Doctoral Dissertation

Independent Set Reconfiguration and Related Problems for Some Restricted Graphs

Hoang Anh Duc

Supervisor: Ryuhei Uehara

School of Information Science
Japan Advanced Institute of Science and Technology

June 2018

Abstract

For the last decade, a collection of combinatorial problems called *reconfiguration problems* has been studied extensively. Roughly speaking, a reconfiguration problem is specified in terms of a given collection of *configurations* and a *reconfiguration rule* that describes how to transform one configuration into another. Several real-world situations involving movement and change can be modeled as reconfiguration problems. As an example, consider a network that delivers electricity from suppliers to consumers. In such a network, it may happen that some devices at a power station S is broken and one need to temporarily shut down S for replacing these broken devices with the new ones. Before shutting down S, one may need to reroute the transmission lines that go through S to some other power station in order to maintain the availability of the network. A technician may wonder which station he/she needs to pick for replacing S such that the network remains active and, for saving resources when S becomes active again, the chosen station should be as near S as possible. Such a situation can be modeled as a PATH RECONFIGURATION problem, where each configuration is a path (transmission line) from the main supplier to customers, and the rule is to change a node (power station) such that the network remains connected (active). Another real-world situation where reconfiguration problems arise is the motion planning of moving objects. For instance, in a 3D printer model where multiple printing heads have been used, one need to plan the printing paths (which the heads will follow) to avoid collisions and other unwanted interactions, as well as making the distance traveled by each head as small as possible. Another situation involves multiple robots moving in an environment and one need to plan their movements such that they can avoid obstacles and each other. Such problems can be modeled as different variants of the TOKEN RECONFIGURATION problem on graphs. A classic variant of TOKEN RECONFIGURATION is the so-called 15-PUZZLE – a research topic since 1879. A configuration of 15-PUZZLE consists of 15 tokens labeled $1, 2, \dots, 15$, placed on a 4×4 grid. The rule is that a token can only be slid to an unoccupied adjacent vertex. The 15-PUZZLE problem asks whether one can transform one configuration into another. 15-PUZZLE and its generalized versions can be used as models for the MULTI-ROBOT PATH PLANNING problem. For an overview on both theoretical and practical perspectives of reconfiguration problems, the readers are referred to the surveys by van den Heuvel [*Surveys in Combinatorics 2013*, 127–160, 2013] and Nishimura [*Algorithms*, 11:4, 52, 2018].

Among several reconfiguration problems, the reconfiguration variants of INDEPENDENT SET are of particular interest. In such variants, an independent set (a set of pairwise non-adjacent vertices of a graph) is often viewed as a set of tokens placing on vertices of the input graph. In this viewpoint, INDEPENDENT SET RECONFIGURATION can be seen as a restricted version of the TOKEN RECONFIGURATION problem where distinct unlabeled tokens are placed on the vertices of a graph and no two tokens are adjacent. Among different reconfiguration rules, the following three models have attracted the attention of many theoretical computer scientists: *Token Sliding* (TS), *Token Jumping* (TJ), and *Token Addition and Removal* (TAR). A TS-step involves moving a token to one of its adjacent vertices. A TJ-step involves moving a token to any other vertex (not necessarily in its neighbors) of the graph. A TAR-step involves either adding or removing a token such that the number of remaining tokens is at least some threshold k. Typically, we are interested in determining whether one can transform an independent set I into another independent set J using TS/TJ/TAR rule such that each intermediate result is also an independent set. For all three rules, the problem is PSPACE-complete even for planar graphs of maximum degree 3 and bounded bandwidth/treewidth/pathwidth/cliquewidth. This raises an open question on whether there exist efficient algorithms for solving the problem (under TS/TJ/TAR) when the bandwidth/treewidth/pathwidth/cliquewidth of the input graph is bounded by some practical (small) constant. Interestingly, when comparing the three rules, TJ and TAR are equivalent, in the sense that for any sequence of p TJ-steps between two independent sets I and J of size k, there is also a sequence of $2p$ TAR-steps between them whose number of tokens in each member is either k or $k - 1$, and vice versa. The TS model seems to be more "restricted", in the sense that any sequence of TS-steps can be seen as a sequence of TJ-steps. However, the reverse direction does not hold. This motivates our study for the TS rule. As a result, in this thesis, we made a significant contribution to the computational complexity of INDEPENDENT SET RECONFIGURATION under TS rule via designing polynomial-time algorithms for solving the problem for different restricted graphs, namely trees, and cactus graphs (whose treewidth is at most 2). As consequences of our algorithms, we show that one can construct an actual sequence of TS-steps (if exists) between two given independent sets using a polynomial number of token-slides.

