*

Proof. Suppose that an instance $(G, S, T, \varphi_0, \varphi_r)$ of Reach-STR under LVE-N is a yes-instance, and hence there exists a embedding sequence $\mathcal{R} = \langle \varphi_0 =$

$\varphi_0, \varphi_1, \ldots, \varphi_\ell = \varphi_r\rangle$ between φ_0 and φ_r. Then for each node $x \in B(T) \cup S$, it is determined that $\mathcal{R}$ is x-touching or x-untouching; let $\alpha : B(T) \cup S \to \{\mathsf{u}, \mathsf{t}\}$ be an assignment which means that $\mathcal{R}$ is x-touching or x-untouching for each node $x \in B(T) \cup S$. Consider $\mathrm{SPR}(x, y) = (G', S', T', \varphi'_0, \varphi'_r)$ for any close pair $(x, y) \in C(T)$, and let P be the unique path between x and y in T. Then for each $i \in \{0, 1, \ldots, \ell\}$, consider the function $\varphi'_i : V(P) \cup \{s, t\} \to V(G')$ such that $\varphi'_i(s) = s$, $\varphi'_i(t) = t$, and $\varphi'_i(z) = \varphi_i(z)$ for each $z \in V(P)$. We then obtain (by removing redundant ones if needed) the embedding sequence $\langle \varphi'_0 = \varphi'_0, \varphi'_1, \ldots, \varphi'_\ell = \varphi'_r\rangle$ between φ'_0 and φ'_r. Furthermore, we observe that the embedding sequence is x-untouching if $\alpha(x) = \mathsf{u}$, and x-touching otherwise. Similarly, the embedding sequence is y-untouching if $\alpha(y) = \mathsf{u}$, and y-touching otherwise. Thus we know that $(\alpha(x), \alpha(y)) \in \mathsf{Touch}(x, y)$. In this way, we can conclude that α is synchronizing assignment. □

Lemma 4.31 *Suppose that $(G, S, T, \varphi_0, \varphi_r)$ is an instance of* Reach-STR *under* LVE-N *such that G is a planar graph. If there exists a synchronizing assignment α, the instance is a* yes*-instance.*

Proof. Suppose that there exists a synchronizing assignment α. For any subset $C' \subseteq C(T)$, let $T(C')$ be a subgraph of T induced by $\bigcup_{(x,y)\in C'} V(P_{xy})$, where P_{xy} is the unique path between x and y in T. Then, we show the following claim by the induction on $|C'|$.

Claim: Let T' be any subtree of T such that there exists a subset $C' \subseteq C(T)$ satisfying $T' = T(C')$. Then, there exists a sequence $\mathcal{R}' = \langle \varphi'_0, \varphi'_1, \ldots, \varphi'_{\ell'}\rangle$ of mapping such that;

(a) for each $i \in \{0, 1, \ldots, \ell'\}$, the mapping $\varphi'_i : V(T') \to V(G)$ is injective homomorphism from T' to G such that $\varphi'_i(x) \in L_T(x)$ holds for any $x \in V(T')$;

(b) for each $x \in V(T')$, both $\varphi'_0(x) = \varphi_0(x)$ and $\varphi'_{\ell'}(x) = \varphi_r(x)$ hold;

(c) for each $i \in \{0, 1, \ldots, \ell' - 1\}$, $|\{x \in V(T') \mid \varphi'_i(x) \neq \varphi'_{i+1}(x)\}| = 1$ holds; and

(d) for each $x \in V(T') \cap (B(T) \cup S)$, there exists φ'_i in $\mathcal{R}'$ such that $\varphi'_0(x) \neq \varphi'_i(x)$ if and only if $\alpha(x) = \mathsf{t}$.

Note that, if $T' = T$ (that is, $C' = C(T)$), $\mathcal{R}'$ is a embedding sequence between φ_0 and φ_r.

Consider the base case where $|C'| = 1$. Let $(x, y) \in C'$ be the unique close pair in C', and let $\mathrm{SPR}(x, y) = (G'', S'', T'', \varphi''_0, \varphi''_r)$. Then there exists a embedding sequence $\langle \varphi''_0 = \varphi''_0, \varphi''_1, \ldots, \varphi''_{\ell''} = \varphi''_r \rangle$ between φ''_0 and φ''_r corresponding to $(\alpha(x), \alpha(y)) \in \mathsf{Touch}(x, y)$. For each $i \in \{0, 1, \ldots, \ell''\}$, we set $\varphi'_i(x) = \varphi''_i(x)$ for any $x \in V(T(C'))$. Then the sequence $\langle \varphi'_0, \varphi'_1, \ldots, \varphi'_{\ell''} \rangle$ is a desired sequence.

We then consider the case where $|C'| > 1$. Let $(x, y) \in C'$ be any close pair such that $T(C' \setminus \{(x, y)\})$ is a tree; such a close pair always exists. Let $T^p = T(C' \setminus \{(x, y)\})$ and $T^q = T(\{(x, y)\})$. By the induction hypothesis, there exists for T^p a sequence $\mathcal{R}^p = \langle \varphi^p_0, \varphi^p_1, \ldots, \varphi^p_{\ell^p} \rangle$ which satisfies the above four conditions. Similarly, by induction hypothesis, there exists for T^q a sequence $\mathcal{R}^q = \langle \varphi^q_0, \varphi^q_1, \ldots, \varphi^q_{\ell^q} \rangle$ which satisfies the above four conditions. We then construct a desired sequence for $T' = T(C')$ by combining $\mathcal{R}^p$ and $\mathcal{R}^q$. By the definition of T^p and T^q, either $V(T^p) \cap V(T^q) = \{x\}$ or $V(T^q) \cap V(T^p) = \{y\}$ holds. We assume without loss of generality that $V(T^p) \cap V(T^q) = \{x\}$ holds; otherwise the proof is symmetric. Let $\varphi[i, j] : V(T') \to V(G)$ be the mapping such that $\varphi[i, j](z) = \varphi^p_i(z)$ for any $z \in V(T^p)$ and $\varphi[i, j](z) = \varphi^q_j(z)$ for any $z \in V(T') \setminus V(T^p) = V(T^q) \setminus \{x\}$.

If $\alpha(x) = \mathsf{u}$, x does not move at all in both $\mathcal{R}^p$ and $\mathcal{R}^q$. We thus have $\varphi^p_i(x) = \varphi^q_j(x)$ for any pair of indices $i \in \{0, \ldots, \ell^p\}$ and $j \in \{0, \ldots, \ell^q\}$. Furthermore, from Lemma 4.16, reconfiguring a vertex in T^p and that in T^q can be done independently, except for reconfiguring x. Therefore, $\langle \varphi[0, 0], \varphi[1, 0], \ldots, \varphi[\ell^p, 0], \varphi[\ell^p, 1], \ldots, \varphi[\ell^p, \ell^q] \rangle$ is a desired sequence satisfying above four conditions.

We thus consider the other case where $\alpha(x) = \mathsf{t}$. Let $P = \{p_1, p_2, \ldots, p_{|P|}\}$ be the set of all indices in $\mathcal{R}^p$ such that $\varphi^p_{p_i}(x) \neq \varphi^p_{p_i+1}(x)$ for any $p_i \in P$, where

$p_1 < p_2 < \ldots < p_{|P|}$. Similarly, let $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$ be the set of all indices in $\mathcal{R}^q$ such that $\varphi^q_{q_i}(x) \neq \varphi^q_{q_i+1}(x)$ for any $q_i \in Q$,where $q_1 < q_2 < \ldots < q_{|Q|}$.

