On the Complexity of Reconfiguration of Clique, Cluster Vertex Deletion, and Dominating Set

by

Youcef Tebbal

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

Waterloo, Ontario, Canada, 2015

Abstract

A graph problem $\mathcal{P}$ is a vertex-subset problem if feasible solutions for $\mathcal{P}$ consist of subsets of the vertices of a graph G. The st-connectivity problem for a vertex-subset problem $\mathcal{P}$ takes as input two feasible solutions S_s and S_t, and determines if there is a sequence of reconfiguration steps that can be applied to transform S_s into S_t, such that each step results in a feasible solution of $\mathcal{P}$ of size bounded by k and each step is a vertex addition or deletion. For most NP-complete problems, this problem has been shown to be PSPACE-complete, while for some problems in P, this problem could be either in P or PSPACE-complete. However, knowing the complexity of a decision problem does not directly imply the complexity of its st-connectivity problem. Therefore, it is natural to ask whether we can find a connection between the complexity of a decision problem and its st-connectivity problem when restricted to graph classes. This question motivated us to study the st-connectivity problems Clique Reconfiguration and Dominating Set Reconfiguration, whose decision problems' complexity for restricted graph classes is extensively studied, to get a better understanding of the boundary between polynomial-time solvable and intractable instances of these reconfiguration problems. Furthermore, we study the Cluster Vertex Deletion Reconfiguration problem, a problem whose decision problem is related to the Clique problem, to find whether there is a connection between the complexity of this problem and the Clique Reconfiguration problem.

Following are the main contributions of this thesis. First, we show that the Clique Reconfiguration problem is linear-time solvable for paths, trees, bipartite graphs, chordal graphs, and cographs. Then, we prove that the Cluster Vertex Deletion Reconfiguration problem is linear-time solvable for paths and trees, and that it is NP-hard on bipartite graphs, and PSPACE-complete in general. Finally, we determine that the Dominating Set Reconfiguration problem is linear-time solvable for paths, cographs, trees, and interval graphs. Furthermore, we show that the problem is PSPACE-complete for general graphs, bipartite graphs, and split graphs.

Acknowledgements

First and foremost I would like to thank my supervisor, Prof. Naomi Nishimura, for all her support, patience, help and guidance with my research. I have benefited a lot from her attention to detail and emphasis on clarity of presentation.

I also wish to express how grateful I am to my friends which supported me during this research, specially Tarek Bouaabdalli, Clément Chuong, Yannick Detchou, Rina Gashaw, Semany Gashaw, Peiran Guo, Amir Kahoul, Caroline Laberge, Conner Mather, Gaby A. Merhi, Biny Negash, Christian N'guessan, Lionel Nseyep, Ghiles A. Ouazzou, Nick Petroff, Aditiya Ramaraju, Paul Simon, Georgi Yankov, Omar Zerbo, and many others. I must also express my gratitude to my friends here in Waterloo, namely Abdulaziz Alarifi, Sami Bahakim, Alexandre Daigle, Martin Derka, Shahin Kamali, Daniela Maftuleac, Amer E. Mouawad, Matthew Robertson, and everyone I met through the Waterloo African Student Association.

I would also like to thank my readers Prof. Therese Biedl and Prof. Alejandro López-Ortiz. It was also a great pleasure to get to know and work with Arash Hadadan, Takehiro Ito, Hirotaka Ono, and Akira Suzuki.

Last but definitely not least, I would like to thank my family, especially my dear parents and my brother, Abderrahmane. Even though we have been living far away for the last few years, they have always been the first and greatest source of love and inspiration. This thesis is dedicated to them.

Table of Contents

List of Figures

Chapter 1

Introduction

Reconfiguration is a new field of study and has gained significant attention over the last decade. The main topic of this thesis is reconfiguration. One of the biggest impacts of algorithmic graph theory has been its usefulness in modelling real-world problems, where the domain of the problem is modelled as a graph and the constraints on the solution define feasible solutions. *Reconfiguration problems* [29] model real-life dynamic situations in which we seek to transform a solution into a more desirable one, maintaining feasibility during the process. Many puzzles can be described as reconfiguration problems. Those are the puzzles of the following type: "Given an initial *configuration* and a collection of moves, can a final configuration be reached in a finite number of moves?".

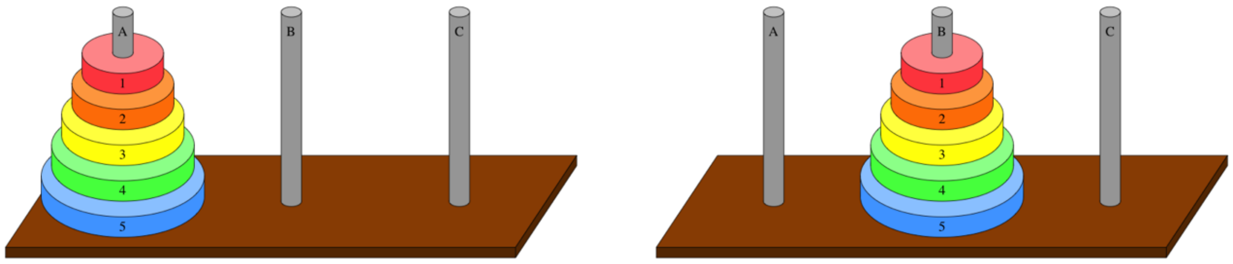


Figure 1.1: A source configuration and a target configuration of the Tower of Hanoi (source: https://www.khanacademy.org/computing/computer-science/algorithms/towers-of-hanoi/a/towers-of-hanoi)

For example, in the Tower of Hanoi, we are given three rods and a number n of disks of different sizes which can slide onto any rod (Figure 1.1). Only one disk can be moved at a time, and a disk can only be moved if it is the uppermost disk on a stack and it can only be placed on top of a bigger disk. The puzzle starts with the *source configuration* where the disks are in a stack in ascending order of size on one rod, and the goal is to reach a configuration in which all the disks form a stack in ascending order of size on another rod, the *target configuration*.

Another puzzle is the Sudoku puzzle. The goal is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 sub-grids that compose the grid contains all of the digits from 1 to 9 (Figure 1.2). A given integer may not appear twice in the same row, column, or in any of the nine 3×3 sub-grids of the 9×9 playing board. The puzzle starts with a partially completed grid as a source configuration, and the target configuration is a fully completed grid.

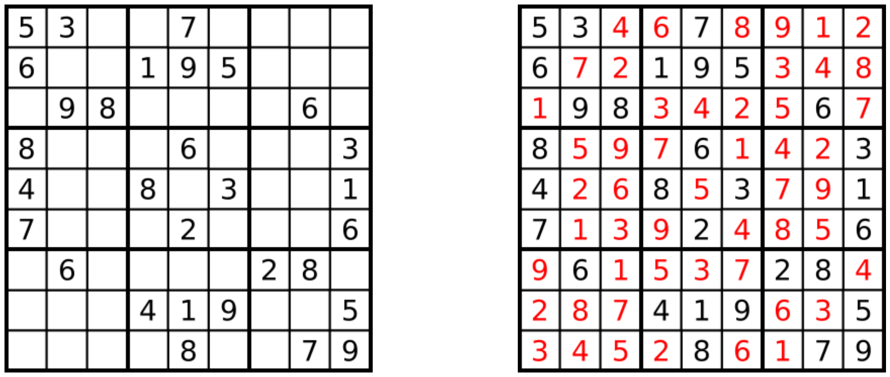


Figure 1.2: A partially completed grid and a solution of Sudoku (source: http://en.wikipedia.org/wiki/Sudoku)

The application of reconfiguration problems goes beyond puzzles. For example, consider the well-known art gallery problem [1]. This problem originates from the real-world problem of guarding an art gallery with the minimum number of guards to monitor all the rooms in the gallery. Consider a graph G where each vertex represents a room and each edge represents the adjacency of two rooms. We assume that a guard in a room can monitor the room itself and all the adjacent rooms. Then, we want to be able to monitor all the rooms by putting guards in only a subset of the rooms. Assuming we can hire at most k guards, we want to find a subset D of at most k guards that can monitor all the rooms of the gallery. For security reasons, each room can only be accessed by exactly one guard. Since guards work in shifts, we frequently need to change the subset of guards into another subset of guards. Throughout this transformation, all the rooms must be monitored (Figure 1.3). This problem is the Dominating Set Reconfiguration [22] problem where each subset of guards represents a dominating set of the graph.

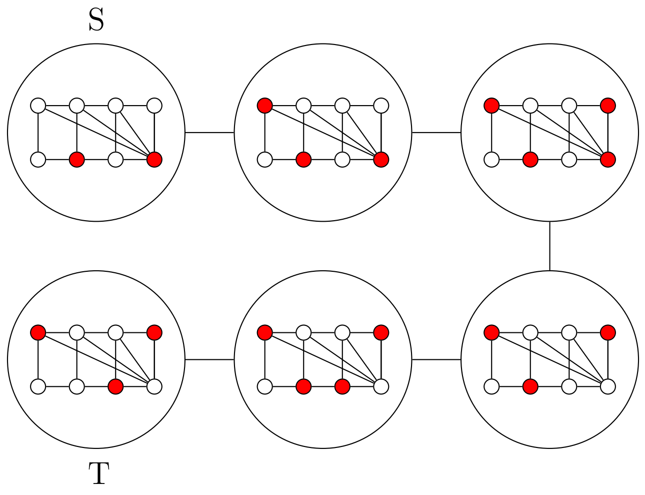


Figure 1.3: A sequence of dominating sets from S to T, where $k = 4$ and the vertices in the dominating sets are depicted by filled circles.

Figure 1.3 represents a transformation between two solutions of the art gallery problem as a graph where each node represents a feasible solution. This type of graph is called a *reconfiguration graph*. The nodes of a reconfiguration graph correspond to all the possible configurations and there is an edge between two nodes whenever the corresponding configurations can be transformed into one another with a single application of a transformation rule, a *reconfiguration step*. Given two configurations in the reconfiguration graph, one can ask if there exists a walk (*reconfiguration sequence*) from one configuration to the other. In more generality, reconfiguration problems have the following structure: a search problem $\mathcal{P}$, a minimum (maximum) size k for each solution of a minimization (maximization) problem $\mathcal{P}$, and a polynomially testable symmetric adjacency relation $\mathcal{A}$ on the set of feasible solutions of $\mathcal{P}$.

We can construct a reconfiguration graph $\mathcal{R}_{\mathcal{P}}(\mathcal{I}, \mathcal{A}, k)$ that consists of a node for each feasible solution to instance $\mathcal{I}$ of optimization problem $\mathcal{P}$, where the size of each solution is at least k for $\mathcal{P}$ a maximization problem (of size at most k for $\mathcal{P}$ a minimization problem, respectively), for positive integer k. There is an edge between two nodes of $\mathcal{R}_{\mathcal{P}}(\mathcal{I}, \mathcal{A}, k)$

whenever the corresponding solutions are adjacent under the adjacency relation $\mathcal{A}$. The *diameter* of graph G is the maximum over all pairs of vertices u and v in $V(G)$ of the length of the shortest path between u and v. We define the following reconfiguration problems that have been studied so far in the reconfiguration framework, where S and T are feasible solutions for $\mathcal{I}$:

(1) Does there exist a path between S and T in $\mathcal{R}_{\mathcal{P}}(\mathcal{I},\ \mathcal{A},\ k)$? This is known as the *st-connectivity* problem [8, 29–31].

(2) Is the reconfiguration graph $\mathcal{R}_{\mathcal{P}}(\mathcal{I}, \mathcal{A}, k)$ connected? This is called the *connectivity* problem [4, 11, 17].

(3) What is a shortest path between S and T in $\mathcal{R}_{\mathcal{P}}(\mathcal{I}, \mathcal{A}, k)$? This is known as the *shortest reconfiguration path* problem [20, 39].

(4) What is the diameter of $\mathcal{R}_{\mathcal{P}}(\mathcal{I}, \mathcal{A}, k)$ [5, 8, 30]?

In this thesis, we will focus on answering question (1) for some reconfiguration problems. We say a problem $\mathcal{P}$ is a *vertex-subset* problem when feasible solutions of $\mathcal{P}$ for graph G correspond to subsets of $V(G)$. Before we find out if there exists a path between S and T in $\mathcal{R}_{\mathcal{P}}(\mathcal{I},\ \mathcal{A},\ k)$, we have to fix the reconfiguration rule of a reconfiguration step. We will define three different reconfiguration rules of vertex-subset graph problems that were originally defined as reconfiguration rules on independent sets [34]. First, there is the *token sliding* (TS) rule, in which we can view two given feasible solutions as two sets of tokens placed on the vertices of a graph, and the reconfiguration rule is to slide a single token along an edge. Second, there is the *token jumping* (TJ) rule, in which one is allowed to move a single token of a feasible solution to any other vertex, forming another feasible solution. Finally, there is the *Token Addition and Removal* (TAR) rule where one can add or remove a single token at a time as long as the resulting set is a feasible solution. Two adjacent nodes in the reconfiguration graphs under the TS and TJ rules are always of the same cardinality since no token is being added or removed. In this thesis, we are interested in reconfiguration search problems that arise from optimization problems. Optimization problems are turned into search problems by setting a threshold (upper bound for minimization problems, lower bound for maximization problems) and the cardinality of each feasible solution must be bounded by that threshold. Since the cardinality of feasible solutions of these problems can vary, we only use the TAR rule as our reconfiguration rule.

The idea of reconfiguration problem in the mathematical literature is not a new concept, but there has been recent interest in reconfiguration problems from the point of view of computational problems. The reconfiguration framework has been applied for a number of problems, including satisfiability [20], independent set [29, 34], vertex coloring [9, 11], list-edge coloring [30], clique, set cover, integer programming, matching [29], block puzzles [25], dominating set [22, 48] and so on.

One of the first key results under the reconfiguration framework was obtained by the study of the satisfiability problem [20]. The complexity questions of the connectivity problem, the st-connectivity problem and the maximal diameter of the reconfiguration graph have all been addressed for the satisfiability problem [45].

The notion of reconfiguration framework was first introduced by Ito et al. [29] when they studied the complexity of st-connectivity in the reconfiguration graph of a host of optimization problems. They proved that the problem of st-connectivity for the NP-complete problems Power Supply, Independent Set, Clique, Vertex Cover, Set Cover and Integer Programming is PSPACE-complete. For most NP-complete problems, the st-connectivity problem has been shown to be PSPACE-complete [29, 30, 34], while for some problems in P, this problem could be either in P [29] or PSPACE-complete [7]. The 3-colorability problem [9] is an example of an NP-complete problem whose st-connectivity problem is in P, and Shortest Path [6] is an example of a problem that is in P but whose st-connectivity problem is PSPACE-complete. Thus, knowing the complexity of a decision problem does not allow us to directly infer the complexity of its st-connectivity problem.

Hence, it would be very interesting to find a connection between the complexity of decision problems and the complexity of the corresponding reconfiguration problem to get a better understanding of both problems. Therefore, it is natural to ask whether there is a connection between these two problems when restricted to graph classes, meaning whether the st-connectivity problem for an NP-complete problem, when restricted to a specific graph, is also PSPACE-complete and in P for problems in P. This question motivated us to study the st-connectivity problems Clique Reconfiguration and Dominating Set Reconfiguration, whose decision problems' complexity for restricted graph classes is extensively studied, to get a better understanding of the boundary between polynomial-time solvable and intractable instances of these reconfiguration problems. Furthermore, we study the Cluster Vertex Deletion Reconfiguration problem, a problem whose decision problem is related to the Clique problem, to find whether there is a connection between the complexity of this problem and the Clique Reconfiguration problem.

Recently, there has been an interest in studying the complexity of the st-connectivity problem for Independent Set [34] and Dominating Set [22, 48] when restricted to graph classes. The study of Independent Set Reconfiguration [34] provided another example of a problem in P whose st-connectivity problem is PSPACE-complete when restricted to perfect graphs. The study of Dominating Set Reconfiguration [22, 48] proved that the reconfiguration graph of this problem is connected when $k = n - 1$ and G

has at least two non-adjacent edges where n represents the number of nodes of the original problem, or when G is a chordal or a bipartite graph and k is one greater than the maximum cardinality of a minimal dominating set. In this thesis we build on this work and study the st-connectivity problem for Dominating Set on a collection of graph classes. We also study the well-known Clique problem and the related Cluster Vertex Deletion [21] problem to obtain a better understanding of their st-connectivity problems and of the connections between their complexities.

This thesis is organized as follows. In Chapter 2, we introduce the terminology used throughout the thesis, and formally define the three underlying problems being studied. In Chapter 3, we familiarize the reader with the problems considered in this thesis by providing a few examples and provide a summary of the known and relevant results related to them. Chapter 4 provides a general scheme to build linear-time algorithms for the Cluster Vertex Editing Reconfiguration and the Dominating Set Reconfiguration problems. In Section 5 we prove that Clique Reconfiguration can be solved in time linear in the number of edges for paths, trees, bipartite graphs, chordal graphs, and cographs. In Section 6, we first prove that Cluster Vertex Editing Reconfiguration can be solved in time linear in the number of edges for paths and trees. Then, we show that the problem is NP-hard on bipartite graphs. Finally, we prove that it is PSPACE-complete in general. In Section 7, we first prove that Dominating Set Reconfiguration can be solved in time linear in the number of edges for paths, cographs, trees, and interval graphs. Then, we show that the problem is PSPACE-complete for general graphs, bipartite graphs, and split graphs. Concluding remarks, open problems, and directions for future research are discussed in Chapter 8.

Chapter 2

Preliminaries

In this chapter, we define some terms, definitions, and notation which will be used throughout the thesis, define the underlying problems of the problems studied in this thesis, and provide some preliminary results. Section 2.1 outlines the notation that will used throughout the thesis, and the formal definitions of the graph classes for which we will prove the complexity of our problems. In Section 2.2, we provide formal definitions of the underlying problems whose st-connectivity problem is studied in the following chapters. In Section 2.3, we provide all the notation and definitions related to reconfiguration as well as general results of reconfiguration problems.

2.1 Graph Theory

The following graph theoretic definitions can be found in Diestel's book [13]. A *simple graph* is an undirected graph that has no loops (edges connected at both ends to the same vertex) and no more than one edge between any two different vertices. We assume that each input graph G is a simple graph with vertex set $V(G)$ and edge set $E(G)$, where $|V(G)| = n$ and $|E(G)| = m$. A *subgraph* of G is a graph G' such that $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. For a set $S \subseteq V(G)$ of vertices, the subgraph of G *induced* by S is denoted by $G[S]$, where $G[S]$ has vertex set S and edge set $\{uv \in E(G[S]) \mid u, v \in S, uv \in E(G)\}$. For a vertex v in a graph G, let $N_G(v) = \{u \in V(G) \mid vu \in E(G)\}$ and $N_G[v] = N_G(v) \cup \{v\}$. For a set $S \subseteq V(G)$ of vertices, we define $N_G[S] = \bigcup_{v \in S} N_G[v]$ and $N_G(S) = N_G[S] \setminus S$. The *degree* of a vertex v is $|N_G(v)|$. We sometimes drop the subscript G if it is clear from context.

A *walk* of length l from v_0 to v_l in G is a vertex sequence $v_0, \ldots, v_l$ such that for all

$i \in \{0, \ldots, l-1\}$, $v_i v_{i+1} \in E(G)$. It is a *path* if all the vertices are distinct. It is a *cycle* if $l \geq 3$, $v_0 = v_l$ and $v_0, \ldots, v_{l-1}$ is a path. An *induced path* in an undirected graph G is a path that is an induced subgraph of G. The *complement* $\overline{G}$ of a graph G is a graph with $V(\overline{G}) = V(G)$ and with the property that two vertices are adjacent in $\overline{G}$ if and only if they are not adjacent in G. We say that a graph class $\mathcal{G}$ (i.e., a set of graphs) is closed under taking complements if $\overline{G} \in \mathcal{G}$ holds for every graph $G \in \mathcal{G}$.

A vertex or edge set satisfying a specific condition is *minimal (maximal)* (with respect to the condition) if no proper subset (superset) of the set satisfies the condition. A graph G is *connected* if there is a path between every pair of vertices. A *connected component* of G is an induced subgraph C of G such that $V(C)$ is a maximal subset of $V(G)$ such that $G[V(C)]$ is connected. The *diameter* of graph G is the maximum over all pairs of vertices u and v in $V(G)$ of the length of the shortest path between u and v. A *matching* $M \in E(G)$ in a graph G is a set of edges of G such that no two of them share a vertex. We write $V(M)$ to denote the set of vertices incident to edges in M. An *induced matching* of G is the set of edges $M \subseteq E(G)$ such that M is a matching and for any distinct edges uu', $vv' \in M$, G has none of the edges in $\{uv, uv', u'v, u'v'\}$.

We adopt known conventions for referring to some special graphs [10]. P_k denotes a path on k vertices and $k-1$ edges. C_k denotes a cycle on k vertices and k edges. *Trees* are connected graphs without cycles. A *subtree* of a tree is a connected subgraph of the tree. A *complete graph* is a simple undirected graph in which each pair of distinct vertices is connected by a single edge. A *spanning tree* of a graph G is a subgraph of G that contains all of the vertices of G. A *maximum spanning tree* is a spanning tree of a weighted graph having maximum weight. A *clique* is a subset S of vertices of an undirected graph such that the induced subgraph of S is complete. An *independent set* is a subset S of vertices of an undirected graph such that for $u, v \in S$, $uv \notin V(G)$. A *vertex cover* is a subset S of vertices of an undirected graph G such that $G[V(G) \setminus S]$ is edgeless.

The following definition, fact, and notation will be used in defining the Cluster Vertex Deletion problem.

Definition 2.1.1. *A graph is a* cluster graph *if every connected component forms a clique.*

Fact 2.1.1. *A cluster graph does not contain an induced path of three vertices.*

For a graph G, a set $D \subseteq V(G)$ is a *deletion set* of G if the removal of D transforms G into a cluster graph, that is, a collection of disjoint cliques. Note that $V(G)$ always forms a deletion set of G.

Figure 2.1 illustrates some inclusion relationships among well-known graph classes. For a problem $\mathcal{P}$, we want to find the boundary in graph classes between polynomial-time

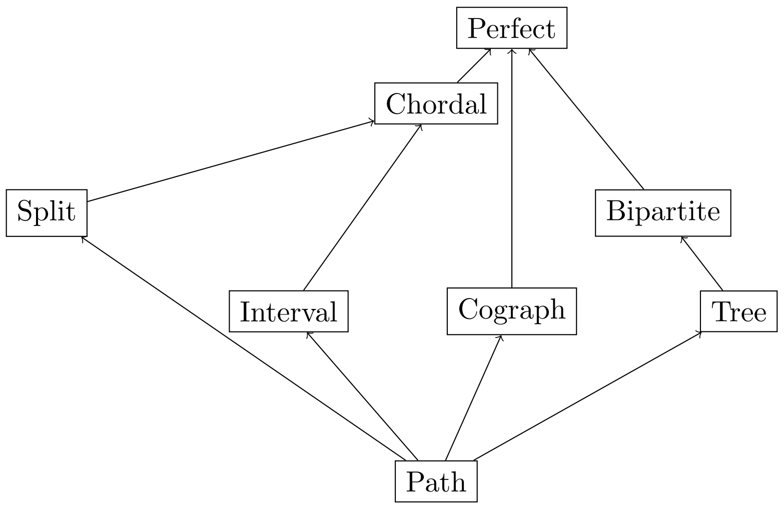


Figure 2.1: Inclusion relationships among various graph classes defined in Section 2.1, where each arrow represents the inclusion relationship between graph classes: $A \to B$ represents that A is properly included in B [10].

solvability and intractability. Thus, if $\mathcal{P}$ has a polynomial-time algorithm for a graph class A, we choose a graph class B that includes A and find the complexity of $\mathcal{P}$ in this graph. Similarly, if $\mathcal{P}$ is PSPACE-complete for a graph class A, we choose a graph class B that is included in A and find the complexity of $\mathcal{P}$ in this graph. When constructing polynomial-time algorithms, we usually start with paths and to prove PSPACE-completeness, we start with general graphs. We now give formal definitions of some graph classes for which we will prove the complexity of the three main problems of this thesis.

Definition 2.1.2. *A graph G is* bipartite *if there exists a partition (A, B) of its vertex set such that $G[A]$ and $G[B]$ are edgeless.*

$K_{n,m}$ denotes a *biclique*, i.e. a complete bipartite graph with n vertices in the first partition, and m vertices in the second. For bipartite graphs, the Clique and Dominating Set problems are in P [33] and the Cluster Vertex Deletion problem is NP-complete [36]. Therefore, it is interesting to study the the st-connectivity problem for these problems for bipartite graphs and show whether it has the same complexity as their underlying problems. We show in this thesis that this conjecture is true.

Based on the fact that we can compute a tree of maximal cliques of a chordal graph G in time linear in $|E(G)|$ [46] and the fact that the Clique problem is in P for chordal graphs, we ask whether the st-connectivity problem is also in P. We prove that this problem is in P in Chapter 5.

Definition 2.1.3. *A graph is* chordal *if every induced cycle is of length three.*

We denote by $\mathcal{MC}(G)$ the set of all maximal cliques of a chordal graph G and we denote by $\mathcal{MC}(G, v)$ the set of maximal cliques of $V(G)$ that contain a vertex $v \in V(G)$. The maximal cliques of a chordal graph can be connected to form a *clique tree* [26]. A tree $\mathcal{T}$ is a clique tree of G if each node of $\mathcal{T}$ corresponds to a maximal clique in $\mathcal{MC}(G)$, and the cliques containing a vertex $v \in V(G)$ yield a subtree of $\mathcal{T}$. Each edge of $\mathcal{T}$ is assigned the set that represents the intersection of the maximal cliques defined by its endpoints. Figure 2.2 (a) illustrates a chordal graph G, and (b) a clique tree T of G. The set S representing the edge $E = (M, M')$ where $M, M' \in \mathcal{T}$ is a proper subset of M and M'. The intersection graph I of $\mathcal{MC}(G)$ is the graph such that each maximal clique of $\mathcal{MC}(G)$ is represented by a vertex, and two vertices of I are connected if and only if the corresponding maximal cliques intersect [19]. A clique tree is a maximum spanning tree of the intersection graph I of G, where the weight of an edge is defined as the size of the intersection [23]. Since a clique tree $\mathcal{T}$ is a maximum spanning tree of the intersection graph I of G, the weight of the minimum-weight edge in the path between two nodes $A, B \in \mathcal{T}$ is the maximum among the minimum-weight edges of all possible paths between A and B in I [27]. We use $\alpha(\mathcal{T})$ to denote the minimum cardinality of any set S represented by an edge of $\mathcal{T}$. We will refer to vertices in G using lower-case letters and to nodes in $\mathcal{T}$ using upper-case letters.