Key Words. polynomial-time algorithm, computational complexity, combinatorial reconfiguration, sliding token, independent set, tree, cactus graph

Acknowledgments

Foremost, this thesis would not have been completed without the great support from my supervisor—Professor Ryuhei Uehara of Japan Advanced Institute of Science and Technology (JAIST). I would like to express my deepest gratitude and appreciation to him for his continuous support of my PhD study and research, for his patience, motivation, enthusiasm, and immense knowledge. Professor Uehara has supported me not only by providing a research assistantship, but also academically and emotionally through the rough road to finish this thesis.

Several ideas presented in this thesis came from insightful discussions with Professor Yota Otachi (Kumamoto Univesity, Japan), Eli Fox-Epstein (Brown University, USA), Professor Takehiro Ito (Tohoku University, Japan) and the members of his laboratory, including Tatsuhiko Hatanaka, Haruka Mizuta, and Yuhei Moriya. I am grateful for their valuable comments and suggestions.

To the staff and students in Uehara laboratory, I am grateful for the chance to study in Japan and be a part of the lab. Thank you for welcoming me as a friend and helping to improve my basic knowledge about algorithms and graph theory.

Additionally, I would like to thank the Japan Advanced Institute of Science and Technology (JAIST) for providing the financial support that enables me to present my works at international conferences.

Lastly, I would like to dedicate this thesis to my family for their constant love, support, understanding and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we first give a brief overview on the general framework of *reconfiguration problems*—a young and growing research area in the field of theoretical computer science. Then, we introduce different *reconfiguration variants* of the INDEPENDENT SET problem—a classic NP-complete problem in computational complexity theory [1]. Finally, we briefly announce the main results of this thesis regarding the computational complexity of a natural reconfiguration variant of INDEPENDENT SET (called the SLIDING TOKEN problem) for some restricted graphs. For any notation related to graph theory that is not mentioned here, the readers are referred to [2].

1.1 Reconfiguration Problems

For the last decade, motivated by the purpose of understanding the structure of the solution space of a computational problem, the *reconfiguration problems* have been extensively studied in the field of theoretical computer science. In general, a reconfiguration problem is defined in terms of a collection of *configurations* and a *reconfiguration rule* which describes how one transforms one configuration into another. In most of the reconfiguration problems considered in the literature, an implicit assumption is that one can decide in polynomial time whether one configuration can be transformed into another via a single transformation. Typically, one may ask if there is a sequence of configurations that transforms some initial configuration A into another configuration B such that each intermediate member of the sequence is obtained from the previous one by a single transformation. Such a sequence is called a *reconfiguration sequence.* A classic example of such problems is the so-called 15-PUZZLE [3]—a research topic since 1879 (see Figure 1.1). A configuration of 15-PUZZLE consists of a placement of 15 tiles numbered $1, 2, \dots, 15$ on a 4×4 board, leaving one empty square. One can also consider a more general setting in which a configuration consists of a placement of $n^2 - 1$ tiles on a $n \times n$ board (the (n^2-1)-PUZZLE). The allowed (reconfiguration) rule is that a tile can move to the empty square if it is on the left/right/top/bottom of that square. Given a collection of configurations and the reconfiguration rule, the $(n^2 - 1)$-PUZZLE (15-PUZZLE) problem asks whether there exists a sequence of configurations that transforms one configuration into another, such that each intermediate member of the sequence is obtained from the previous one by a single tile-move.

Many real-world problems also fall into the category of reconfiguration. For example, in a large network, the following situation may happen: some nodes of the network

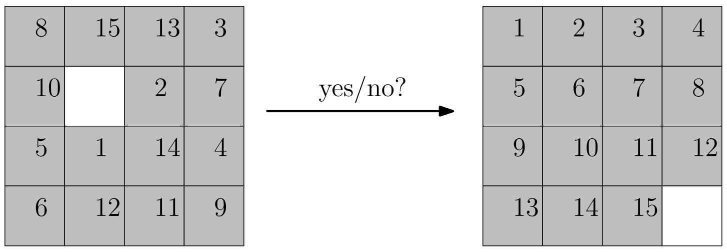


Figure 1.1: An instance of 15-PUZZLE.