We first consider the easy case where $|P| = |Q|$ holds. Then we consider the following sequence:

$$
\begin{aligned}
\langle\ & \varphi[0,0], \varphi[1,0], \ldots, \varphi[p_1-1,0], \varphi[p_1-1,1], \ldots, \varphi[p_1-1,q_1-1], \\
& \varphi[p_1,q_1], \varphi[p_1+1,q_1], \ldots, \varphi[p_2-1,q_1], \varphi[p_2-1,q_1+1], \ldots, \varphi[p_2-1,q_2-1], \\
& \vdots \\
& \varphi[p_i,q_i], \varphi[p_i+1,q_i], \ldots, \varphi[p_{i+1}-1,q_i], \varphi[p_{i+1}-1,q_i+1], \ldots, \varphi[p_{i+1}-1,q_{i+1}-1], \\
& \vdots \\
& \varphi[p_{|P|},q_{|Q|}], \varphi[p_{|P|}+1,q_{|Q|}], \ldots, \varphi[\ell^p,q_{|Q|}], \varphi[\ell^p,q_{|Q|}+1], \ldots, \varphi[\ell^p,\ell^q]\ \rangle
\end{aligned}
$$

Then we show that the sequence is a desired sequence satisfying above four conditions. By the induction hypothesis for T^p and T^q, we observe that conditions (b), (c), and (d) are satisfied. For any $\varphi[i,j]$ in the sequence, we also observe by the induction hypothesis that $\varphi[i,j](x) \in L_T(x)$ holds for any $x \in V(T')$; this implies that $\varphi[i,j]$ is an injection. Therefore, to show that the condition (a) holds, it suffices to show that $\varphi[i,j]$ is a homomorphism, and hence it suffices to show that $\varphi^p_i(x) = \varphi^q_j(x)$. Since $\varphi^p_0(x) = \varphi^q_0(x)$ and $|L_T(x)| = 2$, we have $\varphi^p_{p_k}(x) = \varphi^q_{q_k}(x)$ for each $k \in \{0, 1, \ldots, |P| = |Q|\}$. Then by the definition of P and Q, for each $k \in \{0, 1, \ldots, |P| - 1\}$, we know that $\varphi^p_i(x) = \varphi^q_j(x)$ for any pair of indices $i \in \{p_k, \ldots, p_{k+1} - 1\}$ and $j \in \{q_k, \ldots, q_{k+1} - 1\}$, where $p_0 = q_0 = 0$ for convenience. By similar reason, we know that $\varphi^p_i(x) = \varphi^q_j(x)$ for any pair of indices $i \geq p_{|P|}$ and $j \geq q_{|Q|}$. In this way, we obtain the required statement that $\varphi^p_i(x) = \varphi^q_j(x)$ for any $\varphi[i,j]$ in the sequence.

We finally consider the case of $|P| \neq |Q|$. We assume without loss of generality that $|P| > |Q|$; otherwise the proof is symmetric. Since $|L_T(x)| = 2$, $\varphi^p_0(x) = \varphi^q_0(x)$, and $\varphi^p_{\ell^p}(x) = \varphi^q_{\ell^q}(x)$, we know $|P| - |Q|$ is even. In the sequence $\mathcal{R}^q$, if we repeat

$\langle \varphi^p_{p_{|Q|}-1}, \varphi^q_{q_{|Q|}} \rangle$, that is, considering $\langle \varphi^q_0, \varphi^q_1, \ldots, \varphi^q_{q_{|Q|}-1}, \varphi^q_{q_{|Q|}}, \varphi^q_{q_{|Q|}-1}, \varphi^q_{q_{|Q|}}, \ldots, \varphi^q_{\ell^q} \rangle$, then we can increase $|Q|$ by 2, because $\varphi^q_{q_{|Q|}-1}(x) \neq \varphi^q_{q_{|Q|}}(x)$. By repeating this operation, we can always obtained the new sequence for T^q such that $|P| = |Q|$, and then, this case is included by the previous case. □

4.6 Edge exchanges

In this section, we study the complexity of Reach-STR under EE. In Subsection 4.6.1, we give a sufficient condition and a necessary condition for the existence of a reconfiguration sequence under EE between two Steiner trees, which can be checked in polynomial-time. In Section 4.6.2, the remaining instances (i.e. instances which satisfy nether the sufficient condition nor the necessary condition) can be converted to instances of the auxiliary problem. This conversion give us many complexity results for Reach-STR under EE with respect to graph classes.

In this section, we sometimes $S \neq \emptyset$. Otherwise, the instance is also an instance of the edge version under TJ with the property "a graph is a tree." We have shown that it can be linear-time solvable in Theorem 3.6.

4.6.1 Sufficient condition and necessary condition

We first give a sufficient condition, as follows.

Theorem 4.10 *Let (G, S, T_0, T_r) be an instance of* Reach-STR *under* EE*. If $V(T_0) = V(T_r)$, then it is a* yes*-instance.*

Proof. Suppose that $V(T_0) = V(T_r)$ holds. Then, we have $G[V(T_0)] = G[V(T_r)]$. Therefore, both T_0 and T_r form spanning trees of $G[V(T_0)] = G[V(T_r)]$. It is known that any two spanning trees are reconfigurable each other by exchanging a single edge at a time [18, Proposition 1], and hence the theorem follows. □

Theorem 4.10 says that any two Steiner trees are reconfigurable each other as long

as they consist of the same node set. On the other hand, since we can exchange only a single edge at a time, two adjacent Steiner trees having different node sets satisfy a special property. In the following, we focus on a special node, called "free leaf." For convenience, although any Steiner tree T for S is not a rooted tree generally, we call each degree one node of T a *leaf* of T. We say that a leaf v_f of T is *free* if it is a non-terminal, that is, $v_f \in V(T) \setminus S$. Thus, $T[V(T) \setminus \{v_f\}]$ is also a Steiner tree for S, and hence a Steiner tree having a free leaf is not minimal. Then we have the following proposition.

Proposition 2 *Let T and T' be two Steiner trees for a terminal set S in a graph G. Suppose that $T \overset{\mathsf{EE}}{\leftrightarrow} T'$ and $V(T) \neq V(T')$. Then,*

(a) *$V(T) \setminus V(T')$ consists of exactly one node v_f, and v_f is a free leaf in T; and*

(b) *$V(T') \setminus V(T)$ consists of exactly one node v'_f, and v'_f is a free leaf in T'.*

Proof. We prove only the claim (a), because the proof is symmetric for the claim (b).

Suppose for a contradiction that $V(T) \setminus V(T')$ consists of more than one node. Since $S \neq \emptyset$ and both T and T' are Steiner trees for S, we have $S \subseteq V(T) \cap V(T') \neq \emptyset$. Then, since T is connected, T has at least two edges incident to vertices in $V(T) \setminus V(T')$ and hence $|E(T) \setminus E(T')| \geq 2$. This contradicts the assumption that $T \leftrightarrow T'$.

In this way, we have verified that $V(T) \setminus V(T')$ contains exactly one vertex v_f. Since T has just one edge incident to v_f, it is a leaf in T. Since both T and T' are Steiner trees for S, we have $V(T) \setminus V(T') \subseteq V(G) \setminus S$. Thus, v_f is free. □

We now give a sufficient condition for no-instance; by taking a contrapositive, this yields a necessary condition for yes-instance.

Theorem 4.11 *Let (G, S, T_0, T_r) be an instance of* Reach-STR *under* EE*. Then, it is a* no*-instance if the following two conditions* (a) *and* (b) *hold*:

(a) $V(T_0) \neq V(T_r)$; *and*

(b) *at least one of* $G[V(T_0)]$ *and* $G[V(T_r)]$ *has no Steiner tree for* S *with a free leaf.*

Proof. Suppose for a contradiction that (G, S, T_0, T_r) is a yes-instance even though it satisfies both Conditions (a) and (b). Then, there exists a reconfiguration sequence $\mathcal{T}$ between T_0 and T_r. Let T_{i+1} be the first Steiner tree in $\mathcal{T}$ such that $V(T_{i+1}) \neq V(T_0)$; such a Steiner tree exists since $V(T_0) \neq V(T_r)$. Then, the Steiner tree T_i in $\mathcal{T}$ satisfies $T_i \overset{\mathsf{EE}}{\leftrightarrow} T_{i+1}$ and $V(T_i) = V(T_0)$. By Proposition 2, $V(T_i) \setminus V(T_{i+1})$ contains exactly one node v_f which is a free leaf in T_i. Since $V(T_i) = V(T_0)$, we can conclude that $G[V(T_0)]$ has a Steiner tree T_i for S with a free leaf v_f. By the symmetric arguments, $G[V(T_r)]$ has a Steiner tree for S with a free leaf, too. This contradicts the assumption that Condition (b) holds. □

We notice that both Theorems 4.10 and 4.11 can be checked in polynomial time. Furthermore, we know that for any minimum Steiner tree T, $G[T]$ has no (minimum) Steiner tree having a free leaf; otherwise, it contradicts the fact that T is minimum. Therefore, by combining Theorems 4.10 and 4.11, we have the following necessary and sufficient condition for the reachability between minimum Steiner trees.