We now define the join and union operations that are used in building cographs. The *union* $G = G_1 \cup G_2$ of graphs G_1 and G_2 with disjoint vertex sets V_1 and V_2 and edge sets E_1 and E_2 is the graph with $V(G) = V_1 \cup V_2$ and $E(G) = E_1 \cup E_2$. The *join* $G = G_1 \vee G_2$ of graphs G_1 and G_2 is represented as $\overline{(\overline{G_1} \cup \overline{G_2})}$. Based on the fact that the Independent Set Reconfiguration problem is in P for cographs [34], it is interesting to show whether this problem can be reduced to one of our problems. We show that there exists such a reduction to the Clique Reconfiguration problem, defined in the next section.

Definition 2.1.4. Complement reducible graphs*, also called* cographs*, are defined recursively as follows [12]:*

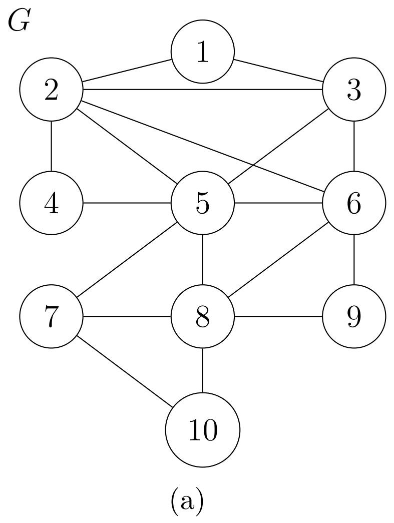
(1) A graph consisting of a single vertex is a cograph.

(2) If G_1 and G_2 are two (disjoint) cographs, then so is their union $G_1 \cup G_2$.

(3) If G is a cograph, then so is its complement $\overline{G}$.

Fact 2.1.2. *Cographs have no induced path on four vertices. Therefore, they are also called* P_4-free graphs *or* P_4 restricted graphs *[12].*

Fact 2.1.3. *Graph G is a cograph if it can be constructed from isolated vertices by disjoint union and join operations [12].*



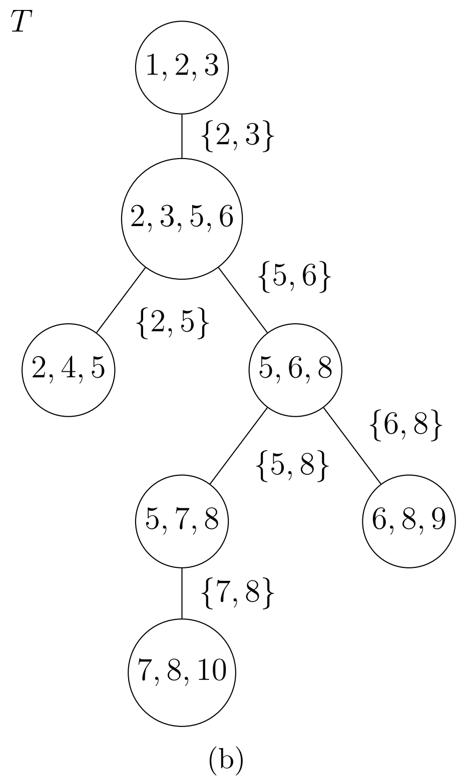


Figure 2.2: (a) A chordal graph G, (b) a clique tree T of G.

If a graph G is an interval graph, we can obtain an interval representation of G in linear time [35]. Therefore, since two intervals intersect only if there exists an edge between their respective vertices and since the Dominating Set problem is in P for cographs, it is interesting to ask whether its st-connectivity is also in P. We show that this problem is in fact in P.

Definition 2.1.5. *A graph G with $V(G) = \{v_1, v_2, \ldots, v_n\}$ is an* interval graph *if there exists a set $\mathcal{I}$ of (closed) intervals $I_1, I_2, \ldots, I_n$ such that $v_iv_j \in E(G)$ if and only if $I_i \cap I_j \neq \emptyset$ for each $i, j \in \{1, 2, \ldots, n\}$. We call the set $\mathcal{I}$ of intervals an* interval representation *of the graph.*

We show in Section 7.2.2 that the Dominating Set Reconfiguration problem, defined in the next section, is PSPACE-complete for split graphs.

Definition 2.1.6. *A graph is is a* split graph *if its vertex set can be partitioned into a clique and an independent set [10].*

2.2 Graph Problems

All the underlying problems of the reconfiguration problems studied in this thesis are graph vertex-subset problems. A problem $\mathcal{P}$ is a *vertex-subset* problem when feasible solutions of $\mathcal{P}$ for graph G correspond to subsets of $V(G)$. We say a problem $\mathcal{P}$ is a *vertex-subset maximization (minimization)* problem whenever the optimal solution of $\mathcal{P}$ corresponds to a subset of $V(G)$ of maximum (minimum) size.

A classical example of a vertex-subset minimization problem is the Vertex Cover (VC) problem. The corresponding decision problem is defined as follows:

Input: An undirected graph G and a non-negative integer k.
Question: Is there $S \subseteq V(G)$ such that $|S| \leq k$ and $G[V(G) \backslash S]$ is edgeless?

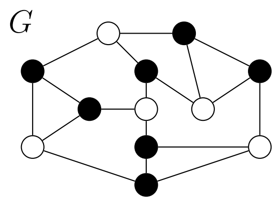


Figure 2.3: Example of a vertex cover (filled circles) and an independent set (white circles) of a graph G.

An example of a vertex-subset maximization problem is the Independent Set (IS) problem. The corresponding decision problem is defined as follows::

Input: An undirected graph G and a non-negative integer k.
Question: Is there $I \subseteq V(G)$ such that $|I| \geq k$ and $G[I]$ is edgeless?

Both of these problems have been proven to be NP-complete [33]. It is easy to see, by the definition of a vertex cover and an independent set, that if S is a vertex cover of G, then $G \setminus S$ forms an independent set of G, and vice versa. Figure 2.3 illustrates a vertex cover and an independent set of a graph G. The relationship between these two sets gives us a reduction between the two problems and proves that these two problems are equally hard [18].

We now introduce the underlying vertex-subset problems of the reconfiguration problems studied in this thesis.

The first problem is the CLIQUE problem, which is a vertex-subset maximization problem that is intensively studied in graph theory [43]. A clique C of G is *maximal* if it is not an induced subgraph of any larger clique.

The corresponding CLIQUE (CL) decision problem is defined as follows:

> **Input:** An undirected graph G and a non-negative integer k.
> **Question:** Does G contain a clique C with $|C| \geq k$?

The following proposition shows the relation between a clique and an independent set of a graph G. This relation will be used to prove the results of Section 5.4.

Proposition 2.2.1. *For $S \subseteq G$, S is a clique in G if and only if S is an independent set in the complement $\overline{G}$ of G.*

Figure 2.4 part (a) illustrates a clique S of four vertices depicted by filled circles and part (b) shows that S is an independent set in $\overline{G}$. Thus, a simple reduction from the decision problem of INDEPENDENT SET proves that this problem is NP-complete [18].

The second problem is the CLUSTER VERTEX DELETION (CVD) problem [28], a vertex-subset minimization problem whose corresponding decision problem is defined as follows:

> **Input:** An undirected graph G and a non-negative integer k.
> **Question:** Does G contain a deletion set $S \subseteq V$ with $|S| \leq k$?

Fact 2.2.1. *If S is a deletion set of G, then $S \cap A$ is a deletion set of $G[A]$, where $A \subseteq V(G)$.*

Figure 2.4 (b) shows a deletion set of size 4 depicted by filled circles.

Finally, the third problem is the vertex-subset minimization DOMINATING SET problem. For a graph G, a set $D \subseteq V(G)$ is a *dominating set* of G if $N[D] = V(G)$. Note that $V(G)$ always forms a dominating set of G. For a vertex $u \in V(G)$ and a dominating set D of G, we say u is *dominated* by $v \in D$ if $u \notin D$ and u is adjacent to v. Figure 2.3 shows a dominating set of size 5 depicted by white circles.

The corresponding DOMINATING SET (DS) decision problem is defined as follows:

> **Input:** An undirected graph G and a non-negative integer k.
> **Question:** Does G contain a dominating set $D \subseteq V(G)$ with $|D| \leq k$?

Observation 2.2.1. *If S is a feasible set of G for a vertex-subset minimization problem P, then any superset S' of S is also a feasible set of G.*

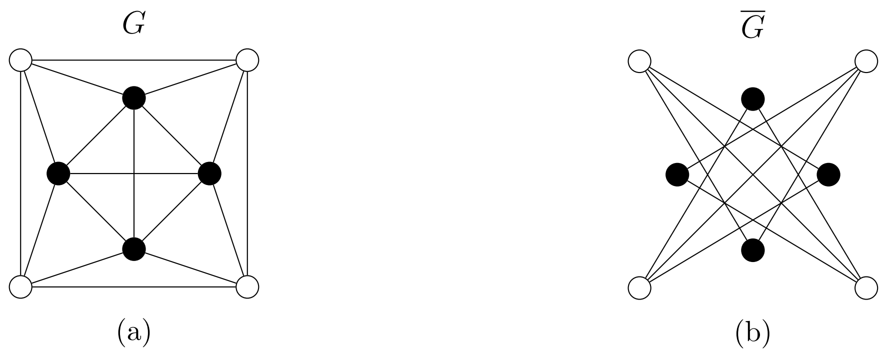


Figure 2.4: (a) A graph G, (b) the complement graph $\overline{G}$ of graph G.

2.3 Reconfiguration

For any vertex-subset problem $\mathcal{P}$, graph $G = (V, E)$, and a positive integer k, we consider the *reconfiguration graph* $\mathcal{R}_{\mathcal{P}}(G, k)$. A set $S \subseteq V$ has a corresponding node in $V(\mathcal{R}_{\mathcal{P}}(G, k))$ if and only if S is a feasible solution for P and $|S| \leq k$ ($|S| \geq k$) if P is a minimization (maximization) problem. We write $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$ ($\mathcal{R}_{\mathcal{P}\text{-MAX}}(G, k)$) to refer to the reconfiguration graph of a vertex-subset minimization (maximization) problem $\mathcal{P}$.

We refer to *vertices* in G using lower case letters (e.g. u, v) and to the *nodes* in $\mathcal{R}_{\mathcal{P}}(G, k)$, and by extension their associated dominating sets, using upper case letters (e.g. A, B). Given $A, B \in V(\mathcal{R}_{\mathcal{P}}(G, k))$, we let $A \Delta B = (A \setminus B) \cup (B \setminus A)$ denote the *symmetric difference* of A and B. Formally, there exists an edge between A and B in $\mathcal{R}_{\mathcal{P}}(G, k)$ if and only if there exists a vertex $u \in V(G)$ such that $(A \setminus B) \cup (B \setminus A) = \{u\}$. Equivalently, there exists an edge between A and B in $\mathcal{R}_{\mathcal{P}}(G, k)$ if and only if $|A \Delta B| = 1$. We say that A and B are *adjacent* if their symmetric difference has size 1.

We write $A \leftrightsquigarrow B$ if there exists a path in $R_P(G, k)$, a *reconfiguration sequence*, joining A and B. Two nodes $A, B \subseteq V(R_P(G, k))$ share an edge whenever the corresponding configurations can each be obtained from the other by the application of a single transformation rule, a *reconfiguration step*. Any reconfiguration sequence between a *source* feasible solution S_s and a *target* feasible solution S_t, denoted $\langle S_0, S_1, \ldots, S_\ell \rangle$ for some positive integer l, has the following properties:

(a) $S_0 = S_s$ and $S_\ell = S_t$;
(b) S_{i-1} and S_i are adjacent for each $i \in \{1, 2, \ldots, \ell\}$; and
(c) S_i is a feasible solution for P for each $i \in \{1, 2, \ldots, \ell\}$.

We say that a reconfiguration sequence $\langle S_0, S_1, \ldots, S_\ell \rangle$ of feasible solutions of P between S_s and S_t is *reversible* if its reverse $\langle S_\ell, S_{\ell-1}, \ldots, S_0 \rangle$ is also a reconfiguration sequence between S_t and S_s. Note that any reconfiguration sequence is reversible since if the sequence $\langle S_0, S_1, \ldots, S_\ell \rangle$ of feasible solutions of P is a reconfiguration sequence between S_s and S_t, then its reverse sequence $\langle S_\ell, S_{\ell-1}, \ldots, S_0 \rangle$ is a reconfiguration sequence between S_t and S_s. We say vertex $v \in V$ is *added* at step i if $v \notin S_{i-1}$ and $v \in S_i$ for each $i \in \{1, 2, \ldots, \ell\}$. Similarly, a vertex v is *deleted* at step i if $v \in S_{i-1}$ and $v \notin S_i$ for each $i \in \{1, 2, \ldots, \ell\}$. We say a vertex $v \in V$ is *touched* in a reconfiguration sequence σ if v is either added or deleted at least once in σ. A vertex w in a feasible solution S is *deletable* if $S \setminus \{w\}$ is also a feasible solution of P. A feasible solution S of G is *minimal* if there is no deletable vertex in S. Similarly, a vertex w in a feasible solution S is *addable* if $S \cup \{w\}$ is also a feasible solution of P. A feasible solution S of G is *maximal* if there is no addable vertex in S. We use $\lambda(G)$ to denote the minimum cardinality of any feasible solution of a graph G. Similarly, $\Lambda(G)$ is the maximum cardinality of any feasible solution of a graph G.

Definition 2.3.1. *For any vertex-subset minimization problem $\mathcal{P}$, graph G, positive integer k, $S_s \subseteq V(G)$, and $S_t \subseteq V(G)$, we define the following decision problem:*

$\mathcal{P}$-MIN-R: For $S_s, S_t \in V(\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k))$, is there a path between S_s and S_t in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$?

Definition 2.3.2. *For any vertex-subset maximization problem $\mathcal{P}$, graph G, positive integer k, $S_s \subseteq V(G)$, and $S_t \subseteq V(G)$, we define the following decision problem:*

$\mathcal{P}$-MAX-R: For $S_s, S_t \in V(\mathcal{R}_{\mathcal{P}\text{-MAX}}(G, k))$, is there a path between S_s and S_t in $\mathcal{R}_{\mathcal{P}\text{-MAX}}(G, k)$?

We denote by (G, S_s, S_t, k) an instance of $\mathcal{P}$-MIN-R ($\mathcal{P}$-MAX-R).

The definition of the st-connectivity problem for an underlying optimization problem follows naturally from the definition of the optimization problem itself. We use INDEPENDENT SET (IS) as an example below, which corresponds to the $\mathcal{IS}$-MAX-R problem, defined as follows:

Input: A graph G, two independent sets D_s and D_t of G, and an integer threshold $k \leq \min\{|D_s|, |D_t|\}$.
Question: Is there a path from D_s to D_t in $\mathcal{R}_{\mathcal{IS}\text{-MAX}}(G, k)$?

By Proposition 2.2.1, we know that a subset S of G forms a clique in G if and only if S forms an independent set in $\overline{G}$. Using this fact, some known results for $\mathcal{IS}$-MAX-R can be converted into results for the st-connectivity problem for CLIQUE denoted as $\mathcal{CL}$-MAX-R. It is known that the $\mathcal{IS}$-MAX-R problem for a cograph G is solvable in time linear in $|E(G)|$ [34]. We will show in Section 5.4 that this result can be converted into a result for $\mathcal{CL}$-MAX-R.

In the following proposition, we show conditions for a no-instance of $\mathcal{P}$-Min-R.

Proposition 2.3.1. *Given an instance (G, S_s, S_t, k) of $\mathcal{P}$-Min-R, if any of the following conditions are true then we have a no-instance:*
(1) $k < \lambda(G)$,
(2) $k = \lambda(G)$ and $S_s \neq S_t$,
(3) S_s is minimal, $k = |S_s|$ and $S_s \neq S_t$, or
(4) S_t is minimal, $k = |S_t|$ and $S_s \neq S_t$.

Proof. In case (1), if $k < \lambda(G)$ then there does not exist any solution of size k since $\lambda(G)$ is the minimum cardinality of any feasible solution of G. In case (2), if $k = \lambda(G)$ and S_s and S_t are feasible solutions of size less than or equal to k, S_s and S_t are feasible solutions of G of minimum cardinality. Since $|S_s| = |S_t| = \lambda(G)$, there does not exist a deletable vertex in S_s or S_t. Hence, S_s and S_t are isolated nodes in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$. Similarly, in cases (3) and (4), if S_s (S_t) is minimal and $k = |S_s|$ ($k = |S_t|$), there does not exist a deletable vertex in S_s (S_t). Hence, S_s (S_t) is an isolated node in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$. □

Fact 2.3.1. *Given an instance (G, S_s, S_t, k) of $\mathcal{P}$-Min-R, if $S_s = S_t$ and $k \geq \lambda(G)$, then we have a yes-instance.*

In the following proposition, we show conditions for a no-instance of $\mathcal{P}$-Max-R.

Proposition 2.3.2. *Given an instance (G, S_s, S_t, k) of $\mathcal{P}$-Max-R, if any of the following conditions are true then we have a no-instance:*
(1) $k > \Lambda(G)$,
(2) $k = \Lambda(G)$ and $S_s \neq S_t$,
(3) S_s is maximal, $k = |S_s|$ and $S_s \neq S_t$, or
(4) S_t is maximal, $k = |S_t|$ and $S_s \neq S_t$.

Proof. In case (1), if $k > \Lambda(G)$ then there does not exist any solution of size k since $\Lambda(G)$ is the maximum cardinality of any feasible solution of G. In case (2), if $k = \Lambda(G)$ and S_s and S_t are feasible solutions of size greater than or equal to k, S_s and S_t are feasible solutions of G of maximum cardinality. Since $|S_s| = |S_t| = \Lambda(G)$, there does not exist an addable vertex in S_s or S_t. Hence, S_s and S_t are isolated nodes in $\mathcal{R}_{\mathcal{P}\text{-Max}}(G, k)$. Similarly, in cases (3) and (4), if S_s (S_t) is maximal and $k = |S_s|$ ($k = |S_t|$), there does not exist an addable vertex in S_s (S_t). Hence, S_s (S_t) is an isolated node in $\mathcal{R}_{\mathcal{P}\text{-Max}}(G, k)$. □

Proposition 2.3.3. *Given an instance (G, S_s, S_t, k) of $\mathcal{P}$-Max-R, if $k = 0$ then we have a yes-instance.*

Proof. Since $k = 0$, there exists a reconfiguration sequence from S_s to S_t in $\mathcal{R}_{\mathcal{P}\text{-}\textsc{Max}}(G, k)$ by removing all vertices of S_s one by one, and then adding all the vertices of S_t one by one. $\square$

Fact 2.3.2. *Given an instance (G, S_s, S_t, k) of $\mathcal{P}$-*Max-R*, if $S_s = S_t$ and $k \leq \Lambda(G)$, then we have a* yes*-instance.*

Chapter 3

Related Work

In this thesis, we are interested in finding a connection between the complexity of decision problems and their respective st-connectivity problems, when restricted to graph classes, to get a better understanding of both problems. Therefore, we work on the two extensively studied problems Clique and Dominating Set, and find results of the complexity of their st-connectivity problems for several graphs. Furthermore, we study the st-connectivity problem for the Cluster Vertex Deletion problem, a problem related to the Clique problem, to find whether there is a connection between their corresponding st-connectivity problems.

Reconfiguration of the Clique problem. The first problem studied in this thesis is the Clique Reconfiguration problem ($\mathcal{CL}$-Max-R). This problem is defined as follows:

Input: A graph G, two cliques S_s and S_t of G, and an integer threshold $k \leq \min\{|S_s|, |S_t|\}$.
Question: Is there a path from S_s to S_t in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$?

The reconfiguration graph $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$ of a graph G contains all cliques of G of size at least k as its node set, and two nodes share an edge whenever the corresponding cliques differ on exactly one vertex. An example of a reconfiguration sequence between two cliques of size 3 of a graph is given in Figure 3.1. Every feasible solution of size greater than or equal to 3 is shown as solid circles. Note that $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, 3)$ is not connected, which implies that there exists at least one pair of feasible solutions of this instance for which there does not exist a reconfiguration sequence.

Ito et al. [29] proved that the st-connectivity problem for a number of NP-complete problems can be shown to be PSPACE-complete by extending the idea developed for the NP-completeness proof of the original problem. To show that $\mathcal{CL}$-Max-R is PSPACE-complete, they used a polynomial-time reduction from $\mathcal{IS}$-Max-R by extending the well-known reduction from the Independent Set problem to the Clique problem. Similarly, they showed that $\mathcal{IS}$-Max-R is PSPACE-complete by extending the original NP-completeness proof that uses a reduction from the 3SAT problem [42], whose st-connectivity problem is also PSPACE-complete [20]. They also showed that the problem of computing the maximum threshold k such that there exists a reconfiguration sequence between two nodes of $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G,k)$ for a graph G cannot be approximated within any constant factor unless $\mathtt{P} = \mathtt{NP}$. To the best of our knowledge, these are all the results in the literature for the $\mathcal{CL}$-Max-R problem.

The Independent Set Reconfiguration problem [34] is one of the most well-studied reconfiguration problems. It is one of the only problems whose complexity was studied restricted to graph classes. It was proven that $\mathcal{IS}$-Max-R is solvable in linear time for even-hole-free graphs and cographs under the token sliding rule [34]. It is well known that the Independent Set problem and the Clique problem are related, since a clique C in a graph G forms an independent set in the complement $\overline{G}$ of G. Thus, given that we have these results for $\mathcal{IS}$-Max-R, it is desirable to obtain results to related problems and extend previous work. Therefore, in this thesis, we study the $\mathcal{CL}$-Max-R problem and delineate its complexity restricted to various graph classes. During the study of this problem, a paper on the Clique Reconfiguration problem [32] that studies this problem in the same context was published. We present in this thesis the results that were obtained independently of this work. We prove that $\mathcal{CL}$-Max-R can be solved in time linear in the number of edges for paths, trees, bipartite graphs, chordal graphs, and cographs.

Reconfiguration of the Cluster Vertex Deletion problem. The second problem studied is the Cluster Vertex Deletion problem ($\mathcal{CVD}$-Min-R). This problem is defined as follows:

Input: A graph G, two deletion sets S_s and S_t of G, and an integer threshold $k \geq \max\{|S_s|, |S_t|\}$.
Question: Is there a path from S_s to S_t in $\mathcal{R}_{\mathcal{CVD}\text{-Min}}(G,k)$?

Figure 3.2 shows the reconfiguration graph $\mathcal{R}_{\mathcal{CVD}\text{-Min}}(G,3)$ for an input graph G consisting of an instance of CVD and $k = 3$. Every feasible solution of size less than or equal to 3 is shown as solid circles. In contrast to the $\mathcal{CL}$-Max-R problem considered above,

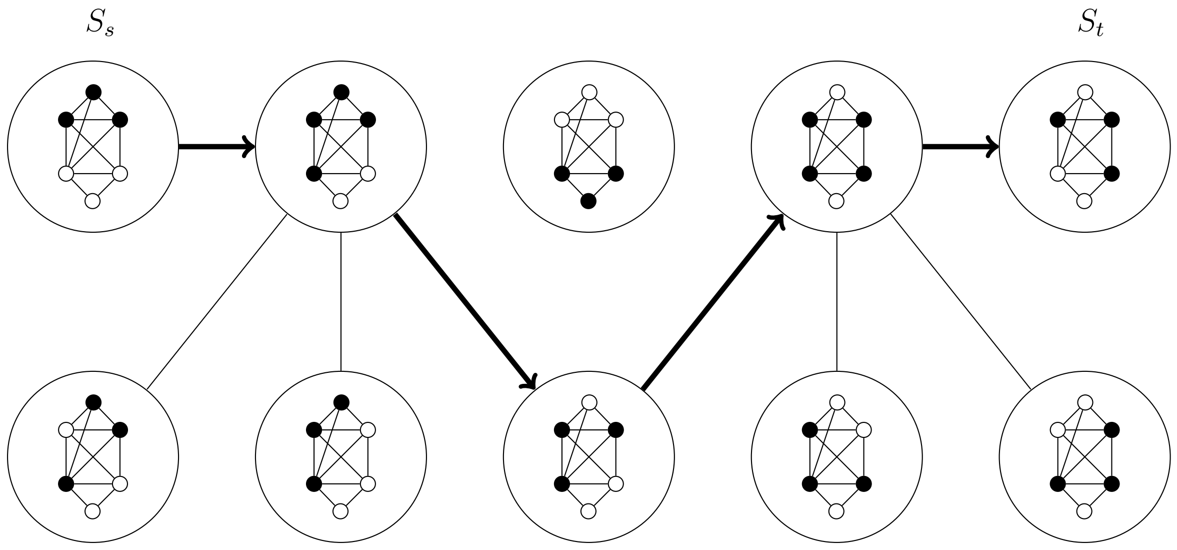


Figure 3.1: A reconfiguration sequence from S_s to S_t in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, 3)$ of length 4 (shown in thick lines) for a graph G.

$\mathcal{R}_{\mathcal{CVD}\text{-Min}}(G, 3)$ is connected, which implies that there exists a reconfiguration sequence between any two feasible solutions of this instance. Also, note that in $\mathcal{R}_{\mathcal{CVD}\text{-Min}}(G, 3)$, there is a path of size at most 4 between any two feasible solutions of this instance.

Note that the underlying decision problem of $\mathcal{CVD}$-Min-R asks to find a set $S \subseteq V(G)$ such that the removal of S from G results in a graph where every connected component forms a clique. Having studied the $\mathcal{CL}$-Max-R problem, and since there is a connection between the underlying decision problems of $\mathcal{CVD}$-Min-R and $\mathcal{CL}$-Max-R, we were interested in studying the $\mathcal{CVD}$-Min-R problem in the same context. We studied this problem when restricted to some graph classes that we studied for $\mathcal{CL}$-Max-R to find a connection between the two related reconfiguration problems under these same graph classes. We show that, similarly to $\mathcal{CL}$-Max-R, $\mathcal{CVD}$-Min-R can be solved in time linear in the number of edges for paths and trees. However, on bipartite graphs, the $\mathcal{CL}$-Max-R problem is solvable in time linear in the number of edges and we prove that the $\mathcal{CVD}$-Min-R problem is `NP`-hard. Finally, we prove that it is `PSPACE`-complete in general.