have been broken and one may need to take them down for modification or replacement. However, taking down a node may cause the system to crash. Therefore, one need to figure out a way of taking down some nodes and maintaining the network's quality at the same time. Clearly, this problem can be modeled as a reconfiguration problem. A typical problem of this type is the FREQUENCY ASSIGNMENT problem, in which we aim to assign frequencies to users of a wireless network in order to minimize the interference between them and managing to use as less range of frequencies as possible during the process. This problem can be modeled as a VERTEX-COLORING problem on graphs, where each frequency corresponds to a color. In a real-world situation, one may need to modify an assignment because of some technical issues while maintaining the network's activities. This is where a VERTEX-COLORING RECONFIGURATION model comes in handy. For a given input graph G, a configuration of VERTEX-COLORING RECONFIGURATION is a coloring of vertices of G (which indeed corresponds to a frequency assignment) such that no two adjacent vertices share the same color, and a reconfiguration step involves changing one vertex-color at a time. Another example is the following generalized version of 15-PUZZLE called the PEBBLE MOTION problem: Given a graph G on n vertices with $k < n$ pebbles (tokens) numbered $1, 2, \dots, k$ on distinct vertices. A pebble can move to an adjacent unoccupied vertex. The question is to decide if one pebble configuration is reachable from another configuration via a sequence of pebble-moves. It is well-known that PEBBLE MOTION can be decided efficiently [4]. Another interesting question is to decide whether one can find a shortest sequence of pebble-moves between two given configurations for different input graphs (in general, the problem is NP-complete [5, 6]). The PEBBLE MOTION problem is strongly related to the problem of MULTI-ROBOT PATH PLANNING where multiple robots are moving in an environment and they must avoid obstacles and each other (e.g., see [7]). Several other practical applications of reconfiguration are reviewed in [8, Section 5].

Over the last decade, researchers have approached reconfiguration problems from different perspectives. The study of reconfiguration problems is, in some sense, the characterization of the so-called *reconfiguration graph*—a graph whose vertices are configurations, and for any pair A, B of vertices (configurations), B is *adjacent* to A if it can be obtained from A via a single transformation. Obviously, it is possible that the transformation goes one way only (i.e., the reconfiguration graph is directed). However, in most problems that have been considered so far in the literature, a typical assumption is that we can go back and forth between configurations (i.e., the reconfiguration graph is undirected). In general, the size of a reconfiguration graph is exponentially large (e.g., the reconfiguration graph for 15-PUZZLE contains 16! vertices). This makes the study of reconfiguration problems quite challenging. Using the language of reconfiguration graphs, one can formulate several problems, some of which are:

- REACHABILITY: Decide if two given vertices of a reconfiguration graph is in the same component. The (n^2-1)-PUZZLE can be categorized as a REACHABILITY question. It is well-known that (n^2-1)-PUZZLE can be decided efficiently [3, 9].
- CONNECTIVITY: Given a reconfiguration graph, decide if it is connected. In 1974, Wilson [10] gave a complete characterization of the connectedness of the reconfiguration graph of (n^2-1)-PUZZLE, which then implies that the CONNECTIVITY question for this reconfiguration graph can be decided in polynomial time.
- SHORTEST RECONFIGURATION SEQUENCE: Decide if it is possible to find a shortest path connecting two vertices in the same component of a reconfiguration graph. Note that this problem is different from the SHORTEST PATH problem, which asks for finding a shortest path between two given vertices in the same component of a graph. The problem of finding a shortest sequence of tile-moves between two vertices (configurations) in the same component of the reconfiguration graph of (n^2-1)-PUZZLE is NP-complete [5, 6].
- BOUNDED RECONFIGURATION SEQUENCE: Decide if there is a path of length at most ℓ between two vertices of a reconfiguration graph, for some given integer ℓ. Goldreich [11] showed that BOUNDED RECONFIGURATION SEQUENCE is NP-complete for the reconfiguration graph of (n^2-1)-PUZZLE.
- DIAMETER: Decide if the diameter (i.e., the *maximum* distance between any two vertices of a graph) of the reconfiguration graph is bounded. For any two vertices in the same component of the reconfiguration graph of (n^2-1)-PUZZLE, it is well-known that one can find a path of length $O(k^3)$ connecting them [4], where $k = n^2$ is the number of squares in the given $n \times n$ board.

In a reconfiguration problem, a configuration is usually given as a *feasible solution* of some computational problem, and a transformation rule can be seen as a simple way of *modifying* a solution without changing its *feasibility.* In that manner, for a *source* problem P, one can define various *reconfiguration variants* of P based on different reconfiguration rules. Recently, many reconfiguration variants of several well-known problems, including SATISFIABILITY, INDEPENDENT SET, VERTEX COVER, DOMINATING SET, CLIQUE, MATCHING, MATROID BASES, SHORTEST PATH, (LIST) VERTEX-COLORING, (LIST) EDGE-COLORING, etc., have been extensively studied. For notational convention, we shall refer to a reconfiguration variant of a problem P with reconfiguration rule R as the P RECONFIGURATION *problem under R rule.* In the literature, the source problem P is usually a graph problem, and an important task is to draw a line between the tractability/intractablility of different reconfiguration variants of P via exploring their computational complexity for several different graph classes. A particular example of problems in this research direction is the extensive study of reconfiguration variants of INDEPENDENT SET (see Section 1.2 and [12, Section 4]). Another task is to characterize the similarity and difference between the source problem and its reconfiguration variants. For example, many reconfiguration problems are PSPACE-complete while their source problems are NP-complete [13]. Such a pattern does not always hold. For instance, 3-COLORING is NP-complete [1], however, 3-COLORING RECONFIGURATION is polynomially solvable [14]. On the other hand, Shortest Path Reconfiguration is PSPACE-complete [15] while the source problem Shortest Path is in P [16]. Another direction is to study the structural properties of reconfiguration graphs, for example, via investigating which graph can be a reconfiguration graph. Particularly, several structural properties of the reconfiguration graphs of k-Coloring (e.g., see [17]) and Dominating Set (e.g., see [18]) have been characterized. For further details and discussion on this research area, the readers are referred to the surveys [8, 12].