Theorem 4.12 Reach-STR *under* EE *is linear-time solvable if given two Steiner trees are minimum.*

4.6.2 Reduction from/to auxiliary problem

We now show that instances which satisfy neither Theorem 4.10 nor Theorem 4.11 can be converted to instances of the auxiliary problem.

Let (G, S, T_0, T_r) be such an instance of Reach-STR under EE. Since it does not satisfy Theorem 4.10, we have $V(T_0) \neq V(T_r)$. Furthermore, it also does not satisfy Theorem 4.11, $G[V(T_0)]$ and $G[V(T_r)]$ have Steiner trees T_0' and T_r' with free

leaves, respectively. We know by Theorem 4.10 that T'_0 is reachable from T_0 and T'_r is reachable from T_r. We thus assume without loss of generality that both T_0 and T_r have a free leaf v_0 and v_r, respectively.

This is the main claim in this subsection.

Lemma 4.32 *Let (G, S, T_0, T_r) be an instance of* Reach-STR *under* EE *such that T_0 and T_r have free leaves v_0 and v_r, respectively. Then, $T_0 \overset{\mathsf{EE}}{\leftrightsquigarrow} T_r$ if and only if there exists a Steiner set sequence between two Steiner sets $V(T_0) \setminus \{v_0\}$ and $V(T_r) \setminus \{v_r\}$.*

Proof. We first prove the if direction. (See Figure 4.13.) Suppose that there exists a Steiner set sequence $\mathcal{F} = \langle F_0, F_1, \ldots, F_\ell \rangle$ between $V(T_0) \setminus \{v_0\}$ and $V(T_r) \setminus \{v_r\}$, where $F_0 = V(T_0) \setminus \{v_0\}$ and $F_\ell = V(T_r) \setminus \{v_r\}$. For each $i \in \{1, 2, \ldots, \ell - 1\}$, consider any spanning tree T'_i of $G[F_i]$; let $T'_0 = T_0 \setminus \{v_0\}$ and $T'_\ell = T_r \setminus \{v_r\}$. Then, T'_i is a Steiner tree whose vertex set is of size $|V(T_0)| - 1 = |V(T_r)| - 1$. For each $i \in \{0, 1, \ldots, \ell - 1\}$, let $F_{i+1} \setminus F_i = \{v_i^+\}$, and let T_i^+ be a Steiner tree obtained by adding v_i^+ to T'_i as a free leaf. Similarly, for each $i \in \{1, 2, \ldots . \ell\}$, let $F_{i-1} \setminus F_i = \{v_i^-\}$, and let T_i^- be a Steiner tree obtained by adding v_i^- to T'_i as a free leaf. Let $T_0^- = T_0$ and $T_\ell^+ = T_r$. Then, we have $T_i^- \overset{\mathsf{EE}}{\leftrightarrow} T_i^+$ (or $T_i^- = T_i^+$) for each $i \in \{0, 1, \ldots, \ell\}$, because T_i^+ can be obtained by exchanging the edge incident to the free leaf v_i^- of T_i^- with the edge incident to the free leaf v_i^+ of T_i^+. In addition, $V(T_i^+) = V(T'_i) \cup \{v_i^+\} = V(T'_{i+1}) \cup \{v_{i+1}^-\} = V(T_{i+1}^-)$ for each $i \in \{0, 1, \ldots, \ell - 1\}$, by Theorem 4.10 we have $T_i^+ \overset{\mathsf{EE}}{\leftrightsquigarrow} T_{i+1}^-$. In this way, we can conclude that $T_0 \overset{\mathsf{EE}}{\leftrightsquigarrow} T_r$ holds.

We then prove the only-if direction. Suppose that $T_0 \leftrightsquigarrow T_r$, and hence there exists a reconfiguration sequence $\mathcal{T}$ between T_0 and T_r. We subdivide $\mathcal{T}$ into maximal subsequences $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ so that all Steiner trees in each $\mathcal{T}_j$, $j \in \{1, 2, \ldots, k\}$, have the same vertex set; let $T_{j,0}$ and $T_{j,r}$ be the first and last Steiner trees in each subsequence $\mathcal{T}_j$, respectively. Then, for each $j \in \{1, 2, \ldots, k - 1\}$, we have $V(T_{j,r}) \neq V(T_{j+1,0})$. Since $T_{j,r} \overset{\mathsf{EE}}{\leftrightarrow} T_{j+1,0}$, by Proposition 2 the Steiner trees $T_{j,r}$ and

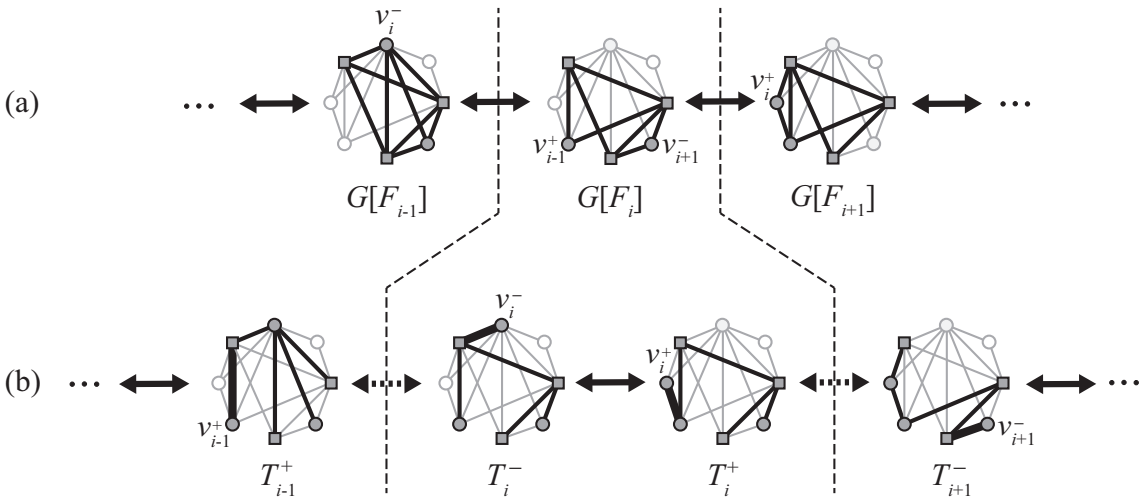


Figure 4.13: (a) A sequence of induced subgraphs $G[F_i]$ obtained by a Steiner set sequence $\mathcal{F} = \langle F_0, F_1, \ldots, F_\ell \rangle$, and (b) its corresponding reconfiguration sequence.

$T_{j+1,0}$ have free leaves $v_{j,r}$ and $v_{j+1,0}$, respectively; we let $v_{1,0} = v_0$ and $v_{k,r} = v_r$ for convenience. Let $F_{j,0} = V(T_{j,0}) \setminus \{v_{j,0}\}$ for each $j \in \{1, 2, \ldots, k\}$. Then, it forms a Steiner set for S, because $T_{j,0} \setminus \{v_{j,0}\}$ is a Steiner tree for S. Similarly, for each $j \in \{1, 2, \ldots, k\}$, $F_{j,r} = V(T_{j,r}) \setminus \{v_{j,r}\}$ is a Steiner set for S. Since $T_{j,r} \overset{\mathsf{EE}}{\leftrightarrow} T_{j+1,0}$ and they differ only in free leaves $v_{j,r}$ and $v_{j+1,0}$, we have $F_{j,r} = V(T_{j,r}) \setminus \{v_{j,r}\} = V(T_{j+1,0}) \setminus \{v_{j+1,0}\} = F_{j+1,0}$ for all $j \in \{1, 2, \ldots, k-1\}$. In addition, since $V(T_{j,0}) = V(T_{j,r})$ for each $j \in \{1, 2, \ldots, k\}$, we have $F_{j,0} \setminus F_{j,r} = \{v_{j,r}\}$ and $F_{j,r} \setminus F_{j,0} = \{v_{j,0}\}$ and hence $F_{j,0} \leftrightarrow F_{j,r}$ holds. In this way, we can obtain a Steiner set sequence $\langle F_{1,0}, F_{2,0}, \ldots, F_{k,0}, F_{k,r} \rangle$ between $F_{1,0} = V(T_0) \setminus \{v_0\}$ and $F_{k,r} = V(T_r) \setminus \{v_r\}$, and hence $V(T_0) \setminus \{v_0\} \leftrightsquigarrow V(T_r) \setminus \{v_r\}$ holds. □

We observe that Reach-STR under EE is in PSACE. When we are given an instance of Reach-STR, we can check in polynomial time if it satisfies Theorems 4.10 and 4.11; in particular, we can do it in linear time on interval graphs and cographs. Therefore, combining Lemma 4.32 and theorems in Section 4.2, we have the following corollary.