Reconfiguration of the Dominating Set problem. The last problem studied is the st-connectivity problem for the Dominating Set problem ($\mathcal{DS}$-Min-R). This problem is

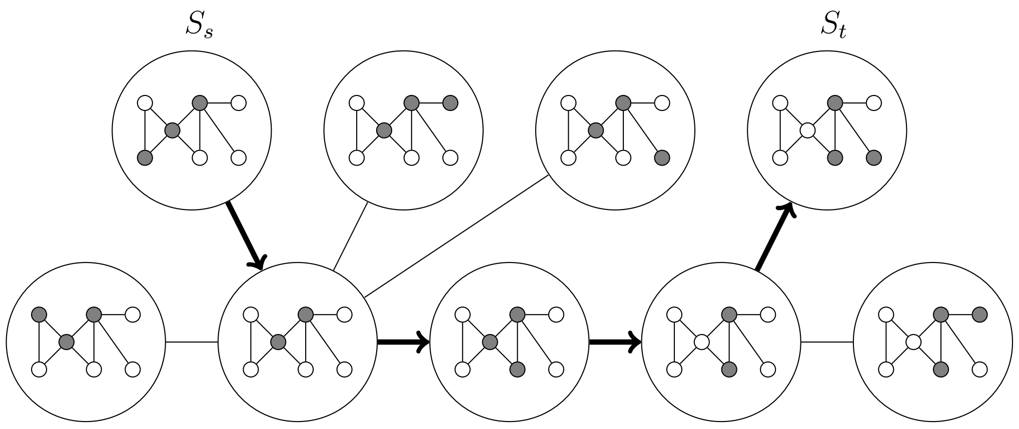


Figure 3.2: A reconfiguration sequence from S_s to S_t in $\mathcal{R}_{\mathcal{CVD}\text{-Min}}(G, 3)$ of length 4 (shown in thick lines) for a graph G.

defined as follows:

> **Input:** A graph G, two dominating sets S_s and S_t of G, and an integer threshold $k \geq \max\{|S_s|, |S_t|\}$.
> **Question:** Is there a path from S_s to S_t in $\mathcal{R}_{\mathcal{DS}\text{-Min}}(G, k)$?

Figure 3.3 shows the reconfiguration graph $\mathcal{R}_{\mathcal{DS}\text{-Min}}(G, k)$ for an input graph G consisting of an instance of DS and $k = 3$, where every feasible solution of size less than or equal to 3 is shown in solid circles.

Haas and Seyffarth [22] initiated the investigation of the reconfiguration of dominating sets. The main question they tried to answer was whether the reconfiguration graph $\mathcal{R}_{\mathcal{DS}\text{-Min}}(G, k)$ for a graph G and a non-negative integer k is connected. Hence, they tried to determine whether there exists a reconfiguration sequence between any two nodes of the reconfiguration graph. They first proved that for any graph G with at least two non-adjacent edges, $\mathcal{R}_{\mathcal{DS}\text{-Min}}(G, k)$ is connected if $k \geq \{|V(G)| - 1, \Lambda(G) + \lambda(G)\}$, where $\lambda(G)$ ($\Lambda(G)$), defined in Section 2.3, denotes the minimum (maximum) cardinality of any dominating set of a graph G. To prove this result, they first showed that $\mathcal{R}_{\mathcal{DS}\text{-Min}}(G, |V(G)| - 1)$ is connected if G is a graph with at least two non-adjacent edges. Then, they showed that $\mathcal{R}_{\mathcal{DS}\text{-Min}}(G, \Lambda(G))$ is not connected if G is a graph with at least one edge. Finally, they combined these results to prove the result stated above. Then, they restricted G to chordal

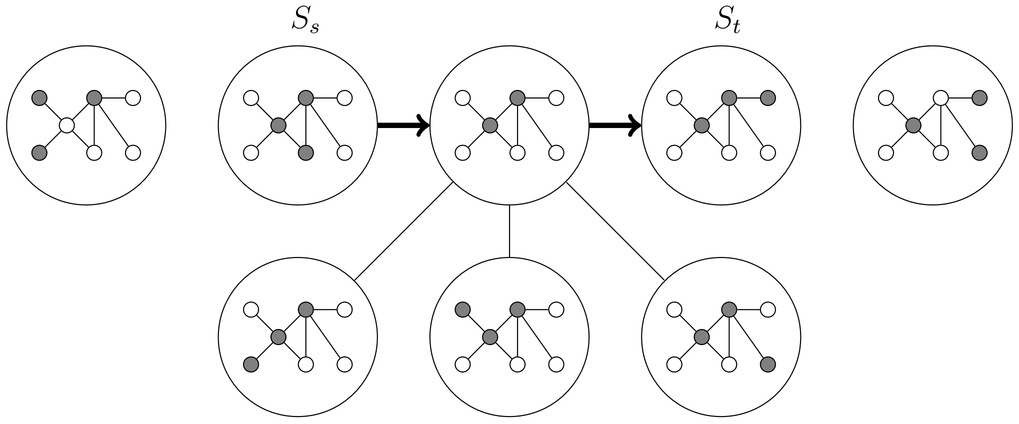


Figure 3.3: A reconfiguration sequence from S_s to S_t in $\mathcal{R}_{DS}(\mathcal{I}, 3)$ of length 4 (shown in thick lines).

and bipartite graphs and proved that $\mathcal{R}_{\mathcal{DS}\text{-Min}}(G, k)$ is connected whenever $k \geq \Lambda(G) + 1$ by using the properties of these graphs. They left as an open question whether the latter results could be extended to all graphs.

Recently, Suzuki et al. [48] improved these results and addressed the open question by Haas and Seyffarth to find an example of a graph G for which $\mathcal{R}_{\mathcal{DS}\text{-Min}}(G, \Lambda(G) + 1)$ is not connected. They improved the former results by showing that $\mathcal{R}_{\mathcal{DS}\text{-Min}}(G, |V(G)| - \mu)$ is connected if G has at least $\mu + 1$ non-adjacent edges, and its diameter is linear in $|V(G)|$. They showed this result by proving that there exists a reconfiguration sequence between any dominating set D of G such that $|D| \leq |V(G)| - \mu$, and the dominating set D' formed by removing an endpoint of each non-adjacent edges of G. Finally, they showed that there exists an infinite family of graphs such that if G belongs to this family, $\mathcal{R}_{\mathcal{DS}\text{-Min}}(G, \Lambda(G)+1)$ is not connected.

Haas and Seyffarth [22] and Suzuki et al. [48], as described above, have studied the connectivity problem for the Dominating Set. In this thesis, we build on this recent work by studying the st-connectivity problem $\mathcal{DS}$-Min-R for the Dominating Set problem when restricted to graph classes. We show the connection between the complexity of the extensively studied Dominating Set problem and its st-connectivity problem $\mathcal{DS}$-Min-R for various graph classes. We chose to study this problem on graph classes for which the complexity of Dominating Set is known and compare it to the complexity of its st-connectivity problem. We show that $\mathcal{DS}$-Min-R, similarly to its underlying decision problem [33], can be solved in time linear in the number of edges for paths, cographs, trees, and interval graphs. Furthermore, we show that the problem is PSPACE-complete for general graphs, bipartite graphs, and split graphs, whereas the underlying decision problem is NP-complete [3].

Chapter 4

Canonical Configuration

In this chapter, we introduce a scheme that will be used in the linear-time algorithms of Sections 6 and 7. In the remainder of this section, $\mathcal{P}$ represents either $\mathcal{CVD}$ or $\mathcal{DS}$.

Our idea is to use the concept of a "canonical" solution for a graph G, which is a well-defined node in the reconfiguration graph which is connected by a path to any other node in the reconfiguration graph. So, proving the existence of a canonical solution proves that there exists a path between any two feasible solutions of a reconfiguration graph. We say that a minimum feasible solution C for an instance $\mathcal{I}$ of $\mathcal{P}$-MIN-R is *universally reachable* if $S \leftrightsquigarrow C$ in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$ holds for every feasible solution S of G and $k = |S|+1$. To prove that there exists a path between any two feasible solutions of a reconfiguration graph, we have to show that there exists a universally reachable solution that is connected by a path to any other node in the reconfiguration graph, called a *canonical solution*. The canonical solution is a universally reachable node that is used to prove that all the universally reachable solutions are connected in the reconfiguration graph. We now state the following theorem that we prove using the results of Lemmas 4.1 and 4.2.

Theorem 1. *If an instance $\mathcal{I} = (G, S_s, S_t, k)$ of $\mathcal{P}$-MIN-R has a canonical solution for a graph G, then $\mathcal{P}$-MIN-R can be solved in time linear in $|E(G)|$.*

We note that problem $\mathcal{P}$-MIN-R is a decision problem asking for the existence of a reconfiguration sequence. Thus, we do not need to find a canonical solution in linear time. Hence, to solve problem $\mathcal{P}$-MIN-R, it suffices to prove the existence of a canonical solution. Before proving Theorem 1, we prove Lemmas 4.1 and 4.2 that together establish the main theorem.

Lemma 4.1. *Suppose an instance $\mathcal{I} = (G, S_s, S_t, k)$ of $\mathcal{P}$-Min-R has a canonical solution for a graph G, where $\mathcal{P}$ corresponds to either $\mathcal{CVD}$ or $\mathcal{DS}$. Then $\mathcal{I}$ is a* yes*-instance if $k \geq \max\{|S_s|, |S_t|\} + 1$.*

Proof. Let C be a canonical solution for $\mathcal{I}$. Then, there exists a reconfiguration sequence σ between S_s and C in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k')$ for $k' = |S_s| + 1$, by definition of a canonical solution. Suppose that $k \geq \max\{|S_s|, |S_t|\} + 1$. Since $k \geq |S_s| + 1 = k'$ and each set in σ is of cardinality at most $k \leq k'$, we clearly have $S_s \leftrightsquigarrow C$ in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$. Similarly, we have $S_t \leftrightsquigarrow C$ in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$. Since any reconfiguration sequence is reversible, we have $S_s \leftrightsquigarrow C \leftrightsquigarrow S_t$ in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$. Thus, this is a yes-instance. □

Lemma 4.1 implies that, if instance $\mathcal{I}$ has a canonical solution C, then it suffices to consider the case where $k = \max\{|S_s|, |S_t|\}$ since all instances are yes-instances if $k > \max\{|S_s|, |S_t|\}$. We will show in the following lemma that there is a no-instance (G, S_s, S_t, k) of $\mathcal{P}$-Min-R in such a case, and that it can be easily determined in time linear in $|E(G)|$.

Lemma 4.2. *Suppose an instance $\mathcal{I} = (G, S_s, S_t, k)$ of $\mathcal{P}$-Min-R has a canonical solution C for a graph G. Let S be an arbitrary feasible solution of $\mathcal{I}$ such that $S \neq C$, and let $k = |S|$. Then, $S \leftrightsquigarrow C$ holds if and only if S is not a minimal feasible solution.*

Proof. We first prove the if direction. Suppose that S is not minimal. Then, S contains at least one vertex x which is deletable from S, that is, $S \setminus \{x\}$ forms a feasible solution of G. Since C is canonical and S is not minimal, $\max\{|S \setminus \{x\}|, |C|\} = |S \setminus \{x\}|$. By Lemma 4.1, since $k = |S| = |S \setminus \{x\}| + 1 = \max\{|S \setminus \{x\}|, |C|\} + 1$, we have $S \setminus \{x\} \leftrightsquigarrow C$ in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$. Since $x \in S$ is a deletable vertex, there exists an edge between the nodes representing $S \setminus \{x\}$ and S in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$. Therefore, we have $S \leftrightsquigarrow S \setminus \{x\}$ and $S \setminus \{x\} \leftrightsquigarrow C$ so $S \leftrightsquigarrow C$ holds in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$.

We now prove the only-if direction by proving its contrapositive. Suppose that S is minimal. Then, S does not contain a deletable vertex. Hence, any feasible solution S' which is adjacent to S in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$ must be obtained by the addition of a vertex to S. Therefore, $|S'| = k + 1 > k = |S|$. Since $|S'| > k$, $S' \notin \mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$. Hence, S is an isolated node in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$. Therefore, $S \leftrightsquigarrow C$ in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$ does not hold. □

Corollary 4.1 can be immediately obtained from Lemma 4.2 as it shows that if an instance $\mathcal{I}$ of $\mathcal{P}$-Min-R has a canonical solution C for a graph G, then there exists a reconfiguration sequence $S \leftrightsquigarrow C$ in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$, where S is an arbitrary feasible solution and $|S| = k$, if and only if S is not minimal.

Corollary 4.1. *Suppose an instance $\mathcal{I} = (G, S_s, S_t, k)$ of $\mathcal{P}$-Min-R has a canonical solution for a graph G, $S_s \neq S_t$ and $k = \max\{|S_s|, |S_t|\}$. Then $\mathcal{I}$ is a* yes*-instance if and only if S_i is not minimal for every $i \in \{s, t\}$ such that $|S_i| = k$.*

Recall that throughout this section, $\mathcal{P}$ is either dominating set or cluster vertex deletion. We now show that we can determine in time linear in $|E(G)|$ whether or not a feasible solution S of an instance of $\mathcal{P}$ is minimal.

If $\mathcal{P}$ corresponds to $\mathcal{DS}$, first scan all vertices in $V(G) \setminus S$. For each vertex in $V(G) \setminus S$, count how many neighbours it has in S and mark it if it has exactly one. This takes $O(|N(v)|)$ time for one such vertex, and hence $O(|E(G)|)$ time total. Then, for each vertex $x \in S$, we check whether there exists a vertex $y \in N[x]$ which is only dominated by x. If such a vertex exists, we discard the vertices in $N(x) \setminus S$ from the graph. Otherwise, S is not minimal. Since a vertex $v \in (V(G) \setminus S)$ is discarded after visiting each vertex in $N(v)$ exactly once, each edge in $G[V(G) \setminus S]$ is visited at most once. Furthermore, a vertex $e = \{u, w\}$ in $G[S]$ is visited at most twice (it is visited when $v = u$ and when $v = w$). Therefore, we can determine in time linear in $|E(G)|$ whether or not a dominating set is minimal.

If $\mathcal{P}$ corresponds to $\mathcal{CVD}$, then by definition of a deletion set, every connected component of $G[V(G) \setminus S]$ forms a clique. Note that S is not minimal if there exists a connected component C of $G[V(G) \setminus S]$ and a vertex $x \in S$ such that $C \cup x$ forms a clique and x is not connected to any other connected component C' of $G[V(G) \setminus S]$. We first enumerate the vertices of each connected component of $G[V(G) \setminus S]$. Since each connected component is a clique, we can repeatedly pick a random vertex $y \in (V(G) \setminus S)$ that has not been visited yet, and enumerate the vertices of $N[y]$ that form a connected component. This can clearly be done in $|E(G)|$. Then, for each connected component C of $G[V(G) \setminus S]$, we check whether there exists a vertex in S which is adjacent to each vertex of C. This can be done by finding the intersection of the neighbourhoods of all the vertices of a connected component. This is similar to finding the intersection of the sets that represent the neighbourhood of each vertex, which can be done in time linear in $|E(G)|$ [14]. Finally, among these sets that represent vertices that are adjacent to each vertex of a component of $G[V(G) \setminus S]$, we count the number of occurrences of each vertex in all the sets. If there exists a vertex that occurs exactly once, then S is not minimal. Otherwise, S is minimal. Therefore, we can determine in time linear in $|E(G)|$ whether or not a deletion set is minimal.

Lemma 4.3. *For an instance $\mathcal{I} = (G, S_s, S_t, k)$ of $\mathcal{P}$-Min-R, we can determine in time linear in $|E(G)|$ whether or not a feasible solution S of $\mathcal{I}$ is minimal.*

We note again that Lemmas 4.1 and 4.2 imply that only the existence of a canonical solution is required to solve the problem. Furthermore, by Corollary 4.3, we can determine if a feasible solution S is minimal in time linear in $|E(G)|$.

We now prove Theorem 1, using the results obtained by the above lemmas.

Proof. If an instance $\mathcal{I} = (G, S_s, S_t, k)$ of $\mathcal{P}$-Min-R has a canonical solution C for a graph G, then by Lemma 4.1, $\mathcal{I}$ is a yes-instance if $k > \max\{|S_s|, |S_t|\}$. By Fact 2.3.1, $\mathcal{I}$ is a yes-instance if $S_s = S_t$ and $k \geq \lambda(G)$. By Corollary 4.1, if $k = \max\{|S_s|, |S_t|\}$ and $S_s \neq S_t$, $\mathcal{I}$ is a yes-instance if and only if S_i is not minimal for every $i \in \{s, t\}$ such that $|S_i| = k$. By Corollary 4.3, we can determine if a feasible solution S is minimal in time linear in $|E(G)|$. By Proposition 2.3.1, $\mathcal{I}$ is a no-instance if $k < \lambda(G)$. So if $\mathcal{I}$ has a canonical solution, then $\mathcal{P}$-Min-R can be solved in time linear in $|E(G)|$. $\square$

The following lemma shows that when constructing a canonical dominating set for an instance of $\mathcal{DS}$-Min-R, it suffices to construct a canonical dominating set for a connected graph.

Lemma 4.4. *Let $\mathcal{I} = (G, S_s, S_t, k)$ be an instance of $\mathcal{DS}$-Min-R where G is a graph consisting of p connected components $G_1, G_2, \ldots, G_p$. Let S be any dominating set of G and suppose that C_i is a canonical dominating set in $\mathcal{R}_{\mathcal{DS}\text{-MIN}}(G_i, k_i)$ for $k_i = |S \cap V(G_i)|$. Then, $C = C_1 \cup C_2 \cup \cdots \cup C_p$ is a canonical dominating set of $\mathcal{I}$.*

Proof. Since S is a dominating set of G and G_i is a connected component of G for each $i \in \{1, 2, \ldots, p\}$, $S \cap V(G_i)$ is a dominating set of G_i. Furthermore, since C_i is a canonical solution in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G_i, k_i)$, we have $S \cap V(G_i) \leftrightsquigarrow C_i$ in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G_i, k_i + 1)$. Therefore, there exists a reconfiguration sequence between $S \cap V(G_i)$ and C_i in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G_i, k_i + 1)$ for each $i \in \{1, 2, \ldots, p\}$.

To prove that C is a canonical dominating set of $\mathcal{I}$, we have to show that $S \leftrightsquigarrow C$ in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$ holds, where $k = |S| + 1$. First, we construct a sequence σ of sets between S and $C = C_1 \cup C_2 \cup \cdots \cup C_p$. Then, we show that each set in σ is a dominating set of G and of cardinality at most $|S| + 1$.

For each $i \in \{1, 2, \ldots, p\}$, we let $S_0 = S$ and construct a subsequence σ_i that transforms S_{i-1} into $S_i = S_{i-1} \setminus ((S_{i-1} \cap V(G_i)) \cup C_i)$ by only transforming the set $S_{i-1} \cap V(G_i)$ into C_i using the sets in the reconfiguration sequence from $S_{i-1} \cap V(G_i)$ to C_i in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G_i, k_i + 1)$. Note that σ_i transforms S_{i-1} into S_i by only adding or removing vertices in $V(G_i)$. Since σ_i corresponds to a reconfiguration sequence in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G_i, k_i + 1)$ and $S_0 = S$ where S is a dominating set of G, each set in σ_i is a dominating set of G. Note that $S = (S \cap V(G_1)) \cup (S \cap V(G_2)) \cup \ldots \cup (S \cap V(G_p))$ and $k_i = |S \cap V(G_i)|$ for each $i \in \{1, 2, \ldots, p\}$. Therefore,

$k_1+k_2+\ldots+k_p = |S|$. Since a subsequence σ_i only adds and removes vertices in $V(G_i)$, the size of each set of σ_i in $V(G)\setminus V(G_i)$ is equal to $|S|-k_i$. Furthermore, since C_i is a canonical solution in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G_i, k_i)$, each set in σ_i is of size at most $|S|-k_i+(k_i+1) = |S|+1$. By definition of a canonical dominating set, C_i is a minimum dominating set of G_i. Therefore, we have $|S_{i-1}\cap V(G_i)| \geq |C_i|$ for each $i \in \{1,2,\ldots,p\}$. Thus, we have $|S_{i-1}| \geq |S_i|$. Since σ is a sequence from $S_0 = S$ to $S_p = C$ and $|S_{i-1}| \geq |S_i|$, each set in σ is of cardinality at most $|S|+1$. Therefore, σ is a reconfiguration sequence from S to C in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G,k)$ where $k = |S|+1$. Consequently, C is a canonical dominating set of $\mathcal{I}$.

□

We will now introduce some notation of a characterization of a representative set $S \subseteq V(G)$, for some rooted tree G, that we will use in Sections 6.1.1, 6.1.2, 7.1.1, and 7.1.2. Suppose that the vertices in S are ordered as $w_1, w_2, \ldots, w_{|S|}$ by a post-order depth-first traversal of the tree starting from the root r of G. For each $i \in \{1,2,\ldots,|S|\}$, we denote by G_i the subtree of G which is induced by w_i and all its descendants in G.

Observation 4.1. *Only the root w_i of G_i is adjacent with a vertex in $V(G)\setminus V(G_i)$.*

Then, for each $i \in \{1,2,\ldots,|S|\}$, we define a vertex subset C_i of $V(G)$, as follows:

$$C_i = \begin{cases} V(G_i)\setminus \bigcup_{j<i} V(G_j) & \text{if } i \neq |S|; \\ V(G)\setminus \bigcup_{j<i} V(G_j) & \text{if } i = |S|. \end{cases}$$

Note that since $V(G_i) = C_1 \cup C_2 \cup \ldots \cup C_i$, $\{C_1, C_2, \ldots, C_{|S|}\}$ forms a partition of $V(G)$. Figures 6.1, 6.2, 7.1, and 7.2 illustrate such a partition of a tree G. Furthermore, notice that

$$S \cap C_i = \{w_i\} \tag{4.1}$$

holds for every $i \in \{1,2,\ldots,|S|\}$.

We will refer to this notation as the *canonical representation of S for G*. We now prove the following lemma that we will use in Sections 6.1.1 and 6.1.2 for an instance of $\mathcal{CVD}$-Min-R on trees, and in Sections 7.1.1 and 7.1.2 for an instance of $\mathcal{DS}$-Min-R on trees.

Lemma 4.5. *Suppose an instance $\mathcal{I} = (G, S_s, S_t, k)$ of $\mathcal{P}$-Min-R, where $\mathcal{P}$ corresponds to either $\mathcal{CVD}$ or $\mathcal{DS}$, has a minimum feasible solution S for a tree G. Furthermore, using the canonical representation of S for tree G, suppose that for every feasible solution T of $\mathcal{I}$ for G, $|T \cap C_i| \geq 1$ holds for every $i \in \{1,2,\ldots,|S|\}$. Also, suppose that $S \cap V(G_i)$ is a feasible solution of G_i. Then, S is a canonical solution of $\mathcal{I}$.*

Proof. By the definition of canonical solution, we have to show that $T \leftrightsquigarrow S$ in $\mathcal{R}_{\mathcal{P}\text{-MIN}}$ (G, k) holds, where $k = |T| + 1$. First, we construct a sequence $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_{|S|} \rangle$ of sets between T and S, where each σ_i represents a subsequence of σ. Then, we prove that each set in σ is a feasible solution of $\mathcal{I}$ when $\mathcal{I}$ is an instance of $\mathcal{CVD}$-MIN-R or $\mathcal{DS}$-MIN-R. Finally, we prove that each set in σ is of cardinality at most $|T| + 1$.

We let $T_0 = T$ be the first set of the sequence σ. For each i from 1 to $|S|$, we set $h = 1$ and we construct a subsequence $\sigma_i = \langle T_i^1, T_i^2, \ldots, T_i^{l_i} \rangle$ of σ in the following phases, for some positive integer l_i, where each set in σ_i is formed after each addition or removal of a vertex, and $T_i = T_i^{l_i}$:

(1) If $w_i \notin T_{i-1}$, add vertex w_i, let T_i^1 be the resulting set, and set $h = 2$.

(2) For $j \in \{h, \ldots, l_i\}$, remove a vertex $c \in T_i^{j-1} \cap \big(C_i \setminus \{w_i\}\big)$ and let T_i^j be the resulting set.

Note that for every j, T_i^j in σ_i is obtained from T_i^{j-1} by the addition or removal of a single vertex. By Equation 4.1, $S \cap C_i = \{w_i\}$ and by construction of T_i, we have $T_i \cap C_i = \{w_i\}$ for every $i \in \{1, 2, \ldots, |S|\}$. Therefore, $T_i \cap C_i = S \cap C_i$. Since $V(G_i) = C_1 \cup C_2 \cup \ldots \cup C_i$ and $T_i \cap C_i = S \cap C_i$ for every $i \in \{1, 2, \ldots, |S|\}$, we have

$$T_i \cap V(G_i) = S \cap V(G_i). \tag{4.2}$$

Furthermore, since $T_i \cap V(G_i) = S \cap V(G_i)$ and $S \cap V(G_i)$ is a feasible solution of G_i, $T_i \cap V(G_i)$ is a feasible solution of G_i. Let U denote the set $V(G) \setminus V(G_i)$. Note that by construction of T_i,

$$T_{i-1} \cap U = T_i \cap U. \tag{4.3}$$

We now show that each set in σ is a feasible solution of $\mathcal{I}$ when $\mathcal{I}$ is an instance of $\mathcal{CVD}$-MIN-R and when it is an instance of $\mathcal{DS}$-MIN-R. For both instance cases, we first show that each set T_i forms a feasible solution of G. Then, we show that each set in σ is a feasible solution of G.

Case 1: Suppose that $\mathcal{I}$ is an instance of $\mathcal{CVD}$-MIN-R. First, using induction, we show that each set T_i forms a deletion set of G. Then, we show that each set in $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_{|S|} \rangle$ is a deletion set of G by showing that each set of σ_i is a deletion set of G.

We now show that each set T_i forms a deletion set of G. Since $T_0 = T$ and T is deletion set of G, the base case holds. Suppose that T_{i-1} is a deletion set of G. By Fact 2.2.1, since T_{i-1} is a deletion set of G, $T_{i-1} \cap V(G_{i-1})$ is a deletion set of G_{i-1}. Similarly, $T_{i-1} \cap U$ is a deletion set of $G[U]$. Since $T_{i-1} \cap U = T_i \cap U$ and $T_{i-1} \cap U$ is a deletion set of $G[U]$, $T_i \cap U$ is a deletion set of $G[U]$. By Equation 4.2, we have $T_i \cap V(G_i) = S \cap V(G_i)$. By the statement of Lemma 4.5, $S \cap V(G_i)$ is a deletion set of G_i. Since $T_i \cap V(G_i) = S \cap V(G_i)$ and $S \cap V(G_i)$ is a deletion set of G_i, $T_i \cap V(G_i)$ is a deletion set of G_i. Therefore, since $T_i \cap V(G_i)$ is a

deletion set of G_i and $T_i \cap U$ is a deletion set of $G[U]$, where $U = V(G) \setminus V(G_i)$, T_i is a deletion set of G. Since both the base case and the induction step hold, by induction, each set T_i is a deletion set of G, for each $i \in \{1, 2, \ldots, |S|\}$.
Case 2: Suppose that $\mathcal{I}$ is an instance of $\mathcal{DS}$-Min-R. Similarly to Case 1, we use induction to show that each set T_i forms a dominating set of G. Then, we show that each set in σ is a dominating set of G by showing that each set of σ_i is a dominating set of G.