1.2 Reconfigurability of Independent Set

One of the most basic problem in computational complexity theory is the Independent Set problem. Given a graph $G = (V, E)$ and an integer $k \leq |V|$, the Independent Set problem asks whether there exists an *independent set* of G of size at least k, that is, a vertex-subset V' of G such that no two vertices of V' are adjacent and $|V'| \geq k$. It is well-known that Independent Set is NP-complete [1]. Over the years, countless problems involving Independent Set have been studying. The study of the reconfigurability of independent set seems to begin with a dynamic version of Independent Set called the Sliding Token problem. This problem was first introduced by Hearn and Demaine [19] as an illustration for their *Nondeterministic Constraint Logic* (NCL) model—a powerful tool for showing PSPACE-completeness of several computational problems. For a given graph G and an independent set I of G, imagine that a token is placed at each vertex of I. For a graph G and two independent sets I, J, the Sliding Token problem asks if there exists a (reconfiguration) sequence $\mathcal{S} = \langle I_1, I_2, \ldots, I_\ell \rangle$ between the *initial* independent set $I = I_1$ and the *target* independent set $J = I_\ell$ such that (i) each member of $\mathcal{S}$ is an independent set of G, and (ii) for $i \in \{2, 3, \ldots, \ell\}$, I_i is obtained from I_{i-1} by sliding a single token along an edge of G. Figure 1.2 illustrates a yes-instance (Figure 1.2(a)) and a no-instance (Figure 1.2(b)) of Sliding Token where the input graph G is a 3×3 grid.

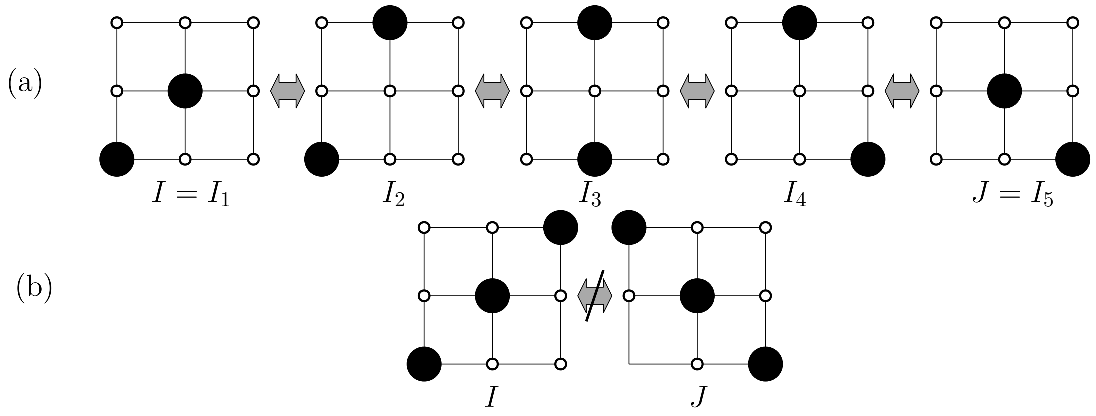


Figure 1.2: (a) A yes-instance and (b) A no-instance of Sliding Token where the input graph G is a 3×3 grid. Tokens are depicted with black circles.