Corollary 4.4 Reach-STR *under* EE *is PSPACE-complete for split graphs and planar graphs. On the other hand,* Reach-STR *under* EE *is linear-time solvable*

for cographs and interval graphs, and polynomial-time solvable for cactus graphs.

Chapter 5 Optimization variant

In this chapter, we study the complexity of the optimization variant of INDEPENDENT SET RECONFIGURATION (OPT-ISR, for short).

5.1 Definition of problem and preliminaries

We formally define OPT-ISR. Recall that for a graph G, an *independent set* is a vertex subset $I \subseteq V(G)$ in which no two vertices are adjacent. Then, INDEPENDENT SET RECONFIGURATION under TAR [18, 22] is the reconfiguration problem whose solution space is defined as follows: For a graph G and an integer l, feasible solutions are defined as all independent sets of G of size at least l, and two feasible solutions I and I' are adjacent if $|I \triangle I'| = |(I \setminus I') \cup (I' \setminus I)| = 1$ holds. This adjacency relation is well studied as TAR [18, 19, 22]. Note that if no restriction on the size of feasible solutions (i.e. $l = 0$), there is a reconfiguration sequence between any two independent sets, because we can remove all vertices from one independent set one by one, and add all vertices in the other one. Thus, considering the adjacency relation TAR, we often restrict feasible solutions by setting the lower bound l on the size of any independent set. To emphasize the lower bound l, we sometimes write $\mathsf{TAR}(l)$ instead of TAR. We write $I \overset{l}{\leftrightsquigarrow} I'$ if I and I' are reachable in the solution space with $\mathsf{TAR}(l)$.

Our problem aims to optimize a given independent set under TAR Specifically, OPT-ISR is defined as follows: We are given a graph G, an integer $l \geq 0$, and an independent set I_0 of G such that $|I_0| \geq l$, then asked to find an independent set I_{sol} of G such that $I_0 \overset{l}{\leftrightsquigarrow} I_{\mathsf{sol}}$ and $|I_{\mathsf{sol}}|$ is maximized.

We denote by a triple (G, l, I_0) an instance of Opt-ISR, and call a desired independent set I_{sol} of G an *output solution* to (G, l, I_0). Note that a given independent set I_0 may itself be an output solution. Opt-ISR simply outputs a solution to (G, l, I_0), and does not require the specification of an actual reconfiguration sequence from I_0 to the output solution.

We close this section with noting the following observation which says that Opt-ISR for an instance $(G, 0, I_0)$ is equivalent to finding a maximum independent set of G.

Lemma 5.1 *Every maximum independent set I_{max} of a graph G is a solution to an instance $(G, 0, I_0)$ of* Opt-ISR*, where I_0 is any independent set of G.*

5.2 Polynomial-time solvability

In this section, we study the polynomial-time solvability of Opt-ISR.

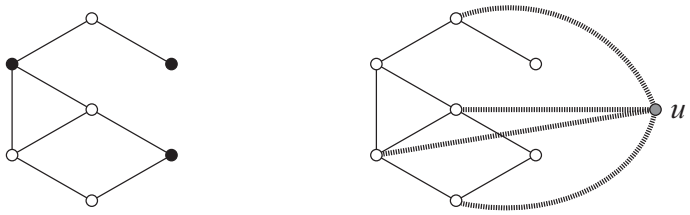
5.2.1 Hardness results

Lemma 5.1 implies that results for the maximum independent set problem can be applied to Opt-ISR for $l = 0$. For example, we have the following theorem, because maximum independent set remains NP-hard for planar graphs [14].

Theorem 5.1 Opt-ISR *is NP-hard for planar graphs and $l = 0$, where l is a lower bound on the size of independent sets.*

It is known that the degeneracy of any planar graph is at most five [23], and hence we have the following corollary.

Corollary 5.1 Opt-ISR *is NP-hard for 5-degenerate graphs and $l = 0$, where l is a lower bound on the size of independent sets.*



(a) G', I'_r (b) G

Figure 5.1: (a) Graph G' and its independent set I'_r for MISR, and (b) the corresponding graph G for Opt-ISR, where newly added edges are depicted by thick dotted lines.

This corollary implies that Opt-ISR admits neither a fixed-parameter algorithm nor an XP algorithm when parameterized by $d + l$ under the assumption that P $\neq$ NP, where d is the degeneracy of an input graph and l is a lower bound on the size of independent sets. We will discuss the fixed parameter (in)tractability of Opt-ISR more deeply in Section 5.3.

We then show that Opt-ISR is PSPACE-hard even if the pathwidth of an input graph is bounded by a constant. The following theorem is the main result of this subsection.

Theorem 5.2 Opt-ISR *is PSPACE-hard for bounded pathwidth graphs.*

Proof. We give a polynomial-time reduction from the reachability variant of maximum independent set reconfiguration problem (MISR for short), defined as follows [35]: We are given a graph G', and two maximum independent sets I'_0 and I'_r of G', and asked to determine whether or not $I'_0 \overset{l'}{\leftrightsquigarrow} I'_r$, where $l' = |I'_0| - 1 = |I'_r| - 1$. We denote by a triple (G', I'_0, I'_r) an instance of MISR. This problem is known to be PSPACE-complete for bounded bandwidth graphs [35]. Since the pathwidth of a graph is at most the bandwidth of the graph [31], MISR is PSPACE-complete also for bounded pathwidth graphs.

Let (G', I'_0, I'_r) be an instance of MISR such that the pathwidth of G' is bounded by a constant. Then, we construct a corresponding instance (G, l, I_0) of Opt-ISR, as

follows. (See also Figure 5.1.) We add a new vertex u to the graph G', and join it with all vertices in $V(G') \setminus I'_r$; let G be the resulting graph, that is, $V(G) = V(G') \cup \{u\}$ and $E(G) = E(G') \cup \{uv : v \in V(G') \setminus I'_r\}$. Since the pathwidth of G' is bounded by a constant and $V(G) = V(G') \cup \{u\}$, the pathwidth of G is also bounded by a constant. Let $l = |I'_0| - 1 = |I'_r| - 1$, and $I_0 = I'_0$. This completes the construction of the corresponding instance (G, l, I_0) of Opt-ISR. This construction can be done in polynomial time.

We now prove the correctness of our reduction. We first claim that G has only one maximum independent set, and it is $I'_r \cup \{u\}$ of size $|I'_r| + 1$. To see this, recall that I'_r is a maximum independent set of G'. Therefore, if G has an independent set I such that $|I| > |I'_r|$, it must contain u. Since u is adjacent to all vertices in $V(G') \setminus I'_r$, only $I'_r \cup \{u\}$ can be such an independent set of size $|I'_r| + 1$, as claimed. Therefore, to complete the correctness proof of our reduction, we prove that Opt-ISR for (G, l, I_0) outputs $I'_r \cup \{u\}$ if and only if $I'_0 \overset{l}{\leftrightsquigarrow} I'_r$ on G'. Since $I'_r \cup \{u\}$ is the unique maximum independent set in G, we indeed prove that $I_0 \overset{l}{\leftrightsquigarrow} I'_r \cup \{u\}$ on G if and only if $I'_0 \overset{l}{\leftrightsquigarrow} I'_r$ on G'.

We first prove the if direction. Suppose that $I'_0 \overset{l}{\leftrightsquigarrow} I'_r$ on G'. Since G contains G' as an induced subgraph, we have $I_0 = I'_0 \overset{l}{\leftrightsquigarrow} I'_r$ on G. Then, $I'_r \cup \{u\}$ can be obtained simply by adding u to I'_r, and hence we can conclude that $I_0 \overset{l}{\leftrightsquigarrow} I'_r \cup \{u\}$ on G.