We now show that each set T_i forms a dominating set of G. Suppose that T_{i-1} is a dominating set of G. By Equation 4.2, $T_i \cap V(G_i) = S \cap V(G_i)$. By the statement of Lemma 4.5, $S \cap V(G_i)$ is a dominating set of G_i. Therefore, $T_i \cap V(G_i)$ is a dominating set of G_i. By construction of T_i, the only vertices added or removed in σ_i belong to C_i. Thus, each vertex removed in Phase (2) is dominated by a vertex in $T_i \cap V(G_i)$. Furthermore, since by Observation 4.1 w_i is the only vertex in G_i that is adjacent with a vertex in $V(G) \setminus V(G_i)$, any vertex in $T_{i-1} \cap U$ that is dominated by a vertex in C_i is dominated by w_i. Hence, since T_{i-1} is a dominating set and, by Equation 4.3, $T_{i-1} \cap U = T_i \cap U$, each vertex in $T_i \cap U$ is dominated by a vertex in T_i. Therefore, T_i is a dominating set of G. Since both the base case and the induction step hold, by induction, each set T_i is a dominating set of G, for each $i \in \{1, 2, \ldots, |S|\}$.

We now show that, for both cases, each set of σ is a feasible solution of G by showing that each set of σ_i is a feasible solution of G. By Observation 2.2.1, all the sets formed in Phases (1) and (2) are feasible solutions since each of them is a superset of T_i. Therefore, since each set in the subsequence σ_i forms a feasible solution of G, each set in σ is a feasible solution of G.

Finally, we prove that each set in the sequence σ from T to S in $\mathcal{R}_{\mathcal{P}\text{-Min}}(G, k)$ is of cardinality at most $|T| + 1$. We first show that $|T_{i-1}| \geq |T_i|$ for each $i \in \{1, 2, \ldots, |S|\}$ and that the maximum cardinality of any set in σ_i is $|T_{i-1}| + 1$. Then, since $|T_0| = |T|$ and $|T_{i-1}| \geq |T_i|$, the cardinality of any set in the sequence from T to S in $\mathcal{R}_{\mathcal{P}\text{-Min}}(G, k)$ is at most $|T| + 1$.

We now show that $|T_{i-1}| \geq |T_i|$ for the cases where $w_i \in T_{i-1}$ and $w_i \notin T_{i-1}$. Then, we conclude that the cardinality of any set in σ is at most $|T| + 1$. If $w_i \in T_{i-1}$, then no vertex is added in Phase (1), hence each set in σ_i is of cardinality at most $|T_{i-1}|$ because we only delete vertices in Phase (2). Therefore, $|T_{i-1}| \geq |T_i|$ holds. We thus consider the case where $w_i \notin T_{i-1}$. By the statement of Lemma 4.5, for any feasible solution T of $\mathcal{I}$, $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$. Therefore, since T_{i-1} is a feasible solution of $\mathcal{I}$, $|T_{i-1} \cap C_i| \geq 1$ holds. Since $w_i \notin T_{i-1}$ and $|T_{i-1} \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$, we have $T_{i-1} \cap \big(C_i \setminus \{w_i\}\big) \neq \emptyset$. Therefore, all the sets obtained in Phase (2) are of cardinality at most $|T_{i-1}|$. Thus, $|T_{i-1}| \geq |T_i|$ holds. Since T_{i-1} is a feasible solution and there is only a single vertex addition in the subsequence σ_i, each set in σ_i is of cardinality at most $|T_{i-1}| + 1$. Therefore, σ is a reconfiguration sequence from T to S in $\mathcal{R}_{\mathcal{P}\text{-MIN}}(G, k)$ such that all intermediate feasible solutions are of cardinality at most $|T| + 1$.

We have thus proved that for the minimum feasible solution S for G, $T \leftrightsquigarrow S$ holds in $\mathcal{R}_{\mathcal{DS}\text{-MIN}}(G, k)$ for every feasible solution T of $\mathcal{I}$ for G and $k = |T| + 1$. Therefore, S forms a canonical solution of $\mathcal{I}$. $\square$

Chapter 5

Clique Reconfiguration

In this chapter, we study the $\mathcal{CL}$-MAX-R problem. The $\mathcal{CL}$-MAX-R problem was proven to be PSPACE-complete on general graphs by Ito et al. [29]. We show in the following sections that $\mathcal{CL}$-MAX-R can be solved in time linear in $|E(G)|$ for paths, trees, bipartite graphs, chordal graphs, and cographs. Recall that these results were found independently from the work by Ito et al. [32]. We refer the reader to that work for polynomial-time algorithms for graphs of bounded clique size and graphs having polynomially many cliques.

5.1 Paths

In this subsection, we show that $\mathcal{CL}$-MAX-R is solvable in time linear in $|E(G)|$ for any path G. Recall that, from Section 2.3, $\Lambda(G)$ represents the maximum cardinality of any feasible solution of a graph G.

Fact 5.1.1. *Given any path G, the cardinality of a maximum clique is $\Lambda(G) = 2$.*

For an instance (G, S_s, S_t, k) of $\mathcal{CL}$-MAX-R, we will show that either we have a trivial no-instance or we can find a reconfiguration sequence from S_s to S_t in time linear in $|E(G)|$.

Theorem 2. *For any instance (G, S_s, S_t, k) of $\mathcal{CL}$-MAX-R where G is a path, $\mathcal{CL}$-MAX-R can be in solved time linear in $|E(G)|$.*

Proof. We will first check whether this is a no-instance by checking if any of the conditions from Proposition 2.3.2 hold, in which case we report a no-instance. Otherwise, we will

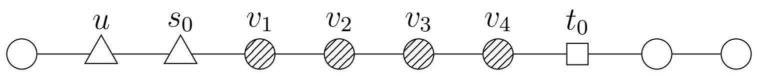


Figure 5.1: An example of a path with $S_s = \{u, s_0\}$ and $S_t = \{t_0\}$.

show that we have a yes-instance.

To prove that we have a yes-instance, we show that when none of the conditions of Proposition 2.3.2 hold, there is a reconfiguration sequence from S_s to S_t in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$. By Fact 5.1.1, the cardinality of a maximum clique is 2. Therefore, if $S_s = S_t$ and $k \leq 2$, then by Fact 2.3.2, we report a yes-instance. Since by Proposition 2.3.2 we have a no-instance when $k > 2$ or when $k = 2$ and $S_s \neq S_t$, we only have to show that $\mathcal{I}$ is a yes-instance for $S_s \neq S_t$ and $k \leq 1$.

By Proposition 2.3.3, if $k = 0$ then we report a yes-instance. So the case remaining is when $k = 1$. Let s_0 and t_0 be two vertices in G such that $s_0 \in S_s$ and $t_0 \in S_t$. We construct a reconfiguration sequence from $S_s^{'} = \{s_0\}$ to $S_t^{'} = \{s_t\}$ such that all intermediate cliques are of cardinalities at most 2. Then, we show that $S_s \leftrightsquigarrow S_s^{'}$ and $S_t^{'} \leftrightsquigarrow S_t$ hold in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, 1)$, and therefore $S_s \leftrightsquigarrow S_s^{'} \leftrightsquigarrow S_t^{'} \leftrightsquigarrow S_t$ holds.

We start by constructing the path $G' = (v_0, v_1, \ldots, v_l)$, for some positive integer l, of vertices between s_0 and t_0 by using depth first search in G, such that $v_0 = s_0$ and $v_l = t_0$ [2]. Then, we construct the reconfiguration sequence between $S_s^{'}$ and $S_t^{'}$ in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, 1)$. For $i \in \{1, 2, \ldots, l\}$, we construct the reconfiguration sequence as follows:

(1) Add vertex v_i.

(2) Remove vertex v_{i-1}.

Clearly, all the sets formed in steps 1 and 2 are cliques as they are either a single vertex or two vertices forming an edge. Since $|S_s'| = 1$ and the sets in the reconfiguration sequence from S_s' to S_t' are obtained by an alternation of a removal and an addition of a single vertex, each set formed is a clique of size at most two. Figure 5.1 illustrates a path G such that triangles are vertices in $S_s = \{u, s_0\}$, squares are vertices in $S_t = \{t_0\}$, striped nodes are vertices in the path from s_0 to s_t, and white circles are every other vertex in $V(G)$.

Note that if $|S_s| = 2$, then $S_s' = S_s \setminus u$ where $u \in S_s$ and $u \neq s_0$. Similarly, if $|S_t| = 2$, then $S_t = S_t' \setminus w$ where $w \in S_t$ and $w \neq t_0$. Therefore, $S_s \leftrightsquigarrow S_s'$ and $S_t' \leftrightsquigarrow S_t$ holds. Hence, there exists a reconfiguration sequence $S_s \leftrightsquigarrow S_t$ in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, 1)$.

The depth first search algorithm finds path G in time linear in $|V(G)|$. Since steps 1 and 2 take constant time and we go through path G only once, our algorithm is linear in $|E(G)|$.

□

5.2 Trees and Bipartite Graphs

In this subsection, we show that $\mathcal{CL}$-MAX-R is solvable in time linear in $|E(G)|$ for any tree or bipartite graph G.

Fact 5.2.1. *Given any tree or bipartite graph G, the cardinality of a maximum clique is $\Lambda(G) = 2$.*

Similarly to paths, for an instance (G, S_s, S_t, k) of $\mathcal{CL}$-MAX-R, we will show that either we have a trivial no-instance or $S_S \leftrightsquigarrow S_t$ in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$.

Theorem 3. *For any instance (G, S_s, S_t, k) of $\mathcal{CL}$-MAX-R where G is a tree or a bipartite graph, $\mathcal{CL}$-MAX-R can be solved in time linear in $|E(G)|$.*

Proof. Similarly to paths, we first check whether this is a no-instance by checking if any of the conditions from Proposition 2.3.2 hold, in which case we report a no-instance. Otherwise, we try to find a path in G that contains a vertex $s \in S_s$ and a vertex $t \in S_t$. If there does not exist such a path, we report a no-instance. Otherwise, we will show that we have a yes-instance. Note that such a path always exists if G is a tree.

When none of the conditions of Proposition 2.3.2 hold, to prove that we have a yes-instance or a no-instance, we will implement a modified breadth-first search algorithm to find if there exists a path P such that $s, t \in V(P)$, where $s \in S_s$ and $t \in S_t$. If G is a bipartite graph and there does not exist such a path, we will show that it is a no-instance. Otherwise, we will show that there is a reconfiguration sequence from S_s to S_t in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$. By Fact 5.2.1, the cardinality of a maximum clique is 2. Therefore, if $S_s = S_t$ and $k \leq 2$, then by Fact 2.3.2, we report a yes-instance. Since by Proposition 2.3.2 we have a no-instance when $k > 2$ or when $k = 2$ and $S_s \neq S_t$, we only have to show that $\mathcal{I}$ is a yes-instance for $S_s \neq S_t$ and $k \leq 1$.

We now show how to find whether there exists a path $P \subseteq G$ such that $s, t \in V(P)$, where $s \in S_s$ and $t \in S_t$.

By Proposition 2.3.3, if $k = 0$ then we report a yes-instance. We find if there exists a path P of vertices between a vertex $s \in S_s$ and a vertex $t \in S_t$ in G by implementing a modified breadth-first search algorithm as follows: We start a breadth-first search at vertex s. For every vertex v we discover, we mark it as visited and store its predecessor. We do not revisit a vertex that has already been visited and we stop the search after vertex t has been visited or after there are no more vertices that can be visited. If G is a bipartite graph and vertex t was not visited after the breadth-first search, then there does not exist a path between s and t and we report a no-instance. Otherwise, if vertex t was visited after the breadth-first search, we can reconstruct the path in reverse order by starting at vertex t and visiting the predecessor of every vertex in the path until we reach vertex s. We now have a path P of G that begins at s and ends at t. Similarly as in the proof of Theorem 2, we can now find a reconfiguration sequence from S_s to S_t by first removing the vertex in $S_s \setminus \{s\}$ (if there exists such a vertex), then reconfiguring the clique from $\{s\}$ to $\{t\}$ along path P as in the proof of Theorem 2, and then adding the vertex in $S_t \setminus \{t\}$ (if there exists such a vertex). This algorithms clearly runs in time linear in $|E(G)|$. Therefore, we can find a reconfiguration sequence between S_s and S_t in time linear in $|E(G)|$. □

Observation 5.2.1. *Theorem 3 holds for all graphs G with clique-size at most 2.*

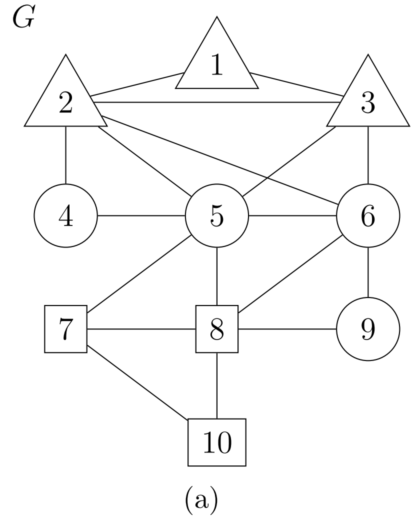
5.3 Chordal Graphs

By Section 2.1, we can compute a clique tree of a chordal graph G in time linear in $|E(G)|$ and we know that a graph is a chordal graph if and only if it has a clique tree. Figure 5.2 (a) illustrates a chordal graph G such that triangles are vertices in $S_s = \{1, 2, 3\}$ and squares are vertices in $S_t = \{7, 8, 10\}$, and (b) its clique tree T where $S_s = \{1, 2, 3\}$ and $S_t = \{7, 8, 10\}$. For an instance $\mathcal{I} = (G, S_s, S_t, k)$ of $\mathcal{CL}$-MAX-R where G is a chordal graph, we will show that $\mathcal{CL}$-MAX-R can be solved in time linear in $|E(G)|$. Note that for a clique tree $\mathcal{T}$ of G, a node $M \in \mathcal{T}$ represents a set of vertices of G such that M is a maximal clique of G. Therefore, there exists nodes $A \in \mathcal{T}$ and $B \in \mathcal{T}$ such that $A \supseteq S_s$ and $B \supseteq S_t$. We start by checking whether any conditions of Proposition 2.3.2 hold, in which case we report a no-instance. Then, we find a path $\mathcal{P} = \langle M_0, M_1, \ldots, M_l \rangle$ in the clique tree $\mathcal{T}$ from a node $M_0 \supseteq S_s$ to a node $M_l \supseteq S_t$. The path $\mathcal{P}$ from M_0 to M_t represented by thick edges is shown in Figure 5.2 (b). Finally, we show that $\mathcal{I}$ is a no-instance if there exists a clique $S_j = M_{j-1} \cap M_j$ such that $k > |S_j|$, for any $j \in \{1, 2, \ldots, l\}$. Otherwise, we show that $\mathcal{I}$ is a yes-instance.

Theorem 4. *For any instance $\mathcal{I} = (G, S_s, S_t, k)$ of $\mathcal{CL}$-MAX-R where G is a chordal graph, $\mathcal{CL}$-MAX-R can be solved in time linear in $|E(G)|$.*

Proof. We start by checking whether any conditions of Proposition 2.3.2 hold, in which case we report a no-instance. Otherwise, we will find a maximal clique $M_0 \in \mathcal{T}$ such that $S_s \subseteq M_0$ and a maximal clique $M_t \in \mathcal{T}$ such that $S_t \subseteq M_t$, for a clique tree $\mathcal{T}$ of G.

Let $C \in \{S_s, S_t\}$. To find a maximal clique M such that $C \subseteq M$, we first set $L = V(G) \setminus C$ where L are potential clique vertices. Then, starting with clique C, we iterate over L's vertices, adding a vertex $c \in L$ to the current clique if c is connected to



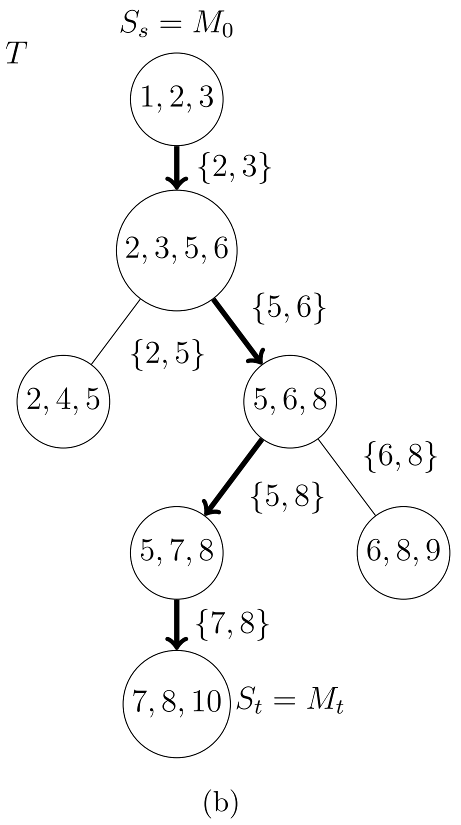


Figure 5.2: (a) A chordal graph G, (b) a clique tree T of G.

each vertex in the current clique, and set $L = L \setminus \{c\}$ otherwise. Note that for each vertex $v \in L$, we go through the vertices $N_G(v)$ checking whether the current clique is a subset of $N_G(v)$, in which case we add it to the current clique, otherwise we discard it from L. Therefore, we go through each edge of G at most once. This algorithm is linear in $|E(G)|$.

Using this algorithm, we find maximal cliques A and B in G such that $S_s \subseteq A$ and $S_t \subseteq B$ in time linear in $|E(G)|$. Then, we find the nodes $M_0 = A$ and $M_t = B$ in the clique tree $\mathcal{T}$ using the depth-first search algorithm on trees that runs in time linear in $|V(\mathcal{T})|$ [2].

Then, we construct the path $\mathcal{P} = (M_0, M_1, \ldots, M_l)$, where $M_l = M_t$ for some positive integer l, of maximal cliques in $\mathcal{T}$ between M_0 and M_t using the depth-first search algorithm on trees that runs in time linear in $|V(\mathcal{T})|$ [2]. Note that $\mathcal{P}$ is the unique path between M_0 and M_t since $\mathcal{T}$ is a tree. Let $C_j = M_{j-1} \cap M_j$ for each $j \in \{1, 2, \ldots, l\}$, and assume that $|C_j|$ is given for each edge of the clique tree.

Then, we will show that $\mathcal{I}$ is a no-instance if there exists a clique C_j such that $|C_j| < k$ for any $j \in \{1, 2, \ldots, l\}$, and a yes-instance otherwise. Note that there exists a reconfiguration sequence S between M_{j-1} and M_j in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$ only if $|M_{j-1} \cap M_j| \geq k$ since the clique $M_{j-1} \cap M_j$ must be in S. Note that $M_{j-1} \cap M_j = C_j$. Furthermore, as stated in Section 2.1, since $\mathcal{T}$ is a maximum spanning tree of the intersection graph I of G, the weight of the minimum-weight edge in $\mathcal{P}$ is the maximum among the minimum-weight edges of all possible paths between M_0 and M_t in I. Hence, if there exists a clique C_j in the unique path $\mathcal{P}$ between M_0 and M_t such that $|C_j| < k$ for any $j \in \{1, 2, \ldots, l\}$, $\mathcal{I}$ is a no-instance. Note that in Figure 5.2, all the edges in the path between S_s and S_t have the same weight of 2. If there is no such C_j, then $C_{j-1} \leftrightsquigarrow C_j$ holds in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$ because $C_{j-1} \cup C_j \subseteq M_{j-1}$ and hence $C_{j-1} \cup C_j$ forms a clique of G for each $j \in \{1, 2, \ldots, l\}$. Thus, we have $C_0 \leftrightsquigarrow C_l$ in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$. Since $C_s \subseteq M_0$ and $C_0 \subseteq M_0$ and $C_s \cup C_0 \subseteq M_0$, $C_s \leftrightsquigarrow C_0$ holds in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$. Similarly, since $C_t \subseteq M_t$ and $C_l \subseteq M_t$ and $C_t \cup C_l \subseteq M_t$, $C_l \leftrightsquigarrow C_t$ holds in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$. Therefore, $C_s \leftrightsquigarrow C_t$ holds in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$.

□

5.4 Cographs

In this section, we show that the $\mathcal{CL}$-Max-R problem can be solved in time linear in $|E(G)|$ where G is a cograph.

By Proposition 2.2.1, we know that $S \subseteq G$ is a clique in G if and only if S is an independent set in $\overline{G}$. Using this fact, we will prove the following lemma.

Lemma 5.4.1. *Let C_i be a clique of a graph G for all $i \in \{1, 2, \ldots, l\}$, where l is some positive integer. A sequence $\langle C_1, C_2, \ldots, C_l \rangle$ is a reconfiguration sequence of cliques in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$ if and only if $\langle C_1, C_2, \ldots, C_l \rangle$ is a reconfiguration sequence of independent sets in $\mathcal{R}_{\mathcal{IS}\text{-MAX}}(\overline{G}, k)$.*

Proof. We first prove the if direction. Suppose that $\langle C_1, C_2, \ldots, C_l \rangle$ is a reconfiguration sequence of independent sets in $\mathcal{R}_{\mathcal{IS}\text{-MAX}}(\overline{G}, k)$. Then, by Proposition 2.2.1, each C_i is a clique in G, for each $i \in \{1, 2, \ldots, l\}$. Therefore, a reconfiguration step between the independent sets C_{i-1} and C_i in $\mathcal{R}_{\mathcal{IS}\text{-MAX}}(\overline{G}, k)$ is a reconfiguration step between the cliques C_{i-1} and C_i in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$. Hence, $\langle C_1, C_2, \ldots, C_l \rangle$ is a reconfiguration sequence of cliques in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$. We then prove the only-if direction. Suppose that

$\langle C_1, C_2, \ldots, C_l \rangle$ is a reconfiguration sequence of cliques in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$. Then, by Proposition 2.2.1, each C_i is an independent set in $\overline{G}$, for each $i \in \{1, 2, \ldots, l\}$. Thus, a reconfiguration step between the cliques C_{i-1} and C_i in $\mathcal{R}_{\mathcal{CL}\text{-MAX}}(G, k)$ is a reconfiguration step between the independent sets C_{i-1} and C_i in $\mathcal{R}_{\mathcal{IS}\text{-MAX}}(\overline{G}, k)$ Therefore, $\langle C_1, C_2, \ldots, C_l \rangle$ is a reconfiguration sequence of independent sets in $\mathcal{R}_{\mathcal{IS}\text{-MAX}}(\overline{G}, k)$. □

We now prove the following theorem using the results of Lemma 5.4.1.

Theorem 5. *For a cograph G, $\mathcal{CL}$-Max-R can be solved in time linear in $|E(G)|$.*

Proof. By the definition of cographs, the class of cographs is closed under taking complements [12]. By Lemma 5.4.1 and by the fact that $\mathcal{IS}$-Max-R can be solved in time linear in $|E(G)|$ for a cograph G [34], $\mathcal{CL}$-Max-R is solvable in time linear in $|E(G)|$. □

Chapter 6

Cluster Vertex Deletion Reconfiguration

In this chapter, we study the $\mathcal{CVD}$-MIN-R problem. We show in the following sections that $\mathcal{CVD}$-MIN-R can be solved in time linear in the number of edges for paths and trees. Furthermore, we prove that the problem is NP-hard on bipartite graphs. Finally, we prove that it is PSPACE-complete in general.

6.1 Linear-Time Algorithms

In this section, we show that $\mathcal{CVD}$-MIN-R is solvable in time linear in $|E(G)|$ for paths and trees. These proofs are very similar in spirit to the proofs that $\mathcal{DS}$-MIN-R is solvable in linear time on paths and trees, given in the PhD thesis by Amer E. Mouawad [38].

6.1.1 Paths

In this section, we show that $\mathcal{CVD}$-MIN-R is solvable in time linear in $|V(G)|$ for any path G. By Theorem 1 it suffices to prove that an instance of $\mathcal{CVD}$-MIN-R has a canonical deletion set for a path G. To do so, we construct a canonical deletion set for a path G. By Proposition 5.1.1, the cardinality of a maximum clique in G is 2. Therefore, the cluster graph formed after the removal of a deletion set from G is composed of disjoint edges and isolated vertices. We assume that $V(G) = \{1, 2, \ldots, n\}$ for some positive integer n. We denote the two 1 degree vertices by r and s, where r denotes the root of G and s the leaf of

G in a tree corresponding to the path. We start by labelling each vertex of G with one of 1, 2 and 3, from vertex s to vertex r in the following pattern: $1, 2, 3, 1, 2, 3$ and so on. We denote by S the set of all vertices in G that are labelled with 3. Figure 6.1 (a) illustrates a path labelled in the described pattern.

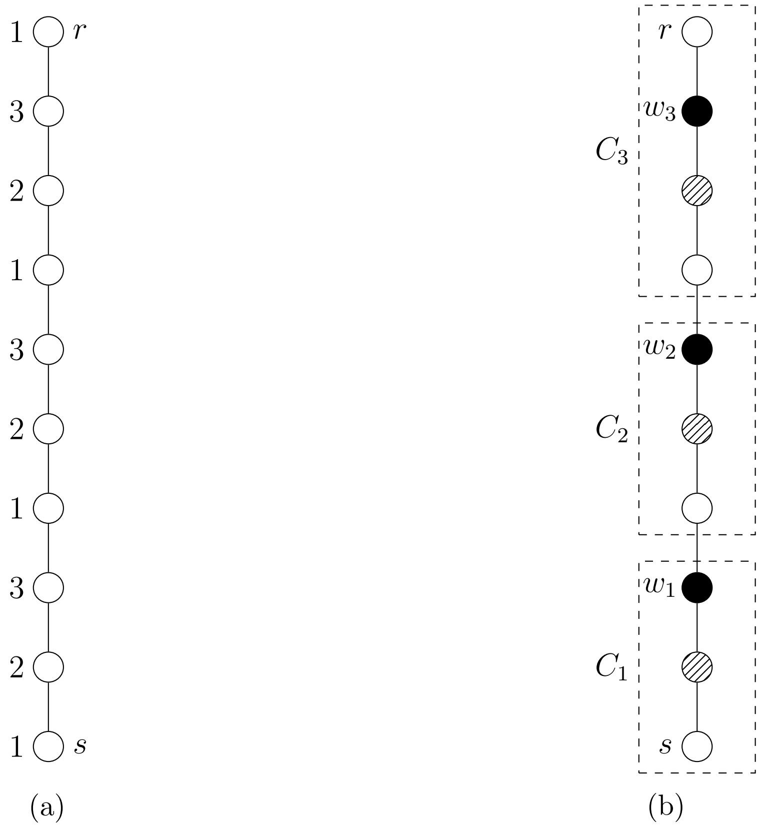


Figure 6.1: (a) The labelling of a path P, (b) the partition of $V(P)$ into C_1, C_2, C_3.

We prove the following theorem by showing that S forms a canonical deletion set of an instance of $\mathcal{CVD}$-Min-R for G.

Theorem 6. *For an instance $\mathcal{I} = (G, S_s, S_t, k)$ of $\mathcal{CVD}$-Min-R for a path G, $\mathcal{CVD}$-Min-R can be in solved in time linear in $|V(G)|$.*

Proof. By Theorem 1, it suffices to prove that S forms a canonical deletion set of $\mathcal{I}$ for path G. By Lemma 4.5, to prove that S forms a canonical deletion set of $\mathcal{I}$, using the canonical representation of S for G, we have to show that S forms a minimum deletion set of G such that for every deletion set T of G, $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$, and that $S \cap V(G_i)$ is a deletion set of G_i. First, we prove that S is a deletion set of G. Afterwards, we show that $S \cap V(G_i)$ is a deletion set of G_i. Then, we prove that $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$. Finally, we prove that S is a minimum deletion set of G.