In the language of reconfiguration, each configuration of Sliding Token is an independent set, and the reconfiguration rule is that one can only slide a token from one vertex to one if its neighbors along an edge of the given graph. In the literature, we refer to this reconfiguration rule as the *Token Sliding* (TS) rule. In a more general setting, one can allow a token to move to any vertex of the graph instead of just its neighbors. Such a reconfiguration rule is called the *Token Jumping* (TJ) rule and was first studied by Kamiński et al. [20]. A simple observation is that in both TS and TJ models, if there exists a reconfiguration sequence between two given independent sets then they must be of the same size. In a recent research, de Berg et al. [21] considered the *Multiple Token Jump* (p-MTJ) rule in which at most p tokens are allowed to move simultaneously. Clearly, TJ is a special case of p-MTJ when $p = 1$. For p-MTJ, the problem of interest is to determine the smallest value of p such that there is a reconfiguration sequence under the p-MTJ rule between any two independent sets of size k. In 2011, Ito et al. [13] introduced a different reconfiguration rule called *Token Addition and Removal* (TAR). In a TAR step, one can either add or remove a single token as long as the number of remaining tokens is at least some threshold k. For convenience, we shall refer to the reconfiguration sequence under TS (resp. TJ, TAR) rule as the TS-*sequence* (resp. TJ-*sequence*, k-TAR-*sequence*). Kamiński et al. [20] showed that the TJ and TAR rules are indeed equivalent, in the sense that for any TJ-sequence of ℓ TJ steps between two independent sets I, J of size k, there also exists a $(k-1)$-TAR-sequence of 2ℓ TAR steps between them, and vice versa. Regarding TS and TJ, observe that since a maximum independent set is also a dominating set (i.e., a vertex-subset D such that every vertex not in D is adjacent to at least one member of D), a TS-sequence between two maximum independent sets I, J is also a TJ-sequence between them, and vice versa [22]. We note that these reconfiguration rules (TS, TJ, p-MTJ, TAR) can also be defined for other vertex-subset problems such as Vertex Cover, Clique, Dominating Set, and so on.

Hearn and Demaine [19] showed that Sliding Token is PSPACE-complete even for planar graphs of maximum degree 3 via a reduction from a variant of their NCL model. This result was not explicitly stated in [19], but was observed later by Bonsma and Cereceda [23]. Since TAR and TJ are equivalent and their proof uses only maximum independent sets, the result can be applied for both TJ and TAR rules. Ito et al. [13] claimed that Independent Set Reconfiguration under TAR rule is PSPACE-complete by extending a reduction from the 3-SAT problem to the Independent Set problem. As before, the result can be applied to TJ and TS too. Kamiński et al. [20] showed that under TS, TJ, and TAR rules, Independent Set Reconfiguration remains PSPACE-complete for perfect graphs. The proof was done via a reduction from another PSPACE-complete reconfiguration problem—the Shortest Path Reconfiguration problem. In Shortest Path Reconfiguration, the configurations are all shortest paths between two given vertices s, t of a graph, and a reconfiguration step involves changing a single vertex of a shortest path. Wrochna [24] proved that the H-Word Reconfiguration problem is PSPACE-complete and used it as a basis to show that Independent Set Reconfiguration under TJ rule is PSPACE-complete for bounded bandwidth graphs, even with additional assumption that all independent sets are maximum. In H-Word Reconfiguration, an *H-word* is a string formed by a given alphabet and a binary relation between symbols such that any two consecutive symbols in the string are in the relation. A reconfiguration step consists of changing a single symbol of an H-word. As before, the result also holds for TS and TAR models. Recall that the *bandwidth* of a graph $G = (V, E)$ is defined as $\min_f \max_{uv \in E} |f(u) - f(v)|$, where $f : V \to \mathbb{Z}$ represents a way of labeling vertices of G with distinct integers. Since a graph of bandwidth b has pathwidth and treewidth (measuring how path-like and tree-like a graph is) at most b, these results hold for graphs of bounded pathwidth and graphs of bounded treewidth. Moreover, since any graph of treewidth at most c is also of cliquewidth (a notion generalizing treewidth) at most $2^{c+1} + 1$ (see [25]), these results can be applied for graphs of bounded cliquewidth. Combining the techniques in [19] and [24], van der Zanden [26] strengthened the known results by showing the PSPACE-completeness of INDEPENDENT SET RECONFIGURATION under TS, TJ, and TAR rules for planar graphs of maximum degree 3 and bounded bandwidth/pathwidth/treewidth/cliquewidth. Very recently, Lokshtanov and Mouawad [27] settled the complexity of INDEPENDENT SET RECONFIGURATION for bipartite graphs by showing that the problem is PSPACE-complete under TS rule, and NP-complete under TJ and TAR rules.

On the positive side, polynomial-time algorithms have been designed for INDEPENDENT SET RECONFIGURATION on several graph classes. Different polynomial-time algorithms have been developed for solving INDEPENDENT SET RECONFIGURATION for even-hole-free graphs (i.e., graphs contain no induced cycles whose number of vertices is even) under TJ and TAR rules [20, 28–30]. This result implies that INDEPENDENT SET RECONFIGURATION under TJ and TAR can be solved efficiently for several subclasses of even-hole-free graphs, some of which are chordal graphs, interval graphs, and trees. Interestingly, to the best of our knowledge, the complexity of both INDEPENDENT SET and INDEPENDENT SET RECONFIGURATION under TS rule for even-hole-free graphs remains open. For cographs (P_4-free graphs), polynomial-time algorithms have been developed for TS model by Kamiński et al. [20] and for TJ and TAR models by Bonsma [30]. Bonsma et al. [22] claimed that under all TS, TJ, and TAR, INDEPENDENT SET RECONFIGURATION can be solved in polynomial time when the input graph is a claw-free graph. Fox-Epstein et al. [31] developed polynomial-time algorithms for solving INDEPENDENT SET RECONFIGURATION under TS rule for bipartite permutation graphs and bipartite distance-hereditary graphs. Recently, Bonamy and Bousquet [32] claimed that one can solve INDEPENDENT SET RECONFIGURATION under TS rule for interval graphs in polynomial time.