We next prove the only-if direction. Suppose that $I_0 \overset{l}{\leftrightsquigarrow} I'_r \cup \{u\}$ on G, that is, there exists a reconfiguration sequence $\mathcal{I} = \langle I_0, I_1, \ldots, I_\ell = I'_r \cup \{u\} \rangle$ on G under the TARl rule. Let I_{q+1} be the first independent set in $\mathcal{I}$ which contains u; notice that $I_q = I_{q+1} \setminus \{u\}$ because we know $u \notin I_0$. Since I_{q+1} is an independent set of G, no vertex in $I_q = I_{q+1} \setminus \{u\}$ is adjacent to u. By the construction of u, we thus have $I_q \subseteq I'_r$. Since I_q appears in $\mathcal{I}$, we know $|I_q| \geq l$. Therefore, we can construct a reconfiguration sequence $\mathcal{I}_{qr}$ between I_q and I'_r under the TARl rule by simply

adding the vertices in $I'_r \setminus I_q$ one by one. Then, by combining $\langle I_0, I_1, \ldots, I_{q-1} \rangle$ and $\mathcal{I}_{qr}$ serially, we can obtain a reconfiguration sequence between I_0 and I'_r under the TARl rule such that all independent sets in the sequence do not contain u. We thus have $I'_0 \overset{l}{\leftrightsquigarrow} I'_r$ on G'. □

5.2.2 Linear-time algorithm for chordal graphs

A graph G is *chordal* if every induced cycle in G is of length three [8]. The main result of this subsection is the following theorem.

Theorem 5.3 Opt-ISR *is solvable in linear time for chordal graphs.*

This theorem can be obtained from the following lemma; we note that a maximum independent set I_{max} of a chordal graph can be found in linear time [13], and the maximality of a given independent set can be checked in linear time.

Lemma 5.2 *Let (G, l, I_0) be an instance of* Opt-ISR *such that G is a chordal graph, and let I_{max} be any maximum independent set of G. Then, a solution I_{sol} to (G, l, I_0) can be obtained as follows*:

$$I_{\mathsf{sol}} = \begin{cases} I_0 & \text{if } I_0 \text{ is a maximal independent set of } G \text{ and } |I_0| = l; \\ I_{\mathsf{max}} & \text{otherwise.} \end{cases}$$

Proof. We first consider the case where I_0 is a maximal independent set of G and $|I_0| = l$. In this case, we cannot remove any vertex from I_0 because $|I_0| = l$. Furthermore, since I_0 is maximal, we cannot add any vertex in $V(G) \setminus I_0$ to I_0 while maintaining independence. Therefore, G has no independent set I' $(\neq I_0)$ which is reachable from I_0, and hence $I_{\mathsf{sol}} = I_0$.

We then consider the other case, that is, I_0 is not a maximal independent set of G or $|I_0| > l$. Observe that it suffices to consider the only case where $|I_0| > l$ holds; if $|I_0| = l$ and I_0 is not maximal, then we can obtain an independent set I''_0 of G such that $|I''_0| = l + 1$ and $I_0 \overset{l}{\leftrightsquigarrow} I''_0$ by adding some vertex in $V(G) \setminus I_0$. To prove $I_{\mathsf{sol}} = I_{\mathsf{max}}$, we below show that $I_0 \overset{l}{\leftrightsquigarrow} I_{\mathsf{max}}$ holds if $|I_0| > l$.

Let $I'_0 \subseteq I_0$ be any independent set of size $l+1$. Then, $I_0 \overset{l}{\leftrightsquigarrow} I'_0$ holds, because we can obtain I'_0 from I_0 by removing vertices in $I_0 \setminus I'_0$ one by one. Similarly, let $I' \subseteq I_{\mathsf{max}}$ be any independent set of size $l+1$; we know that $I' \overset{l}{\leftrightsquigarrow} I_{\mathsf{max}}$ holds. Kamiński et al. [22] proved that any two independent sets of the same size $l+1$ are reachable under the $\mathsf{TAR}(l)$ rule for even-hole-free graphs. Since any chordal graph is even-hole free, we thus have $I'_0 \overset{l}{\leftrightsquigarrow} I'$. Therefore, we have $I_0 \overset{l}{\leftrightsquigarrow} I'_0 \overset{l}{\leftrightsquigarrow} I' \overset{l}{\leftrightsquigarrow} I_{\mathsf{max}}$, and hence we can conclude that $I_0 \overset{l}{\leftrightsquigarrow} I_{\mathsf{max}}$ holds as claimed. □

We note that Lemma 5.2 indeed holds for even-hole-free graphs, which contain all chordal graphs. However, the complexity status of the MAXIMUM INDEPENDENT SET problem is unknown for even-hole-free graphs, and hence we do not know if we can obtain I_{max} in polynomial time.

5.3 Fixed-parameter tractability

In this section, we study the fixed parameter (in)tractability of Opt-ISR. We sometimes take the size s of output solution as the parameter; we call s a *solution size*. More formally, for an instance (G, l, I_0), the problem Opt-ISR *parameterized by solution size* s asks whether G has an independent set I such that $|I| \geq s$ and $I_0 \overset{l}{\leftrightsquigarrow} I$. We may assume that $s > l$; otherwise we are dealing with a yes-instance because I_0 itself is a solution. We sometimes denote by a 4-tuple (G, l, I_0, s) an instance of Opt-ISR parameterized by solution size s.

5.3.1 Single parameter: solution size

We first give an observation that can be obtained from INDEPENDENT SET. Because INDEPENDENT SET is W[1]-hard when parameterized by solution size s [28], Lemma 5.1 implies the following theorem.

Theorem 5.4 Opt-ISR *is* $W[1]$*-hard when parameterized by solution size* s*.*

This theorem implies that Opt-ISR admits no fixed-parameter algorithm with respect to solution size s under the assumption that FPT $\neq$ W[1]. However, it admits an XP algorithm with respect to s, as in the following theorem.

Theorem 5.5 Opt-ISR *parameterized by solution size s can be solved in time $O(s^3n^{2s})$, where n is the number of vertices in a given graph.*

Proof. For an instance (G, l, I_0, s), we construct an *auxiliary graph* G_A, defined as follows: Each node in G_A corresponds to an independent set I of G such that $l \leq |I| \leq s$, and there is an edge in G_A between two nodes corresponding to independent sets I and I' if and only if $|\triangle II'| = 1$ holds. Notice that G_A has a node corresponding to I_0, since $l \leq |I_0| \leq s$. Then, by breadth-first search starting from the node corresponding to I_0, we can check if there is an independent set I of G such that $|I| = s$ and $I_0 \overset{l}{\leftrightsquigarrow} I$.

We now estimate the running time of the algorithm. Let n and m denote the numbers of vertices and edges in G, respectively. The number of (candidates of) nodes in G_A can be bounded by $\sum_{l \leq j \leq s} \binom{n}{j} = O(sn^s)$. For each enumerated vertex subset of G, we check if it forms an independent set of G; this can be done in time $O(n+m)$. Therefore, the node set $V(G_A)$ can be constructed in time $O(sn^s(n+m))$. We then check each pair of nodes in $V(G_A)$; there are $O(|V(G_A)|^2) = O(s^2n^{2s})$ pairs. We join the pair by an edge in G_A if their corresponding independent sets differ in only one vertex; we can check this condition in time $O(s)$ for each pair of nodes. In this way, we can construct the auxiliary graph G_A in time $O(s^3n^{2s})$ in total. Since breadth-first search can be executed in time $O(|V(G_A)| + |E(G_A)|) = O(s^2n^{2s})$, our algorithm runs in time $O(s^3n^{2s})$ in total. □

5.3.2 Two parameters: solution size and degeneracy

As we have shown in Theorem 5.4, Opt-ISR admits no fixed-parameter algorithm when parameterized by the single parameter of solution size s under the assumption

that FPT $\neq$ W[1]. In addition, Theorem 5.2 implies that the problem remains PSPACE-hard even if the degeneracy d of an input graph is bounded by a constant, and hence Opt-ISR does not admit even an XP algorithm with respect to the single parameter d under the assumption that P $\neq$ PSPACE. In this subsection, we take these two parameters, and develop a fixed-parameter algorithm as in the following theorem.

Theorem 5.6 Opt-ISR *admits a fixed-parameter algorithm when parameterized by* $s + d$*, where* s *is the solution size and* d *is the degeneracy of an input graph.*

Before proving the theorem, we note the following corollary which holds for planar graphs, and for bounded treewidth graphs. Recall that Opt-ISR is intractable (from the viewpoint of polynomial-time solvability) for these graphs, as shown in Theorems 5.1 and 5.2.