Using the canonical representation of S for G, we show that S is a deletion set of G. It suffices to show that for every $i \in \{1, 2, \ldots, |S|\}$, each $C_i \backslash w_i$ is a cluster graph. Figure 6.1 illustrates the canonical representation of a path P partitioned into subsets C_1, C_2, C_3 where solid circles are vertices in S, striped circles are vertices labelled with 2, and white circles are vertices labelled with 1. In C_i, w_i has as child a vertex labelled with 2. Vertices labelled with 2, by construction, each have a child labelled with 1 and vertices labelled with 1 do not have children in C_i. So after the removal of w_i from C_i, the edge connecting the vertices labelled with 1 and 2 will form a connected component, hence forming a clique. Therefore, for every $i \in \{1, 2, \ldots, |S|\}$, each $C_i \backslash w_i$ is a cluster graph, so S is a deletion set of G.

Now, we prove that $S \cap V(G_i)$ is a deletion set of G_i. Recall that $V(G_i) = C_1 \cup C_2 \cup \ldots \cup C_i$. By definition of a cluster graph, a union of cluster graphs is a cluster graph. Therefore, since $C_i \backslash w_i$ is a cluster graph for every $i \in \{1, 2, \ldots, |S|\}$, $V(G_i) \setminus S$ is a cluster graph. Thus, $S \cap V(G_i)$ is a deletion set of G_i.

We now show that for an arbitrary deletion set T of G, $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$. Suppose instead that $T \cap C_i = \emptyset$ holds for some index $i \in \{1, 2, \ldots, |S|\}$. We prove that C_i never forms a cluster graph. Then, since $T \cap C_i = \emptyset$ and C_i is not a cluster graph, T is not a deletion set of G; this contradicts the assumption that T is a deletion set of G.

By construction, w_i can only have as child a vertex labelled 2 and a vertex in C_i labelled with 2 has a child labelled with 1. The path formed by the vertices labelled 1, 2, and w_i creates an induced path of length three. Therefore, C_i is not a cluster graph. Hence, we have proved that for an arbitrary deletion set T of G, $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$.

Finally, we prove that S is a minimum deletion set of G. Note that $C_i \backslash \{w_i\}$ is composed of an edge forming a connected component of cardinality 2. By Proposition 5.1.1, the cardinality of a maximum clique in G is 2. Therefore, since C_i, after the removal of w_i, forms a maximum clique and $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$, S forms a minimum deletion set of G.

By Lemma 4.5, since S is a minimum deletion set of G such that for an arbitrary deletion set T of G, $|T \cap C_i| \geq 1$ for every $i \in \{1, 2, \ldots, |S|\}$, and $S \cap V(G_i)$ is a deletion set of G_i, S is a canonical deletion set of $\mathcal{I}$.

□

6.1.2 Trees

In this section, we show that $\mathcal{CVD}$-Min-R is solvable in time linear in $|V(G)|$ for any tree T. Similarly to the case of paths, by Theorem 1 it suffices to prove that an instance of $\mathcal{CVD}$-Min-R has a canonical deletion set for a tree G. To do so, we construct a canonical deletion set for a tree G.

We choose an arbitrary vertex r of degree one in G, and regard G as a rooted tree whose root is r. We first label each vertex in G with one of 1, 2 and 3, from the leaves of G to the root r of G, as in the following steps (1)–(3); intuitively, the vertices labelled with 3 form a deletion set of G, each vertex u labelled with 1 will form a clique by being isolated or by forming an edge with a vertex v labelled with 2:

(1) All leaves in G are labelled with 1.

(2) Pick an internal vertex v of G such that all children of v have already been labelled. Then,
  - label v with 1 if all children of v are labelled with 3;
  - label v with 2 if v has exactly one child labelled 1 and none labelled 2; and
  - otherwise label v with 3.

For each $i \in \{1, 2\}$, we denote by V_i the set of all vertices in G that are labelled with i, and by S the set of all vertices labelled with 3 . Then, $\{V_1, V_2, S\}$ forms a partition of $V(G)$. Figure 6.2 (a) illustrates a tree whose vertices are labelled following the above steps.

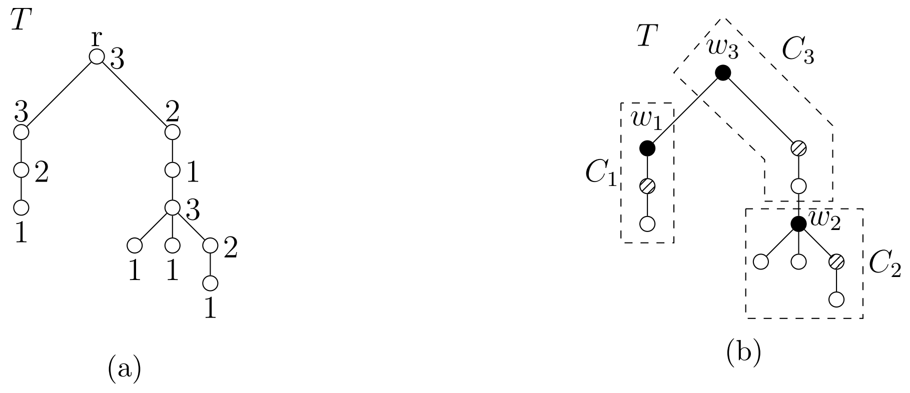


Figure 6.2: (a) The labelling of a tree T, (b) the partition of $V(T)$ into C_1, C_2, C_3.

Theorem 7. *For an instance $\mathcal{I} = (G, S_s, S_t, k)$ of $\mathcal{CVD}$-Min-R for a tree G, $\mathcal{CVD}$-Min-R can be in solved in time linear in $|V(G)|$.*

Proof. By Theorem 1, it suffices to prove that S forms a canonical deletion set of $\mathcal{I}$ for tree G. By Lemma 4.5, to prove that S forms a canonical deletion set of $\mathcal{I}$, we have to use the canonical representation of S for G to show that S forms a minimum deletion set of G such that for every deletion set T of G, $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$, and that $S \cap V(G_i)$ is a deletion set of G_i. First, we prove that S is a deletion set of G. Then, we show that $S \cap V(G_i)$ is a deletion set of G_i. Subsequently, we prove that $|T \cap C_i| \geq 1$ holds. Finally, we prove that S is a minimum deletion set of G.

Using the canonical representation of S for G, we show that S is a deletion set of G. Figure 6.2 illustrates the canonical representation of S for a tree T partitioned into subsets C_1, C_2, C_3, where solid circles are vertices in S, striped circles are vertices in V_2, and white circles are vertices in V_1. It suffices to show that for every $i \in \{1, 2, \ldots, |S|\}$, each $C_i \backslash w_i$ is a cluster graph. In C_i, w_i can have as children vertices labelled with 1 or 2. Note that there is exactly one vertex labelled 3 in each C_i so if a vertex $v \in C_i$ has as child a vertex u labelled with 3, $u \notin C_i$. Vertices labelled with 1 do not have children in C_i because a vertex is labelled with 1 in G if it is a leaf or if all of its children are labelled 3. The removal of w_i from C_i will leave the vertices labelled with 1 isolated, hence each will form a clique. Vertices labelled with 2 by definition must have exactly one child labelled with 1 and can as well have children labelled with 3. Similarly, the vertices labelled with 3 do not belong to C_i and the vertices labelled with 1 do not have children in C_i. So after the removal of w_i from C_i, the edge connecting the vertices labelled with 1 and 2 will form a connected component, hence forming a clique. Therefore for every $i \in \{1, 2, \ldots, |S|\}$, each $C_i \backslash w_i$ is a cluster graph, so S is a deletion set of G.

Now, we prove that $S \cap V(G_i)$ is a deletion set of G_i. By definition of a cluster graph, a union of cluster graphs is a cluster graph. Therefore, since $C_i \backslash w_i$ is a cluster graph for every $i \in \{1, 2, \ldots, |S|\}$ and $V(G_i) = C_1 \cup C_2 \cup \ldots \cup C_i$, $V(G_i) \setminus S$ is a cluster graph. Thus, $S \cap V(G_i)$ is a deletion set of G_i.

Now, we show that for an arbitrary deletion set T of G, $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$. Suppose for a contradiction that $T \cap C_i = \emptyset$ holds for some index $i \in \{1, 2, \ldots, |S|\}$. We prove that C_i never forms a cluster graph. Then, since $T \cap C_i = \emptyset$ and C_i is not a cluster graph, T is not a deletion set of G; this contradicts the fact that T is a deletion set of G.

We first consider the case where w_i has at least two children and then the case where it has exactly one child. First, consider the case where w_i has at least two children in C_i. Then, by definition of a tree, these children do not have any edges between them. Since the path between any two children of w_i that goes through w_i creates an induced path of length three, C_i does not form a cluster graph. This contradicts the assumption that T is a deletion set of G. Finally, consider the case where w_i has exactly one child which, by construction, can only be a vertex labelled 2. By construction, a vertex in C_i labelled with 2 must have at least one child, one of which must be labelled with 1. Then, the path formed by the vertices labelled 1, 2, and w_i creates an induced path of length three. Similarly, this is a contradiction since T is a deletion set of G. Hence, we have proved that for an arbitrary deletion set T of G, $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \dots, |S|\}$.

Finally, we show that S is a minimum deletion set of G. By Equation 4.1, we know that $S \cap C_i = \{w_i\}$ and $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \dots, |S|\}$ where T is an arbitrary deletion set of G. Also, we know that $\{C_1,\ C_2\ , \dots\ , C_{|S|}\}$ forms a partition of $V(G)$. Suppose as a contradiction that there exists a deletion set T' with $|T'| < |S|$. Since $S \cap C_i = \{w_i\}$ holds for every $i \in \{1, 2, \dots, |S|\}$ and $|T'| < |S|$, there exists a C_i such that $T' \cap C_i = \emptyset$. This contradicts the fact that $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \dots, |S|\}$. Hence, S is a minimum deletion set of G.

By Lemma 4.5, since S is a minimum deletion set of tree G such that for an arbitrary deletion set T of G, $|T \cap C_i| \geq 1$ for every $i \in \{1, 2, \dots, |S|\}$, and $S \cap V(G_i)$ is a deletion set of G_i, S is a canonical deletion set of $\mathcal{I}$.

□

6.2 NP-hardness

In this section, we show that $\mathcal{CVD}$-MIN-R is NP-hard for bipartite graphs.

We define the notion of *iterative compression* [21] whose central idea will be used in proving our result. Iterative compression employs a so-called *compression routine*. Given a problem instance and a corresponding feasible solution, a compression routine either calculates a smaller feasible solution or proves that the given feasible solution is of minimum size. Using a compression routine, one finds an optimal feasible solution to a problem by inductively building up the problem structure and iteratively compressing intermediate feasible solutions. In the application of iterative compression of the CLUSTER VERTEX DELETION (CVD) problem, the compression routine first exhaustively considers all possible sets that correspond to the intersection of the given solution and a potentially smaller solution, and discards the elements in these sets from the problem instance. Then, it tries to find a smaller solution which is disjoint from the given solution. Unlike this compression routine, to prove the main result of this section, we only need to find a solution which is smaller than the given solution. Therefore, we define the compression routine of the CVD problem, called the CLUSTER VERTEX DELETION COMPRESSION ($CVDC$) problem, as follows:

Input: An undirected graph G and a deletion set D of G.
Question: Is there a vertex set $D' \subseteq V(G)$ such that D' is a deletion set and $|D'| < |D|$?

Another problem that will be used in proving our result is the NP-complete MAXIMUM INDUCED MATCHING problem [47] defined as follows:

Input: An undirected graph $G = (V, E)$ and a non-negative integer k.
Question: Is there an induced matching $M \subseteq E$ with $|M| \geq k$?

First, we will show in Theorem 8 that the CVD problem is NP-complete on bipartite graphs by a polynomial-reduction from MAXIMUM INDUCED MATCHING on bipartite graphs. Then, we prove in Theorem 9 that the $CVDC$ problem on bipartite graphs is NP-complete by using a Turing reduction from CVD on bipartite graphs. Finally, we prove in Theorem 10 that $\mathcal{CVD}$-MIN-R is NP-hard by a reduction from $CVDC$ on bipartite graphs.

Theorem 8. *The* CLUSTER VERTEX DELETION *problem is* NP*-complete on bipartite graphs.*

Proof. Clearly, CLUSTER VERTEX DELETION is in NP. Given a deletion set D and a non-negative integer k, one can verify in polynomial time if that is a deletion set of size at most k. This can be done by checking whether $|D| \leq k$ and by removing the deletion set, and taking each connected component and checking if it forms a clique.

To show that it is NP-complete, we give a polynomial-time reduction from the NP-complete MAXIMUM INDUCED MATCHING problem on bipartite graphs [47]. Let (G, k) be an instance of MAXIMUM INDUCED MATCHING for a bipartite graph G where $V(G) = \{v_1, v_2, \ldots, v_n\}$ and $E(G) = \{e_1, e_2, \ldots, e_m\}$. Then, $V(G)$ can be partitioned in polynomial time into two subsets A and B such that $G[A]$ and $G[B]$ are edgeless [37]. We construct the corresponding bipartite graph G' as follows. Let I be the set of isolated vertices of G and $V(G') = V(G) \setminus I$. For G' the resulting graph, G' is bipartite. Let $|V(G')| = n'$ and $(G', k' = n' - 2k)$ be the corresponding instance of CLUSTER VERTEX DELETION for bipartite graphs. Clearly, this instance can be constructed in polynomial time. We will prove that G has an induced matching M of size at least k if and only if G' has a deletion set D of size at most k'.

If M is an induced matching of G of size at least k, we will show that there exists a deletion set of size at most k' in G'. M is an induced matching, so by definition of an induced matching, no two distinct edges of M are joined by an edge of G. If M is an induced matching of G, then $D = V(G') \setminus V(M)$ is a deletion set since any component of $G[V(G') \setminus D]$ is an edge of M, hence a clique. The induced matching M is of size at least k which implies that $|V(M)| \geq 2k$, and since $|V(G')| = n'$, $|D| = |V(G')| - |V(M)| \leq n' - 2k$. So if G has an induced matching M of size at least k, then G' has a deletion set of size at most k'.

If G' has a deletion set D of size at most k', we will show that there exists a matching of size at least k in G. Since $G' \setminus D$ is a bipartite graph, there exists a partition (A, B) of its vertex set such that $G[A]$ and $G[B]$ are edgeless. Since the bipartite graph G' does not have any isolated vertices and D is a deletion set, $G' \setminus D$ is a bipartite cluster graph. By Fact 2.1.1, the cluster graph $G' \setminus D$ contains no induced path of three vertices. Furthermore, by Fact 5.2.1, the bipartite graph $G' \setminus D$ has cliques of maximum cardinality 2. Therefore, since $G' \setminus D$ does not have any isolated vertices, the only connected components that form cliques are edges. We know that $|V(G')| = n'$ and $|V(D)| \leq k'$ where $k' = n' - 2k$ so $|V(G')| - |V(D)| \geq 2k$. So we can conclude that $G' \setminus D$ is formed of at least k edges no two of which are joined by an edge of G', which is by definition an induced matching. By construction, $V(G') = V(G) \cup I$ so an induced matching of size at least k in G' yields an induced matching of size at least k in G. So if G' has a deletion set D of size at most k', then G has a matching of size at least k. □

We will now use the result of Theorem 8 to prove that $CVDC$ on bipartite graphs is NP-complete. One can easily show that an efficient algorithm for $CVDC$ implies an efficient algorithm for CVD, and hence NP-hardness of CVD implies NP-hardness of $CVDC$ (see the paper by Fedor V. Fomin et al. [16] for a similar reduction for vertex cover). We give the details here for completeness.

Theorem 9. *The $CVDC$ problem is* NP*-complete on bipartite graphs.*

Proof. To show that the $CVDC$ problem is NP-complete on bipartite graphs, we give a Turing reduction from CVD on bipartite graphs. We will prove that if we have an algorithm A that can solve the $CVDC$ problem, we can solve the CVD problem using a polynomial number of calls to algorithm A.

Let G be the graph for which we are trying to find if there exists a deletion set of size at most k. We will solve the problem by using algorithm A which uses iterative compression. We will start with $D = V(G)$; clearly D is a deletion set. We then give graph G and the deletion set D as the input to A, and let D' be the deletion set that A outputs after applying its compression routine. If $|D'| \geq k$, then we repeatedly try to find a smaller deletion set for G by giving graph G and the deletion set $D = D'$ as input to algorithm

A. At every iteration, A either returns a smaller deletion set for G, or proves that D is optimal. If it is optimal, then we can conclude that G does not have a deletion set of size at most k. Since eventually $D = \emptyset$, we obtain a solution for G once the algorithm A returns a deletion set D' with $|D'| \leq k$. Since the algorithm A either returns a smaller solution or proves that the given solution is of minimum size, we will call A at most $|V|$ times. □

Finally, we can use the result of Theorem 9 to prove that $\mathcal{CVD}$-Min-R is NP-hard by a reduction from $CVDC$ on bipartite graphs.

Theorem 10. *The $\mathcal{CVD}$-Min-R problem is NP-hard on bipartite graphs.*

Proof. We will show that the problem is NP-hard on bipartite graphs by a polynomial-time reduction from $CVDC$ on bipartite graphs.

Let (G, D) be an instance of $CVDC$ for a bipartite graph G where $V(G) = \{v_1, v_2, \ldots, v_n\}$, $E(G) = \{e_1, e_2, \ldots, e_m\}$ and D is a deletion set of size k. Figure 6.3 (a) illustrates a bipartite graph G and a deletion set of size 4 represented by solid circles. We construct a bipartite graph G' as follows. We let $V(G') = A \cup B$, where $A = \{a_1, a_2, \ldots, a_n\}$ and $\{a_i, a_j\} \in E(G')$ if $\{v_i, v_j\} \in E(G)$, and B is a biclique with partitions U and W. We let $U = \{u_1, u_2, \ldots, u_{k+1}\}$ and $W = \{w_1, w_2, \ldots, w_{k+1}\}$. Since B is a biclique, for every two vertices $u \in U$ and $w \in W$, uw is an edge in $E(G')$. The resulting graph G' is bipartite. Let $D_s = D \cup U$, $D_t = D \cup W$, and $k' = 3k$. Clearly, $|D_s| = |D_t| = 2k + 1$. Figure 6.3 (b) illustrates a graph G' where D_s is represented by triangle shaped vertices and D_t is represented by striped vertices. Let (G', D_s, D_t, k') be the corresponding instance of $\mathcal{CVD}$-Min-R for bipartite graphs. Clearly, this instance can be constructed in polynomial

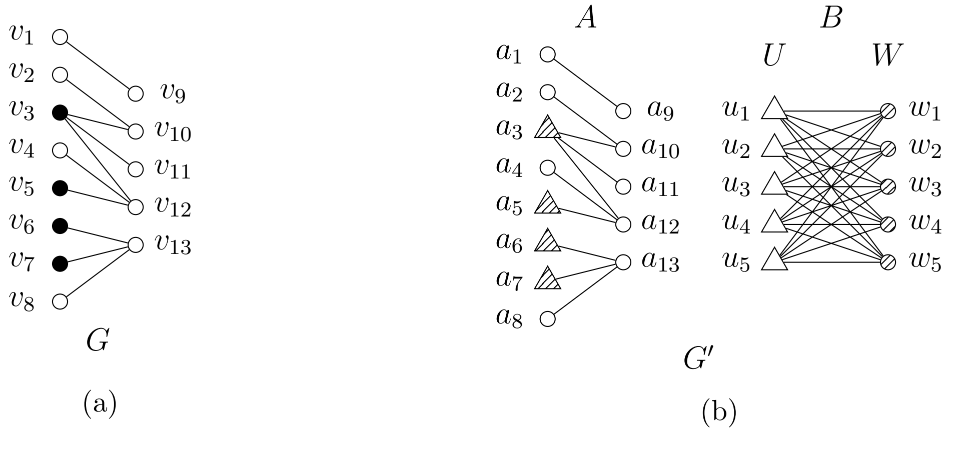


Figure 6.3: (a) A graph G, and (b) a graph G'.

time. Thus, we will prove that $D_s \leftrightsquigarrow D_t$ in $\mathcal{R}_{CVD\text{-}\textsc{Min}}(G', k')$ if and only if there is a deletion set $D' \subseteq V(G)$ and $|D'| < |D|$.

We first prove the if direction. Suppose there is a deletion set $D' \subseteq V(G)$ such that $|D'| < |D|$. Since $|D| = k$ and $|D'| \leq |D| - 1$, then $|D'| \leq k - 1$. We will form a reconfiguration sequence of deletion sets in $\mathcal{R}_{CVD\text{-}\textsc{Min}}(G', k')$ between D_s and D_t. We will construct the reconfiguration sequence σ in 8 stages, and at each stage, we will show that the sets formed are deletion sets of size at most k'. Note that by Observation 2.2.1, all sets formed in the stages where we add vertices to our deletion set are deletion sets. Therefore, we will only prove that the sets formed are deletion sets in stages where we delete vertices from our deletion set. Also, by Observation 2.2.1, to prove that all the sets formed in the stages where we remove vertices are deletion sets, it suffices to show that our resulting set is a deletion set. We form the reconfiguration sequence between D_s and D_t as follows:

(1) Add each vertex of D'. The resulting deletion set size is $2k+1+|D'| \leq 2k+1+k-1 \leq 3k = k'$.
(2) Remove each vertex from D. The resulting set is $U \cup D'$ which is a deletion set of G' since D' is a deletion set of A and U is a deletion set of B. The resulting deletion set size is $|U| + |D'| \leq (k+1) + (k-1) = 2k < k'$.
(3) Add k arbitrary vertices of W. The resulting deletion set size is $|U| + |D'| + k \leq 3k = k'$.
(4) Remove one vertex $u \in U$. The resulting set is a deletion set since the vertex $w \in W$ that is not in our deletion set and u form a cluster graph as they form an edge. The resulting deletion set size is $k' - 1$.
(5) Add vertex $w \in W$ where w is the unique vertex in W that is not in our deletion set. The resulting deletion set size is k'.
(6) Remove each vertex from U. The resulting set is $W \cup D'$ which is a deletion set of G' since D' is a deletion set of A and W is a deletion set of B. The resulting deletion set size is $|W| + |D'| \leq 2k < k'$.
(7) Add each vertex of D. The resulting deletion set size is $|D'| + |W| + |D| \leq 3k = k'$.
(8) Remove each vertex from D'. The resulting set is $D \cup W$ which is a deletion set of G' since D is a deletion set of A and W is a deletion set of B. The resulting deletion set size is $|D| + |W| \leq 2k + 1 < k'$.

Thus by stages (1)-(8), there exists a reconfiguration sequence $D_s \leftrightsquigarrow D_t$ in $\mathcal{R}_{CVD\text{-}\textsc{Min}}$ (G', k'). Consider the instance in Figure 6.3 (b) where $k' = 12$, $D = \{a_3, a_5, a_6, a_7\}$, and $D' = \{a_{10}, a_{12}, a_{13}\}$, the reconfiguration between D_s and D_t in $\mathcal{R}_{CVD\text{-}\textsc{Min}}(G', 12)$ as described above holds as each set formed is a deletion set of size at most $k' = 12$.

We now prove the only-if direction by proving its contrapositive. Suppose that there does not exist a deletion set $D' \in V(G)$ such that $|D'| < |D|$. Suppose for contradiction that there exists a reconfiguration sequence σ between D_s and D_t in $\mathcal{R}_{CVD\text{-MIN}}(G', k')$. Since there does not exist a deletion set $D' \in V(G)$ such that $|D'| < |D|$, D is a minimum deletion set of G and A, and $\lambda(A) = |D| = k$. Since B is a biclique, there exists an induced path (x, y, z) of size three, where $x, z \in U$ and $y \in W$. Similarly, there exists an induced path (a, b, c) of size three, where $a, c \in W$ and $b \in U$. Furthermore, since $|U| = |W| = k + 1$, the size of a minimum deletion set of B is $k + 1$, and $\lambda(B) = k + 1$. Note that all vertices of U must be removed to form D_t. Let N be the first deletion set in σ such that a vertex $u \in U$ was removed, and M the deletion set from which N was obtained through the removal of u. Note that M contains all $k + 1$ vertices of U and at least k vertices of A since M is a deletion set of G'. Similarly, N contains exactly k vertices of U and at least k vertices of A. The deletion set N must also contain at least k vertices of W; otherwise two vertices $v_1, v_2 \in W$ and u would form the induced path $\{v_1, u, v_2\}$ of length three. Therefore, N contains exactly k vertices of U, at least k vertices of A and at least k vertices of W. Hence, $|N| \geq 3k = k'$, so N must contain exactly k vertices of U, k vertices of A and k vertices of W. Note that M is obtained from N by the removal of vertex u, hence $M = N \cup \{u\}$. Therefore, $|M| = |N| + 1 = k' + 1 > k'$. Therefore, there does not exist a reconfiguration step from M to N, a contradiction.

□

6.3 PSPACE-completeness

In this section, we prove that $\mathcal{CVD}$-MIN-R is PSPACE-complete for general graphs. Amer E. Mouawad [38] proved in a similar way that $\mathcal{DS}$-MIN-R is PSPACE-complete. Before stating Theorem 11, we will define the H-WORD RECONFIGURATION problem [49]. Given a pair $H = (\Sigma, R)$, where Σ is an alphabet and $R \subseteq \Sigma^2$ a binary relation between symbols, we say that a word w over Σ is an *H-word* if every two consecutive symbols of w are in the relation R. If one looks at H as a digraph (possibly with loops), where every symbol in the alphabet is represented by a vertex and there is an edge (u, v) if $(u, v) \in R$, a word w is an H-word if and only for two consecutive symbols (x, y) in w, $(x, y) \in E(H)$. The H-WORD RECONFIGURATION problem asks whether two given H-words w_s and w_t of equal length n can be transformed into one another by changing one symbol at a time so that all intermediary steps are also H-words of length n. The H-WORD RECONFIGURATION is know to be PSPACE-complete for general graphs [41].

Theorem 11. *$\mathcal{CVD}$-MIN-R is PSPACE-complete for general graphs.*

Proof. One can observe that the problem is in PSPACE [29, Theorem 1]. We thus show that it is PSPACE-hard for general graphs by a polynomial-time reduction from H-WORD RECONFIGURATION.