So far, we have seen several results regarding the REACHABILITY problem for INDEPENDENT SET RECONFIGURATION (i.e., asking whether there is a reconfiguration sequence between two given independent sets). Other problems, including CONNECTIVITY, SHORTEST RECONFIGURATION SEQUENCE, BOUNDED RECONFIGURATION SEQUENCE, DIAMETER, etc., have also been investigated. It is well-known that the BOUNDED RECONFIGURATION SEQUENCE problem for INDEPENDENT SET RECONFIGURATION under all TS, TJ and TAR rules is NP-hard, even when the input graph G is a perfect graph, and the given length ℓ is polynomial in $|V(G)|$ [33]. Kamiński et al. [20] showed that in an even-hole-free graph, any two independent sets of the same size k are TJ- (TAR-) *reconfigurable*, i.e., there is a TJ- ($(k-1)$-TAR-) sequence between them. Moreover, one can construct in linear time a shortest TJ- ($(k-1)$-TAR) sequence between them. For cographs, shortest TS-sequences, if exist, can be found in linear time [20], while under TJ and TAR models, there exists an upper bound on the diameter of the corresponding reconfiguration graph [30]. Additionally, Bonamy and Bousquet [34] designed a polynomial-time algorithm for solving the CONNECTIVITY problem under TAR rule. When the input graph is a claw-free graph G, the diameter of the corresponding reconfiguration graph (under TS, TJ, and TAR) is bounded polynomially in the number of vertices of G [22]. When the input graphs are either bipartite permutation or bipartite distance-hereditary, it follows from the algorithms of Fox-Epstein et al. [31] that the diameter of the reconfiguration graph under TS rule is bounded polynomially in the number of vertices of the input graphs. Polynomial-time algorithms have been designed for finding shortest TS-sequence in proper interval graphs, trivially perfect graphs, and caterpillars [35]. The algorithm for caterpillars (a subclass of trees) seems to be the first polynomial-time algorithm that handles the situation where a token is required to make "detour" in order to maintain the independence property of the set of tokens. An example of such a situation is illustrated in Figure 1.2(a): the token of I in the middle of the grid needs to make detour by going up to let the token on the bottom left passes through to the bottom right, and then moving back down again. In a recent research, Bonamy and Bousquet [32] showed that the problem of deciding if the reconfiguration graph of INDEPENDENT SET under TS model is connected when the input graph is a split graph is co-NP-hard. Very recently, Fatehi et al. [36] addressed the question of finding which graph can be a reconfiguration graph of INDEPENDENT SET under a modified version of the TAR rule where one can add or remove a single token at each reconfiguration step as long as the number of remaining tokens is *at most* some threshold k. In [36], the authors computed the reconfiguration graphs of INDEPENDENT SET under the described rule for several different input graphs. To the best of our knowledge, the reconfiguration graphs for k-COLORING and DOMINATING SET have been well-studied. However, up to present, for INDEPENDENT SET, there have been very limited results. de Berg et al. [21] considered INDEPENDENT SET RECONFIGURATION under the p-MTJ rule and an equivalent variant of the TAR rule where the number of tokens never exceeds the size k of the initial independent set I and each independent set in the reconfiguration sequence contains at most p fewer tokens than I. The question is to determine the minimum value of p (called the *reconfiguration threshold*) such that there always exists a reconfiguration sequence under these rules between two given independent sets I, J of size k. In [21], the authors estimated lower and upper bounds on the value of p in term of several graph parameters, including the size of a minimum vertex cover, the size of a minimum feedback vertex set, and pathwidth. Recall that a *vertex cover* of a graph G is a vertex-subset V' such that every edge of G is incident with a vertex in V'; and a *feedback vertex set* of a graph G is a vertex-subset V' such that the graph $G - V'$ obtained by removing all vertices in V' contains no cycle. They also identified a structure that causes these reconfiguration thresholds to be large.

Recently, the *parameterized complexity* of INDEPENDENT SET RECONFIGURATION have been considered in the literature. Parameterized complexity provides a framework for classifying the intractable problems by investigating their "tractability/intractability" when some problems' parameters are given as a part of the input. For more information on parameterized complexity, see the textbooks [37, 38]. For an overview on the achieved results and research directions on the parameterized complexity of INDEPENDENT SET RECONFIGURATION and related problems, the readers are referred to [12, Section 5].