Corollary 5.2 Opt-ISR *parameterized by solution size* s *is fixed-parameter tractable for planar graphs, and for bounded treewidth graphs.*

Proof. Recall that the degeneracy of any planar graph is at most five. It is known that the degeneracy of a graph is at most the treewidth of the graph. Thus, the corollary follows from Theorem 5.6. □

Outline of algorithm.

As a proof of Theorem 5.6, we give such an algorithm. We first explain our idea and the outline of the algorithm. Our idea is to extend a fixed-parameter algorithm for Reach-ISR when parameterized by $l + d$ [24].

Consider the case where an input graph G consists of only a fixed-parameter number of vertices, that is, $|V(G)|$ can be bounded by some function of $s+d$. Then, we apply Theorem 5.5 to the instance and obtain the answer in fixed-parameter time (Lemma 5.4). We here use the fact (stated by Lokshtanov et al. [24, Proposition 2])

that a d-degenerate graph consists of a small number of vertices if it has a small number of low-degree vertices (Lemma 5.3).

Therefore, it suffices to consider the case where an input graph has many low-degree vertices. In this case, we will kernelize the instance: we will show that there always exists a low-degree vertex which can be removed from an input graph without changing the answer (yes or no) to the instance. Our kernelization has two stages. In the first stage, we focus on "twins" (two vertices that have the same closed neighborhoods), and prove that one of them can be removed without changing the answer (Lemma 5.5). The second stage will be executed only when the first stage cannot kernelize the instance into a sufficiently small size. The second stage is a bit involved, and makes use of the Sunflower Lemma by Erdös and Rado [12].

Graphs having a small number of low-degree vertices.

We now give our algorithm. Suppose that (G, l, I_0, s) is an instance of Opt-ISR parameterized by solution size such that G is a d-degenerate graph. We assume that $|I_0| < s$; otherwise (G, l, I_0, s) is a yes-instance because I_0 itself is a solution.

We start with noting the following property for d-degenerate graphs, which is a little bit stronger claim than that of Lokshtanov et al. [24, Proposition 2]; however, the proof is almost the same as that of [24].

Lemma 5.3 *Suppose that a graph G is d-degenerate, and let $D \subseteq V(G)$ be the set of all vertices of degree at most $2d$ in G. Then, $|V(G)| \leq (2d+1)|D|$.*

Proof. Suppose for a contradiction that $|V(G)| = (2d+1)|D| + c$ holds for some integer $c \geq 1$. Then, $|V(G) \setminus D| = 2d|D| + c$, and hence we have

$$\begin{aligned} |E(G)| &= \frac{1}{2}\sum_{v \in V(G)} |N_G(v)| \geq \frac{1}{2}\sum_{v \in V(G)\setminus D} (2d+1) \\ &= \frac{1}{2}(2d+1)(2d|D|+c) = d|V(G)| + \frac{1}{2}c > d|V(G)|. \end{aligned}$$

This contradicts the fact that $|E(G)| \leq d|V(G)|$ holds for any d-degenerate graph G [23]. □

Let $D = \{v \in V(G) : |N_G(v)| \leq 2d\}$, and let $D' = D \setminus I_0$. We introduce a function $f(s,d)$ which depends on only s and d; more specifically, let $f(s,d) = (2d+1)!((2s+d+1)-1)^{2d+1}$. We now consider the case where G has only a fixed-parameter number of vertices of degree at most $2d$.

Lemma 5.4 *If $|D'| \leq f(s,d)$, then* Opt-ISR *can be solved in fixed-parameter time with respect to s and d.*

Proof. Since $D' = D \setminus I_0$ and $|I_0| < s$, we have $|D| \leq |D'| + |I_0| < f(s,d) + s$. By Lemma 5.3 we thus have $|V(G)| \leq (2d+1)|D| < (2d+1)(f(s,d)+s)$. Therefore, $|V(G)|$ depends only on s and d. Then, this lemma follows from Theorem 5.5. □

First stage of kernelization.

We now consider the remaining case, that is, $|D'| > f(s,d)$ holds. The first stage of our kernelization focuses on "twins," two vertices having the same closed neighborhoods, and removes one of them without changing the answer.

Lemma 5.5 *Suppose that there exist two vertices b_i and b_j in D' such that $N_G[b_i] = N_G[b_j]$. Then, (G, l, I_0, s) is a* yes*-instance if and only if $(G \setminus \{b_i\}, l, I_0, s)$ is.*

Proof. We note that $b_i \notin I_0$ and $b_j \notin I_0$, because $D' = D \setminus I_0$. Then, the if direction clearly holds, and hence we prove the only-if direction. Suppose that (G, l, I_0, s) is a yes-instance, and hence G has an independent set I_{sol} such that $|I_{\mathsf{sol}}| \geq s$ and $I_0 \overset{l}{\leftrightsquigarrow} I_{\mathsf{sol}}$. Then, there exists a reconfiguration sequence $\mathcal{I} = \langle I_0, I_1, \ldots, I_\ell = I_{\mathsf{sol}} \rangle$. Since $N_G[b_i] = N_G[b_j]$, we know that b_i and b_j are adjacent in G and hence no independent set of G contains b_i and b_j at the same time. We now consider a new sequence $\mathcal{I}' = \langle I'_0, I'_1, \ldots, I'_\ell \rangle$ defined as follows: for each $x \in \{0, 1, \ldots, \ell\}$, let

$$I'_x = \begin{cases} I_x & \text{if } b_i \notin I_x; \\ (I_x \setminus \{b_i\}) \cup \{b_j\} & \text{otherwise.} \end{cases}$$

Since each I_x, $x \in \{0, 1, \ldots, \ell\}$, is an independent set of G and $N_G[b_i] = N_G[b_j]$, each I'_x forms an independent set of G. In addition, since $|\triangle I_{x-1}I_x| = 1$ for all $x \in \{1, 2, \ldots, \ell\}$, we have $|\triangle I'_{x-1}I'_x| = 1$. Therefore, $\mathcal{I}'$ is a reconfiguration sequence such that no independent set in $\mathcal{I}'$ contains b_i. Since $|I'_\ell| = |I_\ell| = |I_{\mathsf{sol}}| \geq s$, we can conclude that $(G \setminus \{b_i\}, l, I_0, s)$ is a yes-instance. □

We repeatedly apply Lemma 5.5 to a given graph, and redefine G as the resulting graph; we also redefine D and D' according to the resulting graph G. Then, any two vertices b_i and b_j in D' satisfy $N_G[b_i] \neq N_G[b_j]$. If $|D'| \leq f(s, d)$, then we have completed our kernelization; recall Lemma 5.3. Otherwise, we will execute the second stage of our kernelization described below.

Second stage of kernelization.

In the second stage of the kernelization, we use the classical result of Erdös and Rado [12], known as the *Sunflower Lemma*. We first define some terms used in the lemma. Let $P_1, P_2, \ldots, P_p$ be p non-empty sets over a universe U, and let $C \subseteq U$ which may be an empty set. Then, the family $\{P_1, P_2, \ldots, P_p\}$ is called a *sunflower* with a *core* C if $P_i \setminus C \neq \emptyset$ holds for each $i \in \{1, 2, \ldots, p\}$, and $P_i \cap P_j = C$ holds for each $i, j \in \{1, 2, \ldots, p\}$ satisfying $i \neq j$. The set $P_i \setminus C$ is called a *petal* of the sunflower. Note that a family of pairwise disjoint sets always forms a sunflower (with an empty core). Then, the following lemma holds.