Let (H, w_s, w_t) be an instance of H-WORD RECONFIGURATION, where $H = (\Sigma, R)$, $\Sigma = \{s_1, s_2, \ldots, s_m\}$, $w_s = a_1a_2\ldots a_n$ and $w_t = c_1c_2\ldots c_n$. We construct an instance G of $\mathcal{CVD}$-MIN-R as follows. We first make n gadgets G_i for all $1 \leq i \leq n$, where each gadget G_i will be used to represent the i^{th} character of an H-word. Each G_i is a split graph with $V(G_i) = C_i \cup S_i$, where C_i is a clique of size $2m$ and $S_i = \{s^i_1, \ldots, s^i_m\}$ an independent set of size m; more precisely, each vertex $s^i_j \in S_i$ corresponds to symbol $s_j \in \Sigma$. We join all vertices in C_i to each vertex in S_i such that $E(G_i) = \{\{c_j, s^i_l\} \mid c_j \in V(C_i), s^i_l \in V(S_i), 1 \leq j \leq 2m, 1 \leq l \leq m\}$. Let $V(G) = V(G_1) \cup V(G_2) \cup \ldots \cup V(G_n)$ and in addition, for each binary relation $(s_a, s_b) \notin R$, add an edge between s^l_a and s^{l+1}_b for all $l = 1, \ldots, n-1$. We let $D_s = L_1 \cup L_2 \cup \ldots \cup L_n$, where $L_i = S_i \setminus \{a_i\}$ for all $1 \leq i \leq n$. Similarly, we let $D_t = U_1 \cup U_2 \cup \ldots \cup U_n$, where $U_i = S_i \setminus \{c_i\}$ for all $1 \leq i \leq n$. The construction of G for $\Sigma = \{a, b, c\}$, $R = \Sigma^2 \setminus \{(a,a), (b,a), (c,b), (c,c)\}$, $w_s = abc$, $w_t = bca$, and $n = 3$ where each vertex in S_i that corresponds to symbol $a \in \Sigma$ is represented by a triangle, each vertex in S_i that corresponds to symbol $b \in \Sigma$ is represented by a square, each vertex in S_i that corresponds to symbol $c \in \Sigma$ is represented by a circle, and vertices of D_s are represented by dashed nodes and vertices of D_t are represented by grey filled nodes, as illustrated in Figure 6.4. Let G be the resulting graph, and let $(G, D_s, D_t, k = n(m-1)+1)$ be the corresponding instance of $\mathcal{CVD}$-MIN-R. Clearly, this instance can be constructed in polynomial time.

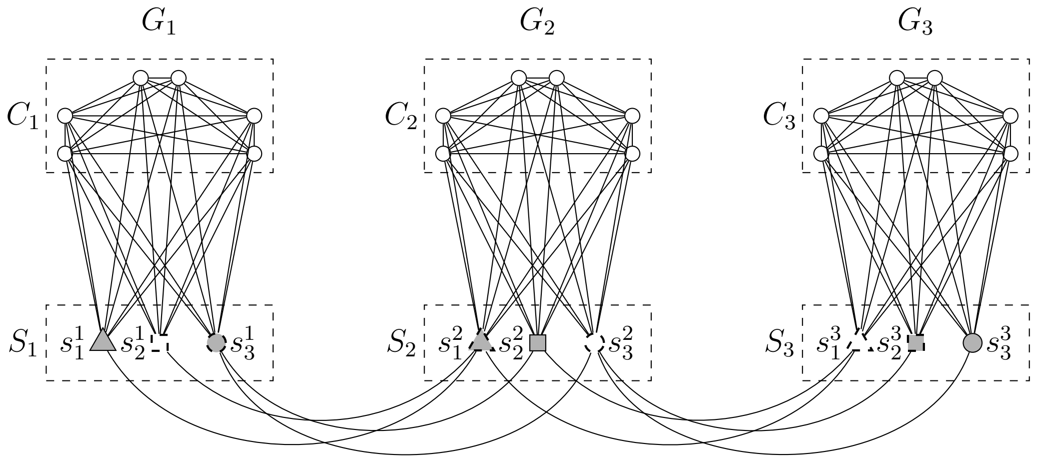


Figure 6.4: A graph G.

The construction that allows us to transform an H-word w of H to a deletion set D of G is as follows: let $w = b_1 b_2 \ldots b_n$. We know that $V(G) = V(G_1) \cup V(G_2) \cup \ldots \cup V(G_n)$ where each G_j is a split graph with an independent set S_j and a clique C_j for all $1 \leq j \leq n$. Note that every vertex in each S_j corresponds to a symbol in the alphabet Σ. For $u \in S_j$, it is easy to see that $S_j \setminus \{u\}$ is a deletion set of G_j since, by construction, $C_j \cup u$ forms a cluster graph. Let $D = D_1 \cup D_2 \cup \ldots \cup D_n$, where $D_i = S_i \setminus \{b_i\}$ for all $1 \leq i \leq n$.

We will now prove that D is a deletion set of G by showing that $G[V(G) \setminus D]$ is a cluster graph. Note that each D_i is a deletion set of G_i. If we look at $G[V(G) \setminus D]$, then we notice that $G[V(G) \setminus D] = (C_1 \cup \{b_1\}) \cup (C_2 \cup \{b_2\}) \cup \ldots \cup (C_n \cup \{b_n\})$. We know by definition of an H-word that every two consecutive symbols of an H-word are in the relation R. Hence, by construction, there are no edges between b_i and b_{i+1} in $G[V(G) \setminus D]$ for all $1 \leq i \leq n$. Thus, each $G[C_i \cup \{b_i\}]$ is a connected component that forms a clique, which makes $G[V(G) \setminus D]$ a cluster graph. Note that $|D_i| = m - 1$ for all i, which makes D a deletion set of size $n(m-1)$.

We will now prove that the size of a minimum deletion set of G is $n(m-1)$. We will first show that the size of a minimum deletion set of D_i for any $i \in \{1, 2, \ldots, n\}$ is $m-1$. Since G_i is a split graph, there exists an induced path (x, y, z) of size three, where $x, z \in S_i$ and $y \in C_i$. We can either remove all the vertices of C_i and get a deletion set of size $2m$, or remove $m-1$ vertices of S_i and get a deletion set of size $m-1$. Hence the size of a minimum deletion set of each D_i is $m-1$. By construction, $E(G) \setminus \{E(G_1), E(G_2), \ldots, E(G_n)\}$ are edges between vertices of $S_1, S_2, \ldots, S_n$. Choosing any vertex in $V(G) \setminus V(G_i)$ to be in our deletion set does not reduce the number of induced paths of size three in G_i, so we have to choose at least $m-1$ vertices of each G_i to be in our deletion set D of G. Since we have shown that there exists a deletion set of size $n(m-1)$, the minimum size of a deletion set of G is $n(m-1)$. Therefore, any deletion set D of G of size $n(m-1)$ must be in the following form: $D = D_1 \cup D_2 \cup \ldots \cup D_n$, where $D_i = S_i \setminus \{s_i\}$ for all $1 \leq i \leq n$ and an arbitrary vertex $s_i \in S_i$. Note that by construction, the deletion set that corresponds to an H-word of size n of H is a minimum deletion set and any minimum deletion set of size $n(m-1)$ corresponds to an H-word of size n. Furthermore, note that since D_s and D_t are minimum deletion sets of size $n(m-1)$ and $k = n(m-1)+1$, a reconfiguration sequence $D_s \leftrightsquigarrow D_t$ in $\mathcal{R}_{CVD\text{-}\textsc{Min}}(G, k)$ is an alternating sequence of deletion sets of size $n(m-1)$ and of deletion sets of size $n(m-1)+1$. Therefore, since any deletion set of G of size $n(m-1)$ corresponds to an H-word of size n, the corresponding deletion sets of two adjacent H-words in H are two reconfiguration steps apart in G.

Now, we will define a reconfiguration step between two adjacent deletion sets D_i and D_{i+1} of G. By definition of adjacency, $(D_i \setminus D_{i+1}) \cup (D_{i+1} \setminus D_i) = \{u\}$ for some $u \in V(G)$. If $u \in D_i$, we remove u to get D_{i+1}. Otherwise, if $u \in D_{i+1}$, we add u to get D_{i+1}.

We now prove that $D_s \leftrightsquigarrow D_t$ in $\mathcal{R}_{CVD\text{-}\textsc{Min}}(G,k)$ holds if and only if there is a reconfiguration sequence of H-words in H between w_s and w_t. We first prove the if direction. Suppose that $w_s \leftrightsquigarrow w_t$ in $\mathcal{R}_{H-word}(H,n)$ holds, and hence there exists a reconfiguration sequence in H between w_s and w_t. It suffices to prove that if there is a reconfiguration sequence between any two adjacent H-words of H, there is a reconfiguration sequence between the two deletion sets that correspond to the two H-words. Let the deletion sets D^{2i} and D^{2i+2} in G correspond to the adjacent H-words $w_i = e_1e_2\ldots e_n$ and $w_{i+1} = f_1f_2\ldots f_n$ in H, respectively. We will now show the correspondence between a reconfiguration step from w_i to w_{i+1} in H and the reconfiguration steps from D^{2i} to D^{2i+2} in G, for any $1 \leq i \leq n-1$. We know that two adjacent H-words differ by exactly one symbol. Let that symbol be the l^{th} symbol of w_i and w_{i+1}, so $e_l \neq f_l$. We know by construction that $D^{2i} = D^{2i}_1 \cup D^{2i}_2 \cup \ldots \cup D^{2i}_n$ where $D^{2i}_j = S_j \setminus \{e_j\}$ for all $1 \leq j \leq n$. Similarly, $D^{2i+2} = D^{2i+2}_1 \cup D^{2i+2}_2 \cup \ldots \cup D^{2i+2}_n$ where $D^{2i+2}_j = S_j \setminus \{f_j\}$ for all $1 \leq j \leq n$. Since w_i and w_{i+1} differ only at the l^{th} symbol, $D^{2i}_j = D^{2i+2}_j$ for all j except for $j = l$ because $e_l \neq f_l$. By construction, $|D^{2i}| = n(m-1) < k$. Let the first reconfiguration step be $D^{2i+1} = D^{2i} \cup \{f_l\}$ where, by Observation 2.2.1, D^{2i+1} is a deletion set of size k. The next and last reconfiguration step is $D^{2i+2} = D^{2i+1} \setminus \{e_l\}$ where D^{2i+2} is the deletion set that corresponds to the H-word w_{i+1}.

We now prove the only-if direction. Suppose that $D_s \leftrightsquigarrow D_t$ in $\mathcal{R}_{CVD\text{-}\textsc{Min}}(G,k)$ holds, and hence there exists a reconfiguration sequence in G between D_s and D_t. Recall that this reconfiguration sequence must be an alternating sequence of deletion sets of size $n(m-1)$ and of deletion sets of size $n(m-1)+1$ since $n(m-1)$ is the size of a minimum deletion set of G and $k = n(m-1)+1$. Since any H-word corresponds to a deletion set of size $n(m-1)$, it suffices to prove that if there is a reconfiguration sequence between two deletion sets D^{2i} and D^{2i+2} of G, where $|D^{2i}| = |D^{2i+2}| = n(m-1)$ and D^{2i} and D^{2i+1} are two reconfiguration steps apart in the reconfiguration sequence, there is a reconfiguration sequence between the two H-words that correspond to D^{2i} and D^{2i+2}. We know that $D^{2i} \setminus D^{2i+2} = \{u\}$ and $D^{2i+2} \setminus D^{2i} = \{v\}$ where $u, v \in V(G)$ and $u \neq v$. Since D^{2i} is adjacent to D^{2i+1} and D^{2i+1} is adjacent to D^{2i+2}, a reconfiguration sequence from D^{2i} to D^{2i+2} will be as follows: add vertex v and let D^{2i+1} be the resulting deletion set of size $n(m-1)+1 = k$. Then, remove u to obtain D^{2i+2}.

We will now show that v and u are in the same independent set S_j of G_j for a $j \in \{1, 2, \ldots, n\}$. Recall that while proving that the size of a minimum dominating set of G is $n(m-1)$, we showed that any deletion set D of G of size $n(m-1)$ must be in the following form: $D = D_1 \cup D_2 \cup \ldots \cup D_n$, where $D_f = S_f \setminus \{s_f\}$ for all $1 \leq f \leq n$ and an arbitrary vertex $s_f \in S_f$. Therefore, since D^{2i} is a deletion set of minimum size $n(m-1)$, it must be of the form $D^{2i} = D^{2i}_1 \cup D^{2i}_2 \cup \ldots \cup D^{2i}_n$ where $D^{2i}_l = S_l \setminus \{x_l\}$ for all $1 \leq l \leq n$ and for some vertex $x_l \in S_l$. Similarly, $D^{2i+2} = D_1^{2i+2} \cup D_2^{2i+2} \cup \ldots \cup D_n^{2i+2}$ where $D_l^{2i+2} = S_l \setminus \{y_l\}$ for all $1 \leq l \leq n$ and for some vertex $y_l \in S_l$. Thus, $D^{2i} \cap V(G_j) = S_j \setminus \{x_j\}$ for some vertex $x_j \in S_j$. Therefore, if $u \in S_j$ and $v \notin S_j$, then the removal of u from D^{2i+1} will create an induced path (u, z, x_j) of size three in the split graph G_j, for an arbitrary vertex $z \in C_j$. Similarly, if $u \notin S_j$ and $v \in S_j$, then $u \in S_h$ for a set $S_h \in G$ and $S_h \neq S_j$. Since $D^{2i+2} \cap V(G_h) = S_h \setminus \{y_h\}$ for some vertex $y_h \in S_h$, the removal of u from D^{2i+2} creates an induced path (u, z, y_h) of size three in the split graph G_h, for an arbitrary vertex $z \in C_h$. Therefore, v and u are in the same independent set S_j of G_j.

If $(x_l, x_{l+1}) \in E(G)$, since $x_l \notin D^{2i}$ for all $1 \leq l \leq n$, (c, x_l, x_{l+1}) creates an induced path in $G_l \cup G_{l+1}$, for an arbitrary vertex $c \in C_l$ where C_l represents the clique of G_l. Therefore, $(x_l, x_{l+1}) \notin E(G)$ and hence $(x_l, x_{l+1}) \in R$ for all $l \in \{1, 2, \ldots, n-1\}$. So $w_i = x_1 x_2 \ldots x_n$ is the corresponding H-word in H of D^{2i} where x_h is the h^{th} character of w_i for each $h \in \{1, 2, \ldots, n\}$. Similarly, since D^{2i+2} is a deletion set, $(y_l, y_{l+1}) \notin E(G)$ and hence $(y_l, y_{l+1}) \in R$ for all $l \in \{1, 2, \ldots, n-1\}$. So $w_{i+1} = y_1 y_2 \ldots y_n$ is the corresponding H-word in H of D^{2i+2} where y_h is the h^{th} character of w_{i+1} for each $h \in \{1, 2, \ldots, n\}$. Since $D^{2i} \setminus D^{2i+2} = \{u\}$ and $D^{2i+2} \setminus D^{2i} = \{v\}$ where $u, v \in V(G)$ and $u \neq v$ and $u, v \in S_j$, w_i and w_{i+1} differ only at the j^{th} symbol. So w_i and w_{i+1} are adjacent H-words. $\square$

Chapter 7

Dominating Set Reconfiguration

In this chapter, we study the $\mathcal{DS}$-MIN-R problem. The following results have appeared (in a slightly different form) in collaborative work with Ito et al. [24]. We show that $\mathcal{DS}$-MIN-R can be solved in time linear in the number of edges for paths, cographs, trees, and interval graphs. Furthermore, we show that the problem is PSPACE-complete for general graphs, bipartite graphs, and split graphs.

7.1 Linear-Time Algorithms

In this section, we show that $\mathcal{DS}$-MIN-R is solvable in time linear in the number of edges in paths, cographs, trees and interval graphs. The results of Sections 7.1.1 and 7.1.2 appeared in a slightly different form in the PhD thesis by Amer E. Mouawad [38].

7.1.1 Paths

In this section, we show that $\mathcal{DS}$-MIN-R is solvable in time linear in $|V(G)|$ for any path G. By Theorem 1 it suffices to prove that any instance of $\mathcal{DS}$-MIN-R has a canonical dominating set for a path G. To do so, we construct a canonical dominating set for a path G. Note that if $|V(G)| = 1$, then the unique vertex of G represents a dominating set. We assume that $|V(G)| \geq 2$ and $V(G) = \{1, 2, \ldots, n\}$ for some positive integer n. We denote the two 1 degree vertices by r and s, where r denotes the root of G and s the leaf of G. We start by labelling vertex s with 1. Then, we label each vertex of G with one of 1, 2 and 3, starting from the neighbouring vertex of s to the neighbouring vertex of r in the following pattern: $3, 2, 1, 3, 2, 1$ and so on. We label vertex r with 2 if its neighbour is labelled with 3, and with 3 otherwise.We denote by S the set of all vertices in G that are labelled with 3. For each $i \in \{1, 2\}$, we denote by V_i the set of all vertices in G that are labelled with i. Then, $\{V_1, V_2, S\}$ forms a partition of $V(G)$. Figure 7.1 (a) illustrates a path labelled in the described pattern.

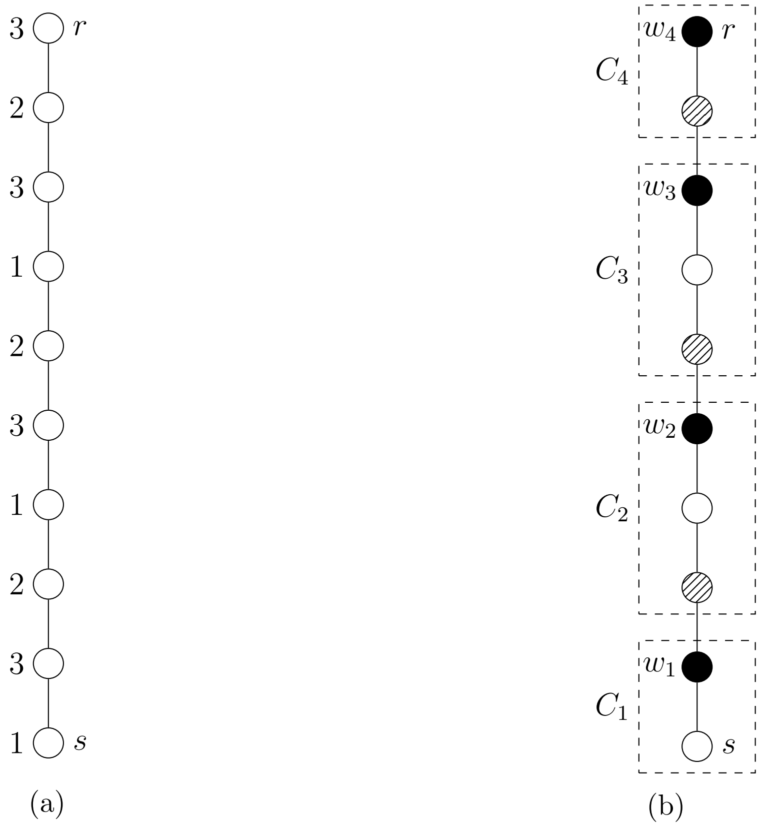


Figure 7.1: (a) The labelling of a path P, (b) the partition of $V(P)$ into C_1, C_2, C_3, C_4.

Using the canonical representation of S for G, we show that S is a deletion set of G. It suffices to show that for every $i \in \{1, 2, \ldots, |S|\}$, each $C_i \backslash w_i$ is a cluster graph.

Theorem 12. *For an instance $\mathcal{I} = (G, S_s, S_t, k)$ of $\mathcal{DS}$-Min-R for a path G, $\mathcal{DS}$-Min-R can be solved in time linear in $|V(G)|$.*

Proof. By Theorem 1, it suffices to prove that S forms a canonical dominating set of $\mathcal{I}$ for path G. By Lemma 4.5, to prove that S forms a canonical dominating set of $\mathcal{I}$, using the canonical representation of S for G, we have to show that S forms a minimum dominating set of G such that for every dominating set T of G, $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$, and that $S \cap V(G_i)$ is a dominating set of G_i. First, we prove that S is a dominating set of G. Then, we show that $S \cap V(G_i)$ is a dominating set of G_i. Afterwards, we prove that $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$. Finally, we show that S is a minimum dominating of G.

We first show that S is a dominating set of G. It suffices to show that both $V_1 \subseteq N(S)$ and $V_2 \subseteq N(S)$ hold. Let v be any vertex in V_1, and hence v is labelled with 1. If $v = s$, then its neighbour is labelled with 3. Note that $v \neq r$ since r is either labelled with 2 or 3. For any other vertex v, by the construction above, v has a neighbour labelled with 3. Therefore, $v \in N(S)$, which implies that $V_1 \subseteq N(S)$. Let u be any vertex in V_2, and hence u is labelled with 2. If $v = r$, it is labelled with 2 only when its neighbour is labelled with 3. Note that $v \neq s$ since s is labelled with 1. By the construction above, any vertex labelled with 2 has for neighbour a vertex labelled with 3. Therefore, $u \in N(S)$, which implies that $V_2 \subseteq N(S)$. Therefore, S forms a dominating set of G.

Now, we prove that $S \cap V(G_i)$ is a dominating set of G_i. By construction, since $V_1 \subseteq N(S)$ and the root $w_i \in S$ of G_i is the only vertex in $V(G_i)$ that is adjacent with a vertex in $V(G) \setminus V(G_i)$, we have $V_1 \cap V(G_i) \subseteq N(S \cap V(G_i))$. Similarly, we have $V_2 \cap V(G_i) \subseteq N(S \cap V(G_i))$. Therefore, $S \cap V(G_i)$ is a dominating set of G_i.

Using the canonical representation of S for G, we show that $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$. Figure 7.1 (b) illustrates the canonical representation of a path P partitioned into subsets C_1, C_2, C_3, C_4 where solid circles are vertices in S, striped circles are vertices in V_2, and white circles are vertices in V_1. We prove that C_i contains at least one vertex u such that $N[u] \subseteq C_i$. Therefore, the vertex u is not dominated by any vertex in T. Thus, $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$.

We first consider the case where C_i contains s, then the case where C_i contains r, and finally the case where all the nodes of C_i are internal vertices of G. First, consider the case where w_i has the degree one vertex s for neighbour. Then, $N[s] \subseteq C_i$ holds since s has w_i for unique neighbour. Second, consider the case where $C_i = C_{|S|}$ contains the degree one vertex r. If r is labelled with 3, then r has for unique neighbour a vertex $u \in C_i$ labelled with 1 or 2. Therefore, $N[r] \subseteq C_i$. Otherwise, if r is labelled with 2, then it has for unique neighbour w_i. Therefore, $N[r] \subseteq C_i$. Finally, consider the case where $i \neq |S|$ and w_i is an internal vertex such that the neighbour $u \in C_i$ of w_i is also an internal vertex in G. By construction, since w_i is labelled with 3, u is labelled with 1. Then, since u is an internal vertex, its neighbour $x \neq w_i$ is labelled with 2. Therefore, $N[u] \subseteq C_i$ holds for vertex u of w_i.

Finally, we prove that S is a minimum dominating of G. Since C_i contains at least one vertex u such that $N[u] \subseteq C_i$ and $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$, S is a minimum dominating set of G.

By Lemma 4.5, since S is a minimum dominating set of tree G such that for an arbitrary dominating set T of G, $|T \cap C_i| \geq 1$ for every $i \in \{1, 2, \ldots, |S|\}$, and $S \cap V(G_i)$ is a dominating set of G_i, S is a canonical dominating set of $\mathcal{I}$. □

7.1.2 Trees

In this section, we show that $\mathcal{DS}$-Min-R is solvable in time linear in $|V(G)|$ for any tree G. By Theorem 1 it suffices to prove that any tree G has a canonical dominating set. To do so, we will construct a canonical dominating set of an instance of $\mathcal{DS}$-Min-R for a tree G, similar to the construction of a canonical deletion set in Section 6.1.2.

We choose an arbitrary vertex r of degree one in G, and regard G as a rooted tree whose root is r. We first label each vertex in G with one of 1, 2 and 3, from the leaves of G to the root r of G, as in the following steps (1)–(3); intuitively, the vertices labelled with 3 form a dominating set of G, the vertices u labelled with 1 will be dominated by the parents of u, and the vertices v labelled with 2 will be dominated by at least one of the children of v:

(1) All leaves in G are labelled with 1.

(2) Pick an internal vertex v of G, which is not the root, such that all children of v have already been labelled. Then,
   - label v with 1 if all children of v are labelled with 2;
   - label v with 3 if at least one child of v is labelled 1; and
   - otherwise label v with 2.

(3) Label the root r with 2 if its child is labelled with 3, otherwise label r with 3.

For each $i \in \{1, 2\}$, we denote by V_i the set of all vertices in G that are labelled with i, and we denote by S the set of all vertices labelled with 3. Then, $\{V_1, V_2, S\}$ forms a partition of $V(G)$. Figure 7.2 (a) illustrates a tree T labelled in the described pattern.

Theorem 13. *For an instance $\mathcal{I} = (G, S_s, S_t, k)$ of $\mathcal{DS}$-Min-R for a tree G, $\mathcal{DS}$-Min-R can be solved in time linear in $|V(G)|$.*

Proof. By Theorem 1, it suffices to prove that S forms a canonical dominating set of $\mathcal{I}$. By Lemma 4.5, to prove that S forms a canonical dominating set of $\mathcal{I}$, we have to use the canonical representation of S for G to show that S forms a minimum dominating set of G such that for every dominating set T of G, $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$, and that $S \cap V(G_i)$ is a dominating set of G_i. First, we prove that S is a dominating set of G. Afterwards, we show that $S \cap V(G_i)$ is a dominating set of G_i. Then, we show that

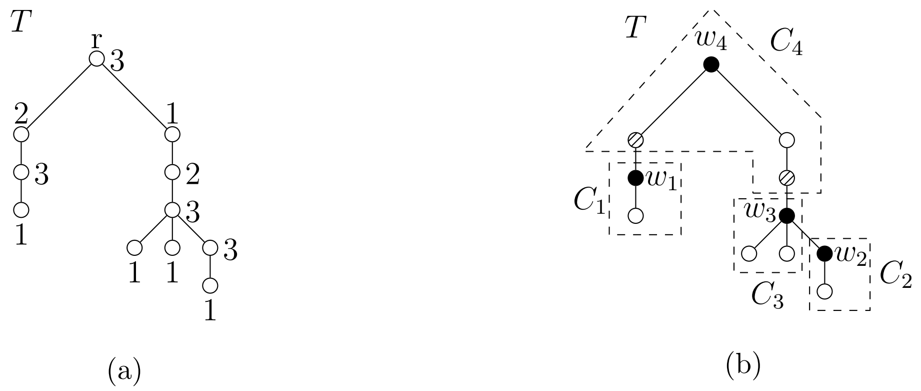


Figure 7.2: (a) The labelling of a tree T, (b) the partition of $V(T)$ into C_1, C_2, C_3, C_4.

$|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$. Finally, we prove that S is a minimum dominating set of G.