1.3 Our Results

In this thesis, we study SLIDING TOKEN (i.e., INDEPENDENT SET RECONFIGURATION under TS rule) and related problems for different graph classes. There are several reasons that motivate our study. First of all, SLIDING TOKEN (and other variants of INDEPENDENT SET RECONFIGURATION) has been used as a basis to prove the PSPACE-completeness of several other problems, for example, the k-COLORING RECONFIGURATION problem for $k \geq 4$ [39], or the CLIQUE RECONFIGURATION problem for perfect

Graph	REACHABILITY	DIAMETER	Reference
trees	$O(n)$	$O(n^2)$	[41–43]
cactus graphs	$O(n^2)$	$O(n^2)$	[44]

Table 1.1: Our results for SLIDING TOKEN. Here n denotes the number of vertices of the corresponding graph.

graphs [40], and so on. Another reason is that, in some sense, the TS rule is more "restricted" than the TJ and TAR rules. As an example, consider the NO-instance of SLIDING TOKEN given in Figure 1.2(b). If one uses TJ or TAR rule instead of TS rule, the instance immediately becomes a YES-instance. Moreover, under TS rule, one may need to deal with the situation where a token needs to make "detour" in order to maintain the independence property of the set of tokens. This makes the problems under TS rule more challenging. Another motivation for our study is that the SLIDING TOKEN problem is PSPACE-complete for graphs of bounded bandwidth/pathwidth/treewidth/cliquewidth [24]. This raises an open question on whether efficient algorithms exist for SLIDING TOKEN (and other variants of INDEPENDENT SET RECONFIGURATION) when the bandwidth/pathwidth/treewidth/cliquewidth of the input graph is bounded by some small constant. Our achieved results partially answer this question.

In this thesis, we made some significant contribution to the structural understanding of the computational complexity of SLIDING TOKEN by showing that the problem can be solved efficiently for some restricted graphs (see Table 1.1), namely trees, and cactus graphs (see Section 2.1 for definitions of these graphs). We note that INDEPENDENT SET RECONFIGURATION under TJ and TAR rules can be solved in polynomial-time for trees [20] (a subclass of even-hole-free graphs) and cactus graphs [28]. As consequences of our algorithms, we show that one can construct an actual TS-sequence (if exists) between two given independent sets using a polynomial number of token-slides. We remark that our constructed TS-sequence is *not* necessarily of shortest length. Here, the *length* of a TS-sequence $\mathcal{S}$ is often defined as the number of token-slides described in $\mathcal{S}$.

The main idea of our algorithms is the characterization of all structures that forbid the existence of a TS-sequence between any two given independent sets. Such a structure is called a *forbidden structure*. A trivial forbidden structure is the *sizes* of the given independent sets. More precisely, if $|I| \neq |J|$ then there is no TS-sequence between them. Moreover, we claim that all forbidden structures can be found in polynomial time, and there must be some TS-sequence between any two independent sets when such forbidden structures do not exist. We note that similar ideas have been applied for designing polynomial-time algorithms for solving 3-COLORING RECONFIGURATION [14] and DEGREE-CONSTRAINED SUBGRAPH RECONFIGURATION [45].

The rest of this thesis is organized as follows.

- In **Chapter** 2, we define some basic notation (Section 2.1) and prove several useful observations for tackling SLIDING TOKEN (Section 2.2).
- In **Chapters** 3 and 4, we present the main results of this thesis.
  - In **Chapter** 3, we present a linear-time algorithm for solving SLIDING TOKEN for trees. Moreover, given a YES-instance (T, I, J) of this problem, we describe

how to transform I to J using $O(n^2)$ token-slides, where n is the number of vertices of T.

  - In **Chapter** 4, we present a $O(n^2)$-time algorithm for solving SLIDING TOKEN for cactus graphs. Moreover, given a YES-instance (G, I, J) of this problem, we describe how to transform I to J using $O(n^2)$ token-slides. Here n is the number of vertices of G.

- In **Chapter** 5, we summary the contents of this thesis and present some interesting open problems for future study.

Chapter 2

Preliminaries

2.1 Basic Terms and Notation

In this section, we define some basic terms and notation. Any graph notation that is not mentioned here can be found in the textbook [2].

Let G be a (simple, undirected) graph with vertex set $V(G)$ and edge set $E(G)$. We always use n and m for indicating $|V(G)|$ and $|E(G)|$, respectively. For a vertex $v \in V(G)$, we denote by $N_G(v)$ the set $\{u \in V(G) : uv \in E(G)\}$ of *neighbors* of v, and by $N_G[v]$ the set $N_G(v) \cup \{v\}$ of *closed neighbors* of v. The *degree* of v in G, denoted by $\deg_G(v)$, is the size of its neighbors. For a vertex set $W \subseteq V(G)$, we define $N_G[W] = \bigcup_{v \in W} N_G[v]$. For $u, v \in V(G)$, we denote by $\mathsf{dist}_G(u, v)$ the length of a shortest uv-path in G.