Lemma 5.6 (Erdös and Rado [12]) *Let $\mathcal{A}$ be a family of sets (without duplicates) over a universe U such that each set in $\mathcal{A}$ is of size at most t. If $|\mathcal{A}| > t!(p-1)^t$, then there exists a family $\mathcal{S} \subseteq \mathcal{A}$ which forms a sunflower having p petals. Furthermore, $\mathcal{S}$ can be computed in time polynomial in $|\mathcal{A}|$, $|U|$, and p.*

We now explain the second stage of our kernelization, and make use of Lemma 5.6. Let $b_1, b_2, \ldots, b_{|D'|}$ denote the vertices in D', and let $\mathcal{A} = \{N_G[b_1],$

$N_G[b_2], \ldots, N_G[b_{|D'|}]\}$ be the set of closed neighborhoods of all vertices in D'. In the second stage, recall that $N_G[b_i] \neq N_G[b_j]$ holds for any two vertices b_i and b_j in D', and hence no two sets in $\mathcal{A}$ are identical. We set $U = \bigcup_{b_i \in D'} N_G[b_i]$. Since each $b_i \in D'$ is of degree at most $2d$ in G, each $N_G[b_i] \in \mathcal{A}$ is of size at most $2d+1$. Notice that $|\mathcal{A}| = |D'| > f(s,d) = (2d+1)!((2s+d+1)-1)^{2d+1}$. Therefore, we can apply Lemma 5.6 to the family $\mathcal{A}$ by setting $t = 2d+1$ and $p = 2s+d+1$, and obtain a sunflower $\mathcal{S} \subseteq \mathcal{A}$ with a core C and p petals in time polynomial in $|\mathcal{A}|$, $|U|$, and $p = 2s+d+1$. Notice that $|\mathcal{A}| \leq n$ and $|U| \leq n$, and hence we can obtain $\mathcal{S}$ in time polynomial in n. Let $S = \{b'_1, b'_2, \ldots, b'_p\} \subseteq D'$ be the set of p vertices whose closed neighborhoods correspond to the sunflower $\mathcal{S}$, that is, $\mathcal{S} = \{N_G[b'_1], N_G[b'_2], \ldots, N_G[b'_p]\} \subseteq \mathcal{A}$. The following lemma says that all vertices in S are contained in the p petals of the sunflower $\mathcal{S}$ (i.e., not in the core C), and they form an independent set of G.

Lemma 5.7 *$S \cap C = \emptyset$ holds. Furthermore, vertices in S form an independent set of G.*

Proof. We first prove that $S \cap C = \emptyset$. Assume for a contradiction that $S \cap C \neq \emptyset$ holds. Then we know that vertices in $S \cap C$ form a clique in G, because for any two vertices b'_i and b'_j in $S \cap C$, $b'_i \in C = N_G[b'_i] \cap N_G[b'_j]$ holds and hence b'_i and b'_j are adjacent. Since d-degenerate graph has no clique of size more than $d+1$, such a clique has the size at most $d+1$, that is, $|S \cap C| \leq d+1$. S contains $p = 2s+d+1 > d+1$ vertices, and hence we can find at least one vertex in $S \setminus C$; let b'_i be any vertex in $S \setminus C$ and b'_j be any vertex in $S \cap C$. Then, b'_i and b'_j must be adjacent because $b'_j \in C = N_G[b'_i] \cap N_G[b'_j]$, however, it contradicts the fact that $b'_i \notin C = N_G[b'_i] \cap N_G[b'_j]$.

We then show that S form an independent set of G. It can be obtained from the fact that any two vertices b'_i and b'_j in S are not contained in $C = N_G[b'_i] \cap N_G[b'_j]$, that is, they are not adjacent. □

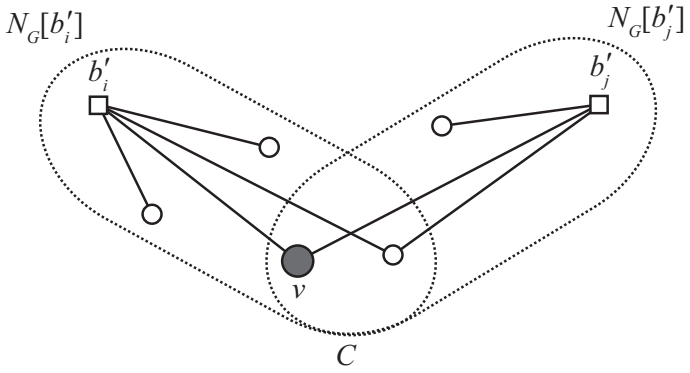


Figure 5.2: Illustration for Lemma 5.8, where two vertices $b'_i, b'_j \in S \setminus C$ are depicted by squares. By the definition of a sunflower, all vertices v adjacent to both b'_i and b'_j must be contained in $C = N_G[b'_i] \cap N_G[b'_j]$.

By using Lemma 5.7, We finally obtain the following lemma, as the second stage of the kernelization.

Lemma 5.8 *Let b'_q be any vertex in S. Then, (G, l, I_0, s) is a* yes*-instance if and only if $(G \setminus \{b'_q\}, l, I_0, s)$ is.*

Proof. Note that $b'_q \notin I_0$ since $S \subseteq D' \subseteq V(G) \setminus I_0$. Then, the if direction clearly holds, and hence we prove the only-if direction. Suppose that (G, l, I_0, s) is a yes-instance, and hence G has an independent set I_{sol} such that $|I_{\mathsf{sol}}| \geq s$ and $I_0 \overset{l}{\leftrightsquigarrow} I_{\mathsf{sol}}$. Then, there exists a reconfiguration sequence $\mathcal{I} = \langle I_0, I_1, \ldots, I_\ell = I_{\mathsf{sol}} \rangle$. If no independent set in $\mathcal{I}$ contains b'_q, then the only-if direction holds. Therefore, we consider the case where at least one independent set in $\mathcal{I}$ contains b'_q. Let I_{r+1} be the first independent set in $\mathcal{I}$ which contains b'_q, that is, $b'_q \notin I_i$ for all $i \in \{0, 1, \ldots, r\}$. We assume without loss of generality that $|I_i| < s$ holds for all $i \in \{0, 1, \ldots, \ell - 1\}$.

Let $S' = S \setminus (\{b'_q\} \cup C \cup N_G[I_r])$, where $N_G[I_r] = \bigcup_{v \in I_r} N_G[v]$. We now claim that $I'_{\mathsf{sol}} = I_r \cup S'$ is an independent set of G such that $|I'_{\mathsf{sol}}| \geq s$ and $I_0 \overset{l}{\leftrightsquigarrow} I'_{\mathsf{sol}}$ on $G \setminus \{b'_q\}$; then $(G \setminus \{b'_q\}, l, I_0, s)$ is a yes-instance. Since $S' \subseteq S$, Lemma 5.7 says that S' is an independent set of G. Furthermore, since S' does not contain any vertex in $N_G[I_r]$, $I'_{\mathsf{sol}} = I_r \cup S'$ is an independent set of G. Recall that $I_0 \overset{l}{\leftrightsquigarrow} I_r$ holds on $G \setminus \{b'_q\}$, and hence we know $|I_r| \geq l$. Then, $I_0 \overset{l}{\leftrightsquigarrow} I_r \overset{l}{\leftrightsquigarrow} I'_{\mathsf{sol}}$ holds on $G \setminus \{b'_q\}$ by adding

the vertices in S' to I_r one by one. We finally prove that $|I'_{\mathsf{sol}}| \geq s$ by showing that $|S'| \geq s$ holds. Since $|S \setminus C| = 2s + d + 1 \geq 2s$ (by Lemma 5.7) and $|I_r| < s$, it suffices to prove that $|N_G[v] \cap (S \setminus C)| \leq 1$ holds for each vertex $v \in I_r$. Since I_{r+1} is obtained by adding b'_q to I_r, we know $I_r \cap N_G[b'_q] = \emptyset$. Since $C \subset N_G[b'_q]$, we thus have $I_r \cap C = \emptyset$. Therefore, each vertex $v \in I_r$ is adjacent to at most one vertex in $S \setminus C$, because otherwise v must be contained in C. (See also Figure 5.2.) □

We can repeatedly apply Lemma 5.8 to G until the resulting graph has the corresponding vertex subset D' such that $|D'| \leq f(s, d)$. Then, by Lemma 5.4 we have completed our kernelization. This completes the proof of Theorem 5.6.

Chapter 6 Conclusions

In this thesis, we mainly studied reconfiguration problems from the following two viewpoints; the first one is to develop efficient algorithms for reconfiguration problems whose feasible solutions are connected subgraphs, and the second one is to introduce a new variant of reconfiguration problems in which we do not need to specify a target solution and are asked for a desirable solution that is reachable from an initial solution.

In Chapter 3 and 4, we focused on several graph properties: path, cycle, tree, clique, biclique, diameter-two, and Steiner tree; all of them require the connectivity. Then, for each property, we studied the computational complexity of the reconfiguration problem whose feasible solutions are subgraphs that satisfy the property. In particular, we deeply analyzed the complexity of the reconfiguration problem of Steiner trees with respect to graph classes.