We first show that S is a dominating set of G. It suffices to show that both $V_1 \subseteq N(S)$ and $V_2 \subseteq N(S)$ hold. Let v be any vertex in V_1, and hence v is labelled with 1. Then, by the construction above, v is not the root of G and the parent of v must be labelled with 3. Therefore, $v \in N(S)$, which implies that $V_1 \subseteq N(S)$. Let u be any vertex in V_2, and hence u is labelled with 2. Then, u is not a leaf of G. Notice that 2 is assigned to a vertex only when at least one of its children is labelled with 3. Therefore, $u \in N(S)$, which implies that $V_2 \subseteq N(S)$. Therefore, S forms a dominating set of G.

Now, we prove that $S \cap V(G_i)$ is a dominating set of G_i. By construction, since $V_1 \subseteq N(S)$ and the root $w_i \in S$ of G_i is the only vertex in $V(G_i)$ that is adjacent with a vertex in $V(G) \setminus V(G_i)$, we have $V_1 \cap V(G_i) \subseteq N(S \cap V(G_i))$. Similarly, we have $V_2 \cap V(G_i) \subseteq N(S \cap V(G_i))$. Therefore, $S \cap V(G_i)$ is a dominating set of G_i.

Using the canonical representation of S for G, we show that for an arbitrary dominating set T of G, $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$. Figure 7.2 (b) illustrates the canonical representation of a tree T partitioned into subsets C_1, C_2, C_3, C_4 where solid circles are vertices in S, striped circles are vertices in V_2, and white circles are vertices in V_1. We prove that C_i contains at least one vertex u such that $N[u] \subseteq C_i$. Thus, the vertex u is not dominated by any vertex in T. Therefore, $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$.

Recall that all leaves in G are labelled with 1, and hence w_i is an internal vertex. We first consider the case where C_i has a vertex u that is a leaf of G, then the case where C_i contains the root r of G, and finally the case where all the nodes of C_i are internal vertices of G. First, consider the case where w_i has a child u which is a leaf of T. Then, $N[u] \subseteq C_i$

holds for the leaf u.

Second, consider the case where $C_i = C_{|S|}$ contains the root r of G. Recall that r is of degree one, and is labelled with either 2 or 3; we prove that $N[r] \subseteq C_{|S|}$ holds. If r is labelled with 2, then by construction, its unique child v is labelled with 3 and hence $v = w_{|S|}$. Therefore, $C_{|S|}$ contains both r and v, and hence $N[r] \subseteq C_{|S|}$ holds. On the other hand, if r is labelled with 3 and hence $r = w_{|S|}$, then its child v is labelled with either 1 or 2. Therefore, $C_{|S|}$ contains both r and v, and hence $N[r] \subseteq C_{|S|}$ holds.

Finally, consider the case where $i \neq |S|$ and w_i is an internal vertex such that all children of w_i are also internal vertices in G. Since w_i is labelled with 3, there exists at least one child u of w_i which is labelled with 1. Then, since u is an internal vertex, all children of u (and hence all "grandchildren" of w_i) are labelled with 2. Therefore, $N[u] \subseteq C_i$ holds for the child u of w_i.

Lastly, we prove that S is a minimum dominating of G. Since C_i contains at least one vertex u such that $N[u] \subseteq C_i$ and $|T \cap C_i| \geq 1$ holds for every $i \in \{1, 2, \ldots, |S|\}$, S is a minimum dominating set of G.

By Lemma 4.5, since S is a minimum dominating set of tree G such that for an arbitrary dominating set T of G, $|T \cap C_i| \geq 1$ for every $i \in \{1, 2, \ldots, |S|\}$, and $S \cap V(G_i)$ is a dominating set of G_i, S is a canonical dominating set of $\mathcal{I}$. $\square$

7.1.3 Cographs

In this section, we show that $\mathcal{DS}$-Min-R is solvable in time linear in $|E(G)|$ for any cograph G. By Theorem 1, it suffices to prove that G has a canonical dominating set. By Lemma 4.4, it suffices to consider the case where G is connected and we may assume that G has at least two vertices, because otherwise the problem is trivial.

Note that since G is connected, it must have been obtained by applying a join operation to two cographs G_u and G_v, hence $G = G_u \vee G_v$. By definition of join, if $a \in G_u$ and $b \in G_v$ then the set $\{a, b\}$ forms a dominating set.

We now construct a minimum dominating set C. If there exists a vertex $x \in V(G)$ that dominates all vertices in G, that is, $N[x] = V(G)$, we let $C = \{x\}$. Otherwise, we let $C = \{u, v\}$ where u and v are arbitrary vertices in G_u and G_v, respectively. It is clear that C forms a minimum dominating set of G.

For an instance $\mathcal{I} = (G, D_s, D_t, k)$ of $\mathcal{DS}$-Min-R, we will show that either we have a trivial no-instance or that for any dominating set D, there exists a reconfiguration sequence $D \leftrightsquigarrow C$ in $\mathcal{R}_{\mathcal{DS}\text{-MIN}}(G, k)$. In other words, either we can easily conclude that there is no reconfiguration sequence from D to C in $\mathcal{R}_{\mathcal{DS}\text{-MIN}}(G, k)$ or we can find a reconfiguration sequence from both D and C in time linear in $|E(G)|$.

We prove the following theorem.

Theorem 14. *The $\mathcal{DS}$-*Min*-R problem is solvable in time linear in $|E(G)|$ for any cograph G.*

Proof. Given an instance $\mathcal{I} = (G, D_s, D_t, k)$ of $\mathcal{DS}$-Min-R, we show that we can determine whether $\mathcal{I}$ is a no-instance or yes-instance in time linear in $|E(G)|$.

First, we check whether any cases of Proposition 2.3.1 hold. If any case holds, then $\mathcal{I}$ is a no-instance. Then, we show that in any other case, $\mathcal{I}$ is a yes-instance. To prove that we have a yes-instance, we show that for any dominating set D of G, $D \leftrightsquigarrow C$ in $\mathcal{R}_{\mathcal{DS}\text{-MIN}}(G, k)$ holds, where $k = |D| + 1$.

We construct a sequence $A = \langle D_0, D_1, \ldots, D_l \rangle$ from D to C such that all vertex subsets in the path from D to C in $\mathcal{R}_{\mathcal{DS}\text{-MIN}}(G, k)$ are dominating sets of cardinalities at most $|D| + 1$. First we show that there is such a sequence in the case where $|C| = 1$, and then in the case where $|C| = 2$.

In the case where $|C| = 1$, then $C = \{x\}$ where $N[x] = V(G)$. We construct the sequence A from D to C as follows, where a set in A is formed after each addition or removal of a vertex:
(1) If $x \notin D$, add x.
(2) Remove all vertices in $D \setminus x$ one by one.

The resulting set in Step (1) is a dominating set of size at most $|D| + 1$ since x dominates all vertices in G. All the vertex subsets formed in Step (2) contain vertex x, thus they are dominating sets. Furthermore, they are all obtained by the removal of a vertex from the resulting vertex of Step (1), therefore they have size at most $|D| + 1$. Hence, all the vertex subsets in the path from D to C in $\mathcal{R}_{\mathcal{DS}\text{-MIN}}(G, k)$ are dominating sets of size at most $|D| + 1$. Hence, in this case, $D \leftrightsquigarrow C$ in $\mathcal{R}_{\mathcal{DS}\text{-MIN}}(G, k)$ holds.

In the case where $|C| = 2$, $C = \{u, v\}$ where $u \in V(G_u)$ and $v \in V(G_v)$. Since C is a minimum dominating set of size 2, $|D| \geq 2$. Therefore, we have $D \subseteq V(G_u) \cup V(G_v)$. Hence, we must have $D \cap V(G_u) \neq \emptyset$ or $D \cap V(G_v) \neq \emptyset$. Assume the latter, the other case is symmetric. We let $h = 0$ and construct the sequence A of G between D and C in $\mathcal{R}_{\mathcal{DS}\text{-MIN}}$ (G, k) as follows:
(1) If $v \notin D$, add v, let D_1 be the resulting set, and set $h = 1$.
(2) Remove a vertex $a \in D_1 \cap V(G_v) \setminus \{v\}$ if it exists, let D_2 be the resulting set, and set $h = h + 1$.
(3) If $u \notin D_2$, add u, let D_3 be the resulting set, and set $h = h + 1$.

Then, for $i \in \{h, \ldots, l\}$, remove a vertex $c \in D_{i-1} \setminus \{u, v\}$. Note that for every i, D_i in the sequence A is obtained from D_{i-1} by the addition or removal of a single vertex.

Now, we prove that each vertex subset of A is a dominating set of size at most $|D|+1$.

In Step (1), D_1 is obtained by the addition of the vertex v if $v \notin D$, so by Observation 2.2.1, D_1 is a dominating set and $|D_1| \leq |D|+1$. In Step (2), D_2 is obtained by the removal of a vertex $a \in D_1 \cap V(G_v) \setminus \{v\}$ and $v \in D_2$, so D_2 is a dominating set of G and $|D_2| \leq |D_1|-1 = |D|$. In Step (3), similarly to Step (1), D_3 is obtained by the addition of the vertex u if $u \notin D_2$, so D_3 is a dominating set and $|D_3| \leq |D_2|+1 = |D|+1$. Note that each D_i contains both $\{u, v\}$ and $|D_i| \leq |D_3|$ for all $i \in \{4, 5, \ldots, l\}$. Since $|D_3| \leq |D|+1$, each D_i is a dominating set of G of cardinality at most $|D|+1$.

□

7.1.4 Interval Graphs

In this subsection, we show that $\mathcal{DS}$-Min-R is solvable in time linear in $|E(G)|$ for any interval graph G. For a given graph G, it can be determined in time linear in $|E(G)|$ whether G is an interval graph, and if so obtain an interval representation of G [35]. By Theorem 1, it suffices to prove that an instance of $\mathcal{DS}$-Min-R has a canonical dominating set for any interval graph. By Lemma 4.4, it suffices to consider the case where G is connected. We will construct a canonical dominating set of an instance of $\mathcal{DS}$-Min-R for an interval graph G using the well-known vertex numbering algorithm [44]. Let H be an interval representation of G. For an interval $I \in H$, we denote by $l(I)$ and $r(I)$ the left and right endpoints of I, respectively. We also refer to the values $l(I)$ and $r(I)$ as the *l-value* and *r-value* of I, respectively.

Fact 7.1.1. *For $u, v \in V(G)$, $uv \in E(G)$ if and only if $r(u) \geq l(v)$ and $r(v) \geq l(u)$ [35].*

We first label each vertex in G with one of 1, 2 and 3, from left to right, such that the vertices labelled with 3 will form a dominating set of G :

(1) Pick the unlabelled vertex v_i which has the minimum r-value among all unlabelled vertices, and label v_i with 1.

(2) Let v_j be the vertex in $N[v_i]$ which has the maximum r-value among all vertices in $N[v_i]$. Note that v_j may have been already labelled, and $v_j = v_i$ may hold. We label or relabel v_j with 3.

(3) For each unlabelled vertex in $N(v_j)$, we label it with 2.

We execute phases (1)–(3) until all vertices are labelled. For each $i \in \{1, 2\}$, we denote by V_i the set of all vertices in G that are labelled with i, and we denote by S the set of vertices labelled with 3. Then, $\{V_1, V_2, S\}$ forms a partition of $V(G)$. Figure 7.3 illustrates the labelling of an interval graph in the interval representation of this graph, where $i \rightarrow j$ represents the relabelling of a vertex from a label i to a label j.

By construction, each vertex in V_1 and V_2 has for neighbour a vertex in S. Therefore, S forms a dominating set of G. Note that $|V_1| \leq |S|$ since a vertex labelled with 1 is dominated by exactly one vertex labelled with 3 and a vertex labelled with 3 dominates at most one vertex labelled with 1. Since a vertex v_i labelled with 1 in Phase (1) is relabelled with 3 in Phase (2) if it has the maximum r-value among all the vertices in $N[v_i]$, the following observation holds.

Observation 7.1.1. *A vertex $u \in V(G)$ that is labelled with 1 and is relabelled with 3 has the maximum r-value among all the vertices in $N[u]$.*

Observation 7.1.2. *A vertex $w \in V(G)$ labelled with 3 was either labelled with 1 and got relabelled with 3, or has for a neighbour a vertex labelled with 1.*

Observation 7.1.3. *A vertex $u \in V(G)$ labelled with 1 has for a neighbour a vertex w labelled with 3 which has the maximum r-value among all vertices in $N[u]$.*

Before stating Theorem 15, we introduce a canonical representation of S for the interval graph G. Assume that the vertices in S are ordered as $w_1, w_2, \ldots, w_{|S|}$ such that $r(w_1) < r(w_2) < \cdots < r(w_{|S|})$. For each $i \in \{1, 2, \ldots, |S|\}$, we define the vertex subset C_i of $V(G)$, as follows:

$$C_i = \begin{cases} \{v \mid \quad r(v) \leq r(w_1) \} & \text{if } i = 1; \\ \{v \mid r(w_{i-1}) < r(v) \leq r(w_i) \} & \text{if } 2 \leq i \leq |S| - 1; \\ \{v \mid r(w_{|S|-1}) < r(v) \} & \text{if } i = |S|. \end{cases} \tag{7.1}$$

For each $i \in \{1, 2, \ldots, |S|\}$, let $C_i^- = C_1 \cup C_2 \cup \cdots \cup C_i$ and $C_i^+ = C_{i+1} \cup C_{i+2} \cup \cdots \cup C_{|S|}$. Note that $\{C_1, C_2, \ldots, C_{|S|}\}$ forms a partition of $V(G)$ such that

$$S \cap C_i = \{w_i\} \tag{7.2}$$

holds for every $i \in \{1, 2, \ldots, |S|\}$. Figure 7.3 illustrates C_1 with solid intervals, C_2 with dashed intervals, C_3 with dotted intervals, and C_4 with dash-dotted intervals.

Note that, by Equation 7.1, $C_i \cap C_j = \emptyset$ where $i \neq j$. Therefore, since a vertex labelled with 3 dominates at most one vertex labelled with 1, two vertices $u, v \in V_1$, where $u \neq v$, belong to two distinct vertex subsets of $\{C_1, C_2, \ldots, C_{|S|}\}$. Since, $|V_1| \leq |S|$ and each vertex of V_1 belongs to a unique vertex subset of $\{C_1, C_2, \ldots, C_{|S|}\}$, there exists $|S| - |V_1|$ vertex subsets of $\{C_1, C_2, \ldots, C_{|S|}\}$ that do not contain a vertex of V_1 and $|V_1|$ vertex subsets that contain a vertex of V_1.

Observation 7.1.4. *The set $\{C_1, C_2, \ldots, C_{|S|}\}$ of vertex subsets is composed of subsets containing exactly one unique vertex of V_1 and of subsets that do not contain a vertex of V_1.*

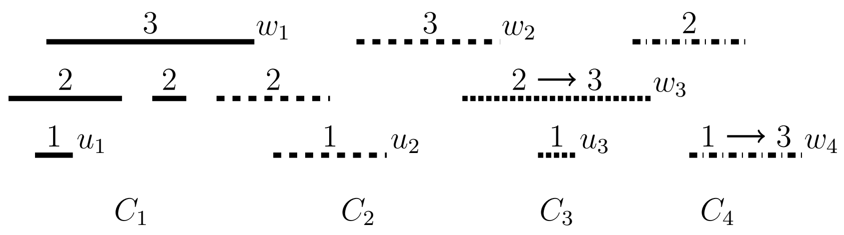


Figure 7.3: The labelling of an interval graph in the interval representation.

Theorem 15. *For an instance $\mathcal{I} = (G, S_s, S_t, k)$ of $\mathcal{DS}$-Min-R for an interval graph G, $\mathcal{DS}$-Min-R can be solved in time linear in $|E(G)|$.*

Proof. By Theorem 1, it suffices to prove that S forms a canonical dominating set of $\mathcal{I}$. Thus, since S is a dominating set, we prove that S is a minimum dominating set and then, we show that $T \leftrightsquigarrow S$ holds in $\mathcal{R}_{\mathcal{DS}\text{-MIN}}(G, k)$ for every dominating set T of G and $k = |T| + 1$.

To prove that S is a minimum dominating set, we show that $|T \cap C_j| \geq 1$ holds for every $j \in \{1, 2, \ldots, |S|\}$, for an arbitrary dominating set T of G. Assume instead that $T \cap C_i = \emptyset$ holds for some index $i \in \{1, 2, \ldots, |S|\}$. By Observation 7.1.4, it suffices to first show that this assumption yields a contradiction when $V_1 \cap C_i = \emptyset$, and then show that it yields a contradiction when $V_1 \cap C_i = \{u_i\}$ where $u_i \in V_1$.

First, consider the case where $V_1 \cap C_i = \emptyset$. Since $V_1 \cap C_i = \emptyset$ and C_i contains the vertex w_i labelled with 3, by Observations 7.1.1 and 7.1.2, w_i has the maximum r-value among $N[w_i]$. Therefore, since by assumption $T \cap C_i = \emptyset$, $w_i \in C_i$ must be dominated by some vertex v in $C_i^- \setminus C_i$. Note that by Equation 7.1, $r(w_i) > r(w_{i-1})$. Then, since v dominates w_i, we have $vw_i \in E(G)$. Note that $r(w_i) > r(v)$ since $v \in C_i^- \setminus C_i$. Therefore, since $vw_i \in E(G)$, by Fact 7.1.1, we must have $l(w_i) \leq r(v)$. Note that v can not be in $C_{|S|}$ since if $i = |S|$, $v \in C_{|S|}^- \setminus C_{|S|}$. Therefore, since $v \in C_i^- \setminus C_i$, by Eq. (7.1), we have $r(v) \leq r(w_{i-1})$. Since $r(w_{i-1}) < r(w_i)$ and $l(w_i) \leq r(v) \leq r(w_{i-1})$, we have $l(w_i) \leq r(w_{i-1}) < r(w_i)$. Therefore, $w_i \in N(w_{i-1})$ holds. Thus, by construction, since w_{i-1} is labelled with 3 and $w_i \in N(w_{i-1})$, w_i must be labelled with 2. This contradicts the assumption that w_i is labelled with 3.

Next, we consider the case where $V_1 \cap C_i = \{u_i\}$. Since by assumption $T \cap C_i = \emptyset$, $u_i \in V_1$ must be dominated by at least one vertex in $C_i^- \setminus C_i$ or C_i^+. If u_i is dominated by a vertex in $C_i^- \setminus C_i$, then by the same arguments as above, u_i must be labelled with 2 even though u_i is in V_1, yielding a contradiction. Therefore, u_i must be dominated by a vertex v in C_i^+. Then, since v dominates u_i, $vu_i \in E(G)$. Hence, we have $v \in N(u_i) \subset N[u_i]$.

Also, since $v \in C_i^+$, by Eq. (7.1), we have $r(w_i) < r(v)$. However, by Observation 7.1.3, $w_i \in S$ is chosen as the vertex in $N[u_i]$ that has the maximum r-value among all vertices in $N[u_i]$. Thus, $r(w_i) < r(v)$ contradicts the assumption that w_i has the maximum r-value among all vertices in $N[u_i]$.

Therefore, we have

$$|T \cap C_j| \geq 1 \tag{7.3}$$

for every $j \in \{1, 2, \ldots, |S|\}$.

Finally, we prove that for every dominating set T of G, $T \leftrightsquigarrow S$ holds in $\mathcal{R}_{\mathcal{DS}\text{-MIN}}$ (G, k), where $k = |T| + 1$. First, we construct a sequence σ of sets between T and S. Then, we prove that each set in σ is a dominating set of G. Finally, we prove that each set in σ is of cardinality at most $|T| + 1$.

We let $T_0 = T$ be the first set of the sequence σ. For each i from 1 to $|S|$, we set $h = 1$ and construct a subsequence $\sigma_i = \langle T_i^1, T_i^2, \ldots, T_i^{l_i} \rangle$ of σ in the following phases, for some positive integer l_i, where each set in σ_i is formed after the addition or removal of a vertex, and $T_i = T_i^{l_i}$:

(1) If $w_i \notin T_{i-1}$, add vertex w_i, let T_i^1 be the resulting set, and set $h = 2$.

(2) For $j \in \{h, \ldots, l_i\}$, remove an arbitrary vertex $c \in T_i^{j-1} \cap \big(C_i \setminus \{w_i\}\big)$ and let T_i^j be the resulting set.

Note that in the transformation σ_i, each set is formed after a removal or addition of a vertex.

We now show that each set in σ forms a dominating set of G. First, we use induction to show that each set T_i forms a dominating set of G for each $i \in \{1, 2, \ldots, |S|\}$. Then, we show that each set in σ is a dominating set of G by showing that each set of σ_i is a dominating set of G.

Using induction, we show that each set T_i forms a dominating set of G. Since $T_0 = T$ and T is a dominating set of G, the base case holds. Suppose that T_{i-1} is a dominating set of G. By Equation 7.2, $S \cap C_i = \{w_i\}$ and by construction of T_i, we have $T_i \cap C_i = \{w_i\}$ for every $i \in \{1, 2, \ldots, |S|\}$. Therefore, $T_i \cap C_i = S \cap C_i$. Furthermore, since $C_i^- = C_1 \cup C_2 \cup \cdots \cup C_i$ and $T_i \cap C_i = S \cap C_i$ for every $i \in \{1, 2, \ldots, |S|\}$, we have $T_i \cap C_i^- = S \cap C_i^-$. By construction of σ_i, the only vertices added or removed in σ_i belong to C_i. Therefore, since $T_0 = T$, the only vertices that are added or removed in forming T_i belong to C_i^-. Thus, we have $T_i \cap C_i^+ = T \cap C_i^+$. Note that if we can show that any vertex $v \in V(G)$ is dominated by a vertex in T_i, then we have proved that T_i is a dominating set.

Case 1: First, suppose that $r(v) \leq r(w_i)$. By Equation 7.1, $v \in C_i^-$. Note that $C_i^- = C_1 \cup C_2 \cup \cdots \cup C_i$ and by Equation 7.2, $S \cap C_i = \{w_i\}$. Therefore, since S is a dominating set of G, a vertex in C_i^- is dominated by a vertex in $S \cap C_i^-$. Then, since $T_i \cap C_i^- = S \cap C_i^-$,

v is dominated by a vertex in $T_i \cap C_i^-$.
Case 2: Second, suppose that $r(w_i) \le l(v)$. By Equation 7.1, $v \in C_i^+$. Note that $C_i^+ = C_{i+1} \cup C_{i+2} \cup \cdots \cup C_{|S|}$ and by Equation 7.2, $S \cap C_i = \{w_i\}$. Therefore, since S is a dominating set of G, a vertex in C_i^+ is dominated by a vertex in $S \cap C_i^+$. Then, since $T_i \cap C_i^+ = S \cap C_i^+$, v is dominated by a vertex in $T_i \cap C_i^+$.
Case 3: Finally, suppose that $l(v) < r(w_i) < r(v)$. Then, by definition of an interval graph, $vw_i \in E(G)$. Hence, v is dominated by $w_i \in T_i$.

Since both the base case and the induction step hold, by induction, each set T_i is a dominating set of G, for each $i \in \{1, 2, \dots, |S|\}$.

We now show that each set of σ_i is a dominating set of G. By Observation 2.2.1, all the sets formed in Phases (1) and (2) are dominating sets since each of them is a superset of T_i. Therefore, since each set in the subsequence σ_i forms a dominating set of G, each set in σ is a dominating set of G.

Finally, we prove that each set in the sequence from T to S in $\mathcal{R}_{\mathcal{P}\text{-MIN}}\,(G,k)$ is a set of cardinality at most $|T|+1$. We show that $|T_{i-1}| \ge |T_i|$ for each $i \in \{1, 2, \dots, |S|\}$ and that the maximum cardinality of any set in σ_i is $|T_{i-1}|+1$. Then, since $|T_{i-1}| \ge |T_i|$, the cardinality of any set in the sequence from $T_0 = T$ to $T_{|S|} = S$ in $\mathcal{R}_{\mathcal{P}\text{-MIN}}\,(G,k)$ is at most $|T|+1$.

We show that $|T_{i-1}| \ge |T_i|$ holds in the cases where $w_i \in T_{i-1}$ and $w_i \notin T_{i-1}$.
Case 1: Suppose $w_i \in T_{i-1}$. Since $w_i \in T_{i-1}$, no vertex is added in Phase (1) nor in Phase (2). Therefore, each set in σ_i is of cardinality at most $|T_{i-1}|$.
Case 2: Suppose $w_i \notin T_{i-1}$. Since T_{i-1} is a dominating set of G, by Equation 7.3, $|T_{i-1} \cap C_i| \ge 1$ holds for every $i \in \{1, 2, \dots, |S|\}$. Therefore, since $w_i \notin T_{i-1}$, we have $T_{i-1} \cap \big(C_i \setminus \{w_i\}\big) \ne \emptyset$. Since there is only a single vertex addition in the subsequence σ_i, each set in σ_i is of cardinality at most $|T_{i-1}|+1$.

We have thus proved that S is a minimum dominating set and that $T \leftrightsquigarrow S$ holds in $\mathcal{R}_{\mathcal{DS}\text{-MIN}}(G,k)$ for every dominating set T of G and $k = |T|+1$. By definition of a canonical dominating set, S forms a canonical dominating set of $\mathcal{I}$. $\square$

7.2 PSPACE-completeness

7.2.1 General Graphs

In this section, we prove that $\mathcal{DS}$-MIN-R is PSPACE-complete for general graphs. Stronger results are proven in the collaborative work with Ito et al. [24].

A problem that will be used in proving our results is the PSPACE-complete $\mathcal{VC}$-MIN-R problem [29] defined as follows:

Input: A graph G, two vertex covers S_s and S_t of G, and an integer threshold $k \geq \max\{|S_s|, |S_t|\}$.
Question: Is there a path from S_s to S_t in $\mathcal{R}_{\mathcal{VC}\text{-Min}}(G, k)$?

We will show that $\mathcal{DS}$-MIN-R is PSPACE-hard for general graphs by a polynomial-time reduction from $\mathcal{VC}$-MIN-R.

Theorem 16. *$\mathcal{DS}$-MIN-R is* PSPACE*-complete for general graphs.*

Proof. Our reduction follows from the classical reduction from VERTEX COVER to DOMINATING SET [18]. Let (G, C_s, C_t, k) be an instance of $\mathcal{VC}$-MIN-R where $V(G) = \{v_1, v_2, \ldots, v_n\}$ and $E(G) = \{e_1, e_2, \ldots, e_m\}$. Without loss of generality, it is assumed that G has no isolated vertices. We construct the corresponding graph G' as follows. For every edge $(v_i, v_j) \in E(G)$, we add a new vertex v_{ij} and join it with each of v_i and v_j by two new edges (v_i, v_{ij}) and (v_{ij}, v_j). Then, let $(G', D_s = C_s, D_t = C_t, k' = k)$ be the corresponding instance of $\mathcal{DS}$-MIN-R. Note that this instance can be constructed in polynomial time.