In Chapter 5, we introduce the new variant of reconfiguration problems which is called the optimization variant. As the first example of this variant, we applied this variant to the reconfiguration problem of independent sets which is one of the most well-studied reconfiguration problems. We then analyzed its polynomial-time solvability with respect to graph classes and parameterized complexity for three parameters, the lower bound of independent sets, the size of an independent set we are asked for, and the degeneracy of an input graph.

Finally, we discuss about future works. We developed in this thesis several polynomial-time algorithms solving the reconfiguration problems with connected subgraphs. Then, one of future works is to solve other reconfiguration problems

with connected subgraphs by using our algorithm techniques. Regarding the optimization variant, since it has been just proposed, it is applied to little reconfiguration problems; to the best of the author's knowledge, it has been just applied to INDEPENDENT SET RECONFIGURATION, DOMINATING SET RECONFIGURATION [3], and VERTEX COLORING RECONFIGURATION [15]. Therefore, a future work for this variant is to introduce and study the optimization variant for other reconfiguration problems.

Bibliography

[1] D. Andrade, M. Resende, and R. Werneck. Fast local search for the maximum independent set problem. *Journal of Heuristics*, 18(4):525–547, 2012.

[2] B. Baker. Approximation algorithms for np-complete problems on planar graphs. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science* (*FOCS 2015*), pages 265–273, 1983.

[3] A. Blanché, H. Mizuta, P. Ouvrard, and A. Suzuki. Decremental optimization of dominating sets under reachability constraints. `arXiv:1906.05163`, 2019.

[4] P. Bonsma. The complexity of rerouting shortest paths. *Theoretical Computer Science*, 510:1–12, 2013.

[5] P. Bonsma. Rerouting shortest paths in planar graphs. *Discrete Applied Mathematics*, 231:95–112, 2017.

[6] P. Bonsma and L. Cereceda. Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. *Theoretical Computer Science*, 410(50):5215–5226, 2009.

[7] G. Borradaile, C. Kenyon-Mathieu, and P. Klein. A polynomial-time approximation scheme for steiner tree in planar graphs. In *Proceedings of the 18th Annual ACM-SIAM symposium on Discrete algorithms* (*SODA 2007*), pages 1285–1294, 2007.

[8] A. Brandstädt, V. Le, and J. Spinrad. *Graph Classes: A Survey*. SIAM, Philadelphia, PA, 1999.

[9] L. Cereceda, J. van den Heuvel, and M. Johnson. Finding paths between 3-colorings. *Journal of Graph Theory*, 67(1):69–82, 2011.

[10] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms (3rd Edition)*. MIT Press, 2009.

[11] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. van Rooji, and J. Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science* (*FOCS 2011*), pages 150–159, 2011.

[12] P. Erdös and R. Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 35:85–90, 1960.

[13] A. Frank. Some polynomial algorithms for certain graphs and hypergraphs. In *Proceedings of the 5th British Combinatorial Conference* (*BCC 1975*), pages 211–226, 1975.

[14] M. Garey and D. Johnson. *Computers and Intractability*: *A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.

[15] T. Hatanaka. Open problem - optimizing a coloring via a reconfiguration sequence. presented in the second international workshop on combinatorial reconfiguration (CoRe 2017), 2017.

[16] R. Hearn and E. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1–2):72–96, 2005.

[17] J. van den Heuvel. The complexity of change. In *Surveys in Combinatorics 2013*, volume 409 of *London Mathematical Society Lecture Note Series*, pages 127–160. Cambridge University Press, 2013.

[18] T. Ito, E. D. Demaine, N. J. A. Harvey, C. H. Papadimitriou, M. Sideri, R. Uehara, and Y. Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12–14):1054–1065, 2011.

[19] T. Ito, H. Ono, and Y. Otachi. Reconfiguration of cliques in a graph. In *Proceedings of the 12th Annual Conference on Theory and Applications of Models of Computation* (*TAMC 2015*), pages 212–223, 2015.

[20] W. Johnson and W. Story. Notes on the "15" puzzle. *American Journal of Mathematics*, 2(4):397–404, 1879.

[21] M. Kamiński, P. Medvedev, and M. Milanič. Shortest paths between shortest paths. *Theoretical Computer Science*, 412(39):5205–5210, 2011.

[22] M. Kamiński, P. Medvedev, and M. Milanič. Complexity of independent set reconfigurability problems. *Theoretical Computer Science*, 439:9–15, 2012.

[23] D. Lick and A. White. k-degenerate graphs. *Canadian Journal of Mathematics*, 22:1082–1096, 1970.

[24] D. Lokshtanov, A. Mouawad, F. Panolan, M. Ramanujan, and S. Saurabh. Reconfiguration on sparse graphs. *Journal of Computer and System Sciences*, 95:122–131, 2018.

[25] A. Mouawad, N. Nishimura, V. Raman, and S. Siebertz. Vertex cover reconfiguration and beyond. *Algorithms*, 11(2):1–21, 2018.

[26] A. Mouawad, N. Nishimura, V. Raman, N. Simjour, and A. Suzuki. On the parameterized complexity of reconfiguration problems. *Algorithmica*, 78(1):274–297, 2017.

[27] M. Mühlenthaler. Degree-constrained subgraph reconfiguration is in P. In *Proceedings of the 40th International Symposium on Mathematical Foundations*

of Computer Science (*MFCS 2015*), volume 9235 of *Lecture Notes in Computer Science*, pages 505–516, 2015.

[28] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford, 2006.

[29] N. Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):paper id 52, 2018.

[30] W. Pullan and H. Hoo. Dynamic local search for the maximum clique problem. *Journal of Artificial Intelligence Research*, 25(1):159–185, 2006.

[31] T. Schiex. A note on CSP graph parameters. *Technical Report 1999/03, French National Institute for Agricultural Research* (*INRA*), 1999.

[32] S. T. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing* (*STOC 1978*), 1978.

[33] A. Takaoka. Complexity of hamiltonian cycle reconfiguration. *Algorithms*, 11(9):1–15, 2018.

[34] K. Wasa, K. Yamanaka, and H. Arimura. The complexity of induced tree reconfiguration problems. *IEICE Transactions on Information and Systems*, 102-D(3):464–469, 2019.

[35] M. Wrochna. Reconfiguration in bounded bandwidth and tree-depth. *Journal of Computer and System Sciences*, 93:1–10, 2018.

List of papers

Refereed papers in journals

1. Tesshu Hanaka, Takehiro Ito, Haruka Mizuta, Benjamin Moore, Naomi Nishimura, Vijay Subramanya, Akira Suzuki and Krishna Vaidyanathan, Reconfiguring Spanning and Induced Subgraphs, *Theoretical Computer Science*, Vol. 806, pp. 553–566, 2020.

2. Haruka Mizuta, Takehiro Ito and Xiao Zhou, Reconfiguration of Steiner trees in an unweighted graph, *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, E100-A(7), pp. 1532-1540, 2017.

Refereed papers in international conferences

1. Haruka Mizuta, Tatsuhiko Hatanaka, Takehiro Ito and Xiao Zhou, Reconfiguration of minimum Steiner trees via vertex exchanges, in *Proceedings of the 44th International Symposium on Mathematical Foundations of Computer Science* (*MFCS 2019*), *Leibniz International Proceedings in Informatics* (*LIPIcs*), Vol. 138, pp. 79:1-79:11, 2019.

2. Takehiro Ito, Haruka Mizuta, Naomi Nishimura and Akira Suzuki, Incremental optimization of independent sets under the reconfiguration framework, in *Proceedings of the 25th International Computing and Combinatorics Conference* (*COCOON 2019*), *Lecture Notes in Computer Science* (*LNCS*), Vol. 11653, pp. 313-324, 2019.

3. Tesshu Hanaka, Takehiro Ito, Haruka Mizuta, Benjamin Moore, Naomi Nishimura, Vijay Subramanya, Akira Suzuki and Krishna Vaidyanathan, Reconfiguring spanning and induced subgraphs, in *Proceedings of the 24th International Computing and Combinatorics Conference* (*COCOON 2018*), *Lecture Notes in Computer Science* (*LNCS*), Vol. 10976, pp. 428-440, 2018.

4. Haruka Mizuta, Takehiro Ito and Xiao Zhou, Reconfiguration of Steiner trees in an unweighted graph, in *Proceedings of the 27th International Workshop on Combinatorial Algorithms* (*IWOCA 2016*), *Lecture Notes in Computer Science* (*LNCS*), Vol. 9843, pp. 163-175, 2016.