We now prove that $D_s \leftrightsquigarrow D_t$ in $\mathcal{R}_{DS\text{-Min}}(G', k')$ holds if and only if $C_s \leftrightsquigarrow C_t$ in $\mathcal{R}_{VC\text{-Min}}(G, k)$ holds. We first prove the if direction. Note that the reconfiguration rule of these two problems is the same since the symmetric difference of each problem is of size one. Because both problems employ the same reconfiguration rule, it suffices to prove that any vertex cover C of G forms a dominating set of G'. By definition of a vertex cover, a vertex cover C of G must include at least one vertex of an edge $(v_i, v_j) \in G$. Since v_i and v_j dominate each other and dominate v_{ij} in G', any vertex cover of G forms a dominating set of G'. Therefore, if there exists a reconfiguration sequence between C_s and C_t in $\mathcal{R}_{VC\text{-Min}}(G, k)$, then there exists is a reconfiguration sequence between D_s and D_t in $\mathcal{R}_{DS\text{-Min}}(G', k')$.

We now prove the only-if direction. Suppose that there is a reconfiguration sequence σ between D_s and D_t in $\mathcal{R}_{DS\text{-Min}}(G', k')$. Note that since $D_s = C_s$ and $D_t = C_t$, D_s and D_t do not contain a vertex in $V(G') \setminus V(G)$. Thus, if a vertex v_{ij} in $V(G') \setminus V(G)$ is touched in σ, then v_{ij} must first have been added in σ. By construction, $N_G(v_{ij}) = \{v_i, v_j\}$ and $v_i \in N_G(v_j)$ and $v_j \in N_G(v_i)$, thus both $N_G[v_{ij}] \subseteq N_G[v_i]$ and $N_G[v_{ij}] \subseteq N_G[v_j]$ hold. Therefore, instead of adding v_{ij} to a dominating set of σ, we can either add v_i or v_j since they both dominate v_{ij} and its neighbours, and obtain a reconfiguration sequence of dominating sets in G' between D_s and D_t which only touch vertices in G. Then, this reconfiguration sequence of dominating sets between D_s and D_t in $\mathcal{R}_{DS\text{-Min}}(G', k')$ is also a reconfiguration sequence of vertex covers between C_s and C_t in $\mathcal{R}_{VC\text{-Min}}(G, k)$. $\square$

7.2.2 Split Graphs

In this section, we prove that $\mathcal{DS}$-Min-R is PSPACE-complete for split graphs. We will show that it is PSPACE-complete for split graphs by a polynomial-time reduction from $\mathcal{VC}$-Min-R.

Theorem 17. *$\mathcal{DS}$-Min-R is PSPACE-complete for split graphs.*

Proof. Our reduction follows from the NP-completeness proof of Dominating Set on split graphs. [3].

Let (G, C_s, C_t, k) be an instance of $\mathcal{VC}$-Min-R where $V(G) = \{v_1, v_2, \ldots, v_n\}$ and $E(G) = \{e_1, e_2, \ldots, e_m\}$. Figure 7.4 (a) illustrates a vertex cover $\{v_2, v_4\}$ of a graph. Without loss of generality, it is assumed that G has no isolated vertices. We construct the corresponding graph G' as follows. We let $V(G') = A \cup B$, where $A = \{a_i \mid v_i \in V(G)\}$, and $B = \{b_1, b_2, \ldots, b_m\}$. We add all edges between the vertices of A to form a clique, and each vertex $b_i \in B$ has two neighbours in A which correspond to the endpoints of the edge $e_i \in E(G)$. Figure 7.4 (b) illustrates a dominating set $\{a_2, a_4\}$ of the corresponding split graph of the graph in part (a). Let G' be the resulting graph, and let $(G, D_s = C_s, D_t = C_t, k' = k)$ be the corresponding instance of $\mathcal{DS}$-Min-R. Clearly, this instance can be constructed in polynomial time. We now prove that $D_s \leftrightsquigarrow D_t$ in $\mathcal{R}_{DS\text{-Min}}(G', k')$ holds if and only if $C_s \leftrightsquigarrow C_t$ in $\mathcal{R}_{VC\text{-Min}}(G, k)$ holds.

We first prove the if direction. Because both of these problems employ the same reconfiguration rule, it suffices to prove that any vertex cover C of G forms a dominating set of G'. Since $C \subseteq V(G) = A$ and A is a clique, C dominates all the vertices in A. Furthermore, since C contains at least one endpoint of each edge $e \in E(G)$ and each vertex $b_i \in B$ has two neighbours which correspond to the endpoints of the edge $e_i \in E(G)$, at least one neighbour of each vertex b_i is in C. Thus, C is a dominating set of G'.

We now prove the only-if direction. Suppose that there is a reconfiguration sequence σ between D_s and D_t in $\mathcal{R}_{DS\text{-Min}}(G', k')$. By construction, for each vertex $b_i \in B$ corresponding to the edge $e_i = v_u v_w$ in $E(G)$, both $N_G[b_i] \subseteq N_G[v_u]$ and $N_G[b_i] \subseteq N_G[v_w]$ hold. Therefore, if a vertex b_i is added in σ, we can instead add either of its neighbours, since they dominate b_i and its neighbours, and obtain a reconfiguration sequence of dominating sets in G' between D_s and D_t which only touch vertices in A. Since for any dominating set $D \subseteq A = V(G)$ each vertex $b_i \in B$ is dominated by at least one vertex in $D \subseteq V(G)$, D forms a vertex cover of G. Thus, if $D_s \leftrightsquigarrow D_t$ in $\mathcal{R}_{DS\text{-Min}}(G', k')$ holds, $C_s \leftrightsquigarrow C_t$ in $\mathcal{R}_{VC\text{-Min}}(G, k)$ holds. □

By the reduction given in the proof of Theorem 17, the following lemma clearly holds.

Lemma 7.2.1. *Let (G, D_s, D_t, k) be an instance of $\mathcal{DS}$-*Min*-R, then $\mathcal{DS}$-*Min*-R is* `PSPACE`*-complete for split graphs even if $D_s, D_t \subseteq A$.*

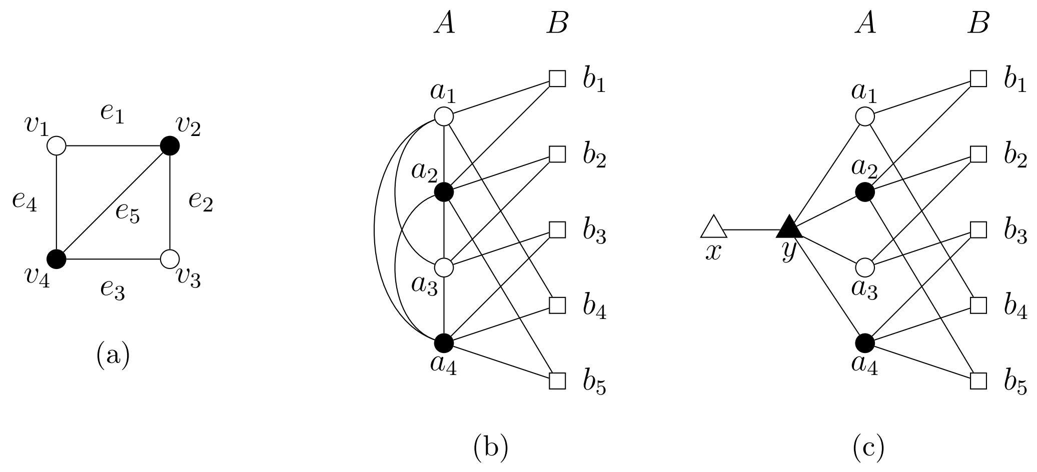


Figure 7.4: (a) Vertex cover $\{v_2, v_4\}$ of a graph, (b) dominating set $\{a_2, a_4\}$ of the corresponding split graph, and (c) dominating set $\{a_2, a_4, y\}$ of the corresponding bipartite graph.

7.2.3 Bipartite Graphs

In this section, we prove that $\mathcal{DS}$-Min-R is `PSPACE`-complete for bipartite graphs by a polynomial-time reduction from the same problem on split graphs.

Theorem 18. *$\mathcal{DS}$-*Min*-R is PSPACE-complete for bipartite graphs.*

Proof. Our reduction follows from the `NP`-completeness proof of Dominating Set on bipartite graphs that uses a polynomial-time reduction from the same problem on split graphs [3].

Given an instance (G, D_s, D_t, k) of $\mathcal{DS}$-Min-R on split graphs where $V(G) = \{v_1, v_2, \ldots, v_n\}$, $E(G) = \{e_1, e_2, \ldots, e_m\}$, and G is a split graph, by definition of a split graph in Section 2.1, G can be partitioned into two subsets $A = \{a_i \mid v_i \in V(G)\}$ and $B = \{b_1, b_2, \ldots, b_m\}$ such that A forms a clique and B an independent set. By Lemma 7.2.1, the $\mathcal{DS}$-Min-R problem remains `PSPACE`-complete even if $D_s, D_t \subseteq A$, therefore we can assume that $D_s, D_t \subseteq A$. We construct the corresponding graph G' as follows. We make

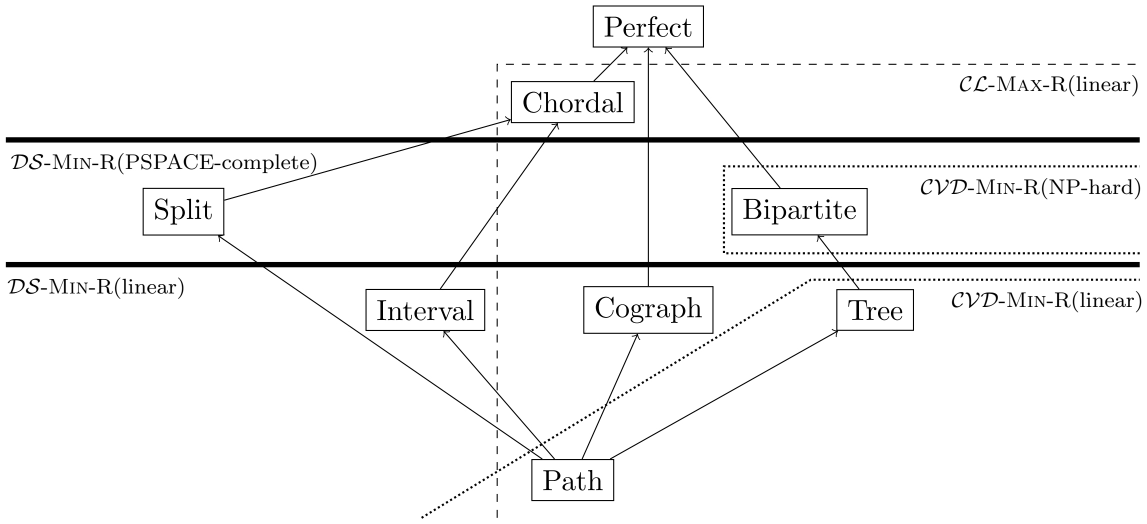
A an independent set by removing all the edges that join two vertices of A and we add two vertices x and y. In addition, we add an edge $e = (x, y)$ and join vertex y with every vertex of A. Let G' be the corresponding graph. Note that G' is a bipartite graph as it can be partitioned into two independent sets $A' = A \cup \{x\}$ and $B' = B \cup \{y\}$. Figure 7.4 (c) illustrates a dominating set $\{a_2, a_4, y\}$ of the corresponding bipartite graph of the split graph in part (b). Let $(G', D'_s = D_s \cup \{y\}, D'_t = D_t \cup \{y\}, k' = k + 1)$ be the corresponding instance of $\mathcal{DS}$-MIN-R for bipartite graphs. We now prove that $D'_s \leftrightsquigarrow D'_t$ in $\mathcal{R}_{DS\text{-MIN}}(G', k')$ holds if and only if $D_s \leftrightsquigarrow D_t$ in $\mathcal{R}_{DS\text{-MIN}}(G, k)$ holds.

We first prove the if direction. Suppose that there is a reconfiguration sequence between D_s and D_t in $\mathcal{R}_{DS\text{-MIN}}(G, k)$. Note that it suffices to show that for any dominating set D of G, $D \cup \{y\}$ forms a dominating set of size k' in G'. By construction, since the only removed edges are those that joined vertices in A and $B \subset N_G[D]$, $B \subset N_{G'}[D]$. The vertex y dominates A and x, hence $D \cup \{y\}$ forms a dominating set of size $k + 1 = k'$. Therefore, there exists a reconfiguration sequence between D'_s and D'_t in $\mathcal{R}_{DS\text{-MIN}}(G', k')$.

We then prove the only-if direction. Suppose that there is a reconfiguration sequence σ between D'_s and D'_t in $\mathcal{R}_{DS\text{-MIN}}(G', k')$. Note that the only vertices that dominate the vertex x are x and y, hence any dominating set must contain x or y and $N_{G'}[x] \subseteq N_{G'}[y]$. Since $y \in D'_s$ and $y \in D'_t$ and y is the unique vertex in G' that dominates x, we can assume that $y \in D'$ for any dominating set D' in σ. Also, note that since G is a split graph with A as a clique and B as an independent set, $N_G(b_i) \subseteq A$ for a vertex $b_i \in B$. Furthermore, since the only removed edges are those that joined vertices in A and the only edges added join y to every vertex of $V(G') \setminus B$, $N_{G'}(b_i) \subseteq A$ for a vertex $b_i \in B$. Also, note that y dominates each vertex in A. Recall that $D_s \subseteq A$ and $D_t \subseteq A$, and therefore if a vertex $b_i \in B$ is touched in σ, there exists a reconfiguration step in σ where b_i is added. Therefore, if a vertex $b_i \in B$ is added in σ, we can instead add either of its neighbours in A and obtain a reconfiguration sequence σ' of dominating sets in $\mathcal{R}_{DS\text{-MIN}}(G', k')$ between D'_s and D'_t which only touch vertices in A. Consider any dominating set $D' \in \sigma'$. We now show that $D' \cap V(G)$ forms a dominating set of size at most k in $V(G)$. Note that $|D'| \leq k'$, since D' is a dominating set in $\mathcal{R}_{DS\text{-MIN}}(G', k')$. Since $y \in D'$ and $y \notin V(G)$, we have $|D' \cap V(G)| \leq k' - 1 = k$. Note that since σ' is a reconfiguration sequence in $\mathcal{R}_{DS\text{-MIN}}(G', k')$ between $D'_s = D_s \cup \{y\}$ and $D'_t = D_t \cup \{y\}$ which only touches vertices in A, where $D_s, D_t \subseteq A$, we have $D' \cap V(G) \subseteq A$. Also, since $D' \cap V(G) \subseteq A$ and A forms a clique in G, $A \subseteq N_G[D' \cap V(G)]$. In addition, since $N_{G'}(y) = A \cup \{x\}$, each vertex in B is dominated by some vertex in $D' \cap V(G)$. We have thus shown that $D' \cap V(G)$ is a dominating set of G of cardinality at most k. Therefore, if $D'_s \leftrightsquigarrow D'_t$ in $\mathcal{R}_{DS\text{-MIN}}(G', k')$ holds, $D_s \leftrightsquigarrow D_t$ in $\mathcal{R}_{DS\text{-MIN}}(G, k)$ holds. □

Chapter 8

Conclusions and Directions For Future Work



Perfect
Chordal
$\mathcal{CL}$-Max-R(linear)
$\mathcal{DS}$-Min-R(PSPACE-complete)
$\mathcal{CVD}$-Min-R(NP-hard)
Split
Bipartite
$\mathcal{DS}$-Min-R(linear)
Interval
Cograph
Tree
$\mathcal{CVD}$-Min-R(linear)
Path

Figure 8.1: Our results, where each arrow represents the inclusion relationship between graph classes: $A \to B$ represents that A is properly included in B [10].

The results obtained in this thesis paint an interesting picture of the boundary between intractability and polynomial-time solvability of Clique Reconfiguration, Cluster Vertex Deletion Reconfiguration, and Dominating Set Reconfiguration from the viewpoint of graph classes, as illustrated in Figure 8.1. A problem of interest is to establish a connection between the complexity of the st-connectivity problem of a problem and the complexity of the underlying decision problem. We know that this relationship is not as easy as "P implies P" and "NP-complete implies PSPACE-complete".

In Section 5, we showed that the Clique Reconfiguration is solvable in time linear in the number of edges for paths, trees, cographs, bipartite and chordal graphs. The study by Ito et al. [32] extended this work and proved that this problem is solvable in polynomial time on planar and even-hole-free graphs, and is PSPACE-complete for perfect graphs. This is an example of a problem whose st-connectivity problem is solvable in polynomial time for many graph classes, and is PSPACE-complete in general. The only graph class for which the relationship between Clique Reconfiguration and its underlying decision problem is not "P implies P" is perfect graphs, where the st-connectivity problem is PSPACE-complete and the underlying decision problem is solvable in polynomial time. This result further shows that this relationship is not as subtle as "P implies P".

In Section 6, we showed that the Cluster Vertex Deletion Reconfiguration problem is solvable in time linear in the number of edges for paths and trees. We also proved that this problem is NP-hard on bipartite graphs and PSPACE-complete in general. In contrast to the related Clique Reconfiguration problem, the complexity of this problem on bipartite graphs is NP-hard. It would be interesting to solve the complexity of this problem on super-classes of paths, such as cographs, interval, and split graphs, as this would give us a better understanding on the connection between the complexity of these two problems. Furthermore, such a result would highlight the structure of the graph classes that lie at the boundary between tractability and PSPACE-completeness.

In Section 7, we showed that the Dominating Set Reconfiguration problem is solvable in time linear in the number of edges for paths, trees, cographs, and interval graphs. Furthermore, we showed that this problem is PSPACE-complete for split graphs, bipartite graphs, and in general. Having proven these results, we notice that for interval graphs, that have a path-like structure of cliques, this problem is solvable in linear time. However, for chordal graphs, which can be represented as a tree of cliques, this problem is PSPACE-complete. Therefore, the clique structure of a graph may be a good indicator of the complexity of this problem.

Having studied these three reconfiguration problems under classical complexity for familiar graph classes, a direction for future work would be to study these problems under the parameterized complexity [15] for the same graph classes. Parameterized complexity tries to confine the exponential "explosion" in the running time of a problem to the parameter instead of the input size. The main hierarchy of parameterized complexity classes is $FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq W[t]$, for every integer $t \geq 1$, where FPT denotes the fixed-parameter tractable class and W-hardness is the analogue of NP-hardness in classical complexity. To the best of our knowledge, Dominating Set Reconfiguration is the only problem among the three problems studied in this thesis that has been studied under parameterized complexity [40]. This problem was proven to be $W[2]$-hard when parameterized by $k + l$, where k is the size of the solutions and l is the length of the sequence of steps. A new problem of interest would be to find a connection between the classical complexity and parameterized complexity of these problems to get a "finer" classification of these problems, not only in the tractable and intractable case, but also the fixed-parameter tractable case.

References

[1] A. Aggarwal. *The Art Gallery Theorem: Its Variations, Applications and Algorithmic Aspects.* PhD thesis, 1984.

[2] A. V. Aho, J. E. Hopcroft, and J. Ullman. *Data Structures and Algorithms.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1983.

[3] A. A. Bertossi. Dominating Sets for Split and Bipartite Graphs. *Inf. Process. Lett.*, 19(1):37–40, 1984.

[4] M. Bonamy and N. Bousquet. Recoloring bounded treewidth graphs. *Electronic Notes in Discrete Mathematics*, 44:257–262, 2013.

[5] M. Bonamy, M. Johnson, I. Lignos, V. Patel, and D. Paulusma. On the diameter of reconfiguration graphs for vertex colourings. *Electronic Notes in Discrete Mathematics*, 38:161–166, 2011.

[6] P. S. Bonsma. Shortest Path Reconfiguration is PSPACE-hard. *CoRR*, abs/1009.3217, 2010.

[7] P. S. Bonsma. The complexity of rerouting shortest paths. *Theor. Comput. Sci.*, 510:1–12, 2013.

[8] P. S. Bonsma and L. Cereceda. Finding Paths Between Graph Colourings: PSPACE-completeness and Superpolynomial Distances. *Theor. Comput. Sci.*, 410(50):5215–5226, 2009.

[9] P. S. Bonsma, L. Cereceda, J. van den Heuvel, and M. Johnson. Finding Paths between Graph Colourings: Computational Complexity and Possible Distances. *Electronic Notes in Discrete Mathematics*, 29:463–469, 2007.

[10] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: A Survey*, volume 3 of *Monographs on Discrete Mathematics and Applications*. SIAM Society for Industrial and Applied Mathematics, Philadelphia, 1999.

[11] L. Cereceda, J. van den Heuvel, and M. Johnson. Connectedness of the graph of vertex-colourings. *Discrete Mathematics*, 308(5-6):913–919, 2008.

[12] D. Corneil, H. Lerchs, and L. Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163 – 174, 1981.

[13] R. Diestel. *Graph Theory.* Springer, August 2005.

[14] B. Ding and A. C. König. Fast Set Intersection in Memory. *PVLDB*, 4(4):255–266, 2011.

[15] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer, 1999.

[16] F. V. Fomin, S. Gaspers, D. Kratsch, M. Liedloff, and S. Saurabh. Iterative compression and exact algorithms. *Theor. Comput. Sci.*, 411(7-9):1045–1053, 2010.

[17] G. Fricke, S. M. Hedetniemi, S. T. Hedetniemi, and K. R. Hutson. γ-graphs of graphs. *Discussiones Mathematicae Graph Theory*, 31(3):517–531, 2011.

[18] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[19] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47 – 56, 1974.

[20] P. Gopalan, P. G. Kolaitis, E. N. Maneva, and C. H. Papadimitriou. The Connectivity of Boolean Satisfiability: Computational and Structural Dichotomies. *SIAM J. Comput.*, 38(6):2330–2355, 2009.

[21] J. Guo, H. Moser, and R. Niedermeier. Iterative Compression for Exactly Solving NP-Hard Minimization Problems. In *Algorithmics of Large and Complex Networks - Design, Analysis, and Simulation [DFG priority program 1126]*, pages 65–80, 2009.

[22] R. Haas and K. Seyffarth. The k-Dominating Graph. *Graphs and Combinatorics*, 30(3):609–617, 2014.

[23] M. Habib and J. Stacho. Reduced clique graphs of chordal graphs. *Eur. J. Comb.*, 33(5):712–735, 2012.

[24] A. Haddadan, T. Ito, A. E. Mouawad, N. Nishimura, H. Ono, A. Suzuki, and Y. Tebbal. The complexity of dominating set reconfiguration. *CoRR*, abs/1503.00833, 2015.

[25] R. A. Hearn and E. D. Demaine. The Nondeterministic Constraint Logic Model of Computation: Reductions and Applications. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, pages 401–413, 2002.

[26] W. Hsu and T. Ma. Fast and Simple Algorithms for Recognizing Chordal Comparability Graphs and Interval Graphs. *SIAM J. Comput.*, 28(3):1004–1020, 1999.

[27] T. C. Hu. Letter to the EditorThe Maximum Capacity Route Problem. *Operations Research*, 9(6):898–900, 1961.

[28] F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier. Fixed-Parameter Algorithms for Cluster Vertex Deletion. *Theory Comput. Syst.*, 47(1):196–217, 2010.

[29] 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. *Theor. Comput. Sci.*, 412(12-14):1054–1065, 2011.

[30] T. Ito, M. Kaminski, and E. D. Demaine. Reconfiguration of List Edge-Colorings in a Graph. In *Algorithms and Data Structures, 11th International Symposium, WADS 2009, Banff, Canada, August 21-23, 2009. Proceedings*, pages 375–386, 2009.

[31] T. Ito, K. Kawamura, H. Ono, and X. Zhou. Reconfiguration of list L(2,1)-labelings in a graph. *Theor. Comput. Sci.*, 544:84–97, 2014.

[32] T. Ito, H. Ono, and Y. Otachi. Reconfiguration of Cliques in a Graph. In *Theory and Applications of Models of Computation - 12th Annual Conference, TAMC 2015, Singapore, May 18-20, 2015, Proceedings*, pages 212–223, 2015.

[33] D. S. Johnson. The NP-Completeness Column: An Ongoing Guide. *J. Algorithms*, 6(3):434–451, 1985.

[34] M. Kaminski, P. Medvedev, and M. Milanic. Complexity of independent set reconfigurability problems. *Theor. Comput. Sci.*, 439:9–15, 2012.

[35] N. Korte and R. H. Möhring. An Incremental Linear-Time Algorithm for Recognizing Interval Graphs. *SIAM J. Comput.*, 18(1):68–81, 1989.

[36] J. M. Lewis and M. Yannakakis. The Node-Deletion Problem for Hereditary Properties is NP-Complete. *J. Comput. Syst. Sci.*, 20(2):219–230, 1980.

[37] G. MacGillivray and M. H. Siggers. On the complexity of H-colouring planar graphs. *Discrete Mathematics*, 309(18):5729–5738, 2009.

[38] A. E. Mouawad. *On Reconfiguration Problems: Structure and Tractability.* PhD thesis, 2015.

[39] A. E. Mouawad, N. Nishimura, V. Pathak, and V. Raman. Shortest reconfiguration paths in the solution space of Boolean formulas. *CoRR*, abs/1404.3801, 2014.

[40] A. E. Mouawad, N. Nishimura, V. Raman, N. Simjour, and A. Suzuki. On the Parameterized Complexity of Reconfiguration Problems. In G. Gutin and S. Szeider, editors, *Parameterized and Exact Computation*, volume 8246 of *Lecture Notes in Computer Science*, pages 281–294. Springer International Publishing, 2013.

[41] A. E. Mouawad, N. Nishimura, V. Raman, and M. Wrochna. Reconfiguration over Tree Decompositions. In *Parameterized and Exact Computation - 9th International Symposium, IPEC 2014, Wroclaw, Poland, September 10-12, 2014. Revised Selected Papers*, pages 246–257, 2014.

[42] C. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[43] P. Pardalos and J. Xue. The maximum clique problem. *Journal of Global Optimization*, 4(3):301–328, 1994.

[44] G. Ramalingam and C. P. Rangan. A Unified Approach to Domination Problems on Interval Graphs. *Inf. Process. Lett.*, 27(5):271–274, 1988.

[45] K. W. Schwerdtfeger. A Computational Trichotomy for Connectivity of Boolean Satisfiability. *CoRR*, abs/1312.4524, 2013.

[46] J. P. Spinrad. *Efficient graph representations.* Fields Institute monographs. American Mathematical Society, Providence, RI, 2003.

[47] L. J. Stockmeyer and V. V. Vazirani. NP-Completeness of Some Generalizations of the Maximum Matching Problem. *Inf. Process. Lett.*, 15(1):14–19, 1982.

[48] A. Suzuki, A. E. Mouawad, and N. Nishimura. Reconfiguration of Dominating Sets. In *Computing and Combinatorics - 20th International Conference, COCOON 2014, Atlanta, GA, USA, August 4-6, 2014. Proceedings*, pages 405–416, 2014.

[49] M. Wrochna. Reconfiguration in bounded bandwidth and treedepth. *CoRR*, abs/1405.0847, 2014.