A vertex $v \in V(G)$ is a *cut vertex* of G if $G - v$ is not connected; otherwise, v is a *non-cut vertex*. For a graph G, a *block* of G is a maximal connected subgraph with no cut vertex. A graph G is a *tree* if it is connected and contains no cycle (i.e., every block of G is an edge). A graph G is a *cactus graph* (or *cactus* for short) if every block of G is either an edge or a cycle. We note that if G is either a tree, or a cactus graph, then so is any induced subgraph of G. This property will be used implicitly in several statements of this thesis.

For a vertex subset $W \subseteq V(G)$, we write $W \cap G$ and $W - G$ to indicate the sets $W \cap V(G)$ and $W \setminus V(G)$, respectively. The subgraph of G induced by W is denoted by $G[W]$. We write $G - W$ to indicate the subgraph $G[V(G) \setminus W]$. Similarly, for an induced subgraph H of G, $G - H$ indicates the subgraph $G - V(H)$, and we say that $G - H$ is the graph *obtained from G by removing H*.

Let I, J be two independent sets of a graph G. Imagine that a token is placed at each vertex of I. We sometimes identify a token and the vertex where it is placed and simply say "a token in an independent set I" instead of "a token placed on a vertex in an independent set I", and "sliding u to v" instead of "sliding a token on u to v".

For two independent sets I, J of a graph G, if there exists a TS-sequence between I and J, we write $I \overset{G}{\leftrightsquigarrow} J$, and say that $\mathcal{S}$ *reconfigures* I to J in G. For a TS-sequence $\mathcal{S}$, we say that $\mathcal{S}$ *slides* (or *moves*) the token t on u to some vertex v if after performing $\mathcal{S}$, t is placed at v. We write $I \in \mathcal{S}$ if I is a member of $\mathcal{S}$. The *length* of $\mathcal{S}$ is simply the number of independent sets in $\mathcal{S}$ minus one (i.e., it is the number of token-slides performed in $\mathcal{S}$). We say that two TS-sequences $\mathcal{S}$ and $\mathcal{S}'$ can be performed *independently* if $\mathcal{S}$ does not move any token used in $\mathcal{S}'$, and vice versa. In other words, performing $\mathcal{S}$ and then $\mathcal{S}'$ yields the same result as performing $\mathcal{S}'$ and then $\mathcal{S}$. For two TS-sequences $\mathcal{S}_1 = \langle I^1_1, I^1_2, \dots, I^1_\ell \rangle$ and

$\mathcal{S}_2 = \langle I_1^2, I_2^2, \ldots, I_{\ell'}^2 \rangle$, if $I_\ell^1 = I_1^2$ then one can *append* $\mathcal{S}_2$ *to* $\mathcal{S}_1$ to form a new TS-sequence $\mathcal{S} = \langle I_1^1, I_2^1, \ldots, I_\ell^1, I_2^2, \ldots, I_{\ell'}^2 \rangle$.

For a vertex subset $W \subseteq V(G)$, we say that the token t placed at $u \in I \cap W$ is (G, I, W)-*confined* if no TS-sequence in G slides t to a vertex not in W (e.g., the tokens t_3 and t_5 in Figure 2.1). In other words, t can be slid only along edges of $G[W]$. In particular, if $W = \{u\}$, we say that t is (G, I)-*rigid* (e.g., the tokens t_7 and t_8 in Figure 2.1). We say that t is (G, I)-*movable* if it is not (G, I)-rigid. We denote by $\mathsf{R}(G, I)$ the set of all vertices where (G, I)-rigid tokens are placed. Deciding if a token is (G, I)-rigid is PSPACE-complete even when G is a planar graph of maximum degree 3 [19].

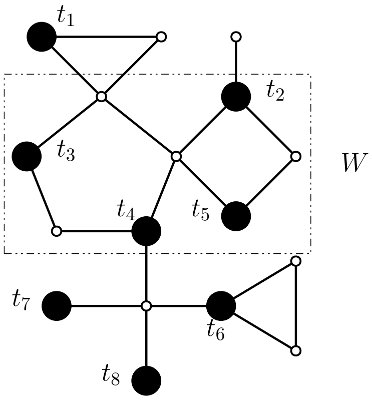


Figure 2.1: The tokens t_3 and t_5 are (G, I, W)-confined, while t_2 and t_4 are not. The tokens t_7 and t_8 are (G, I)-rigid, and all other tokens are (G, I)-movable.

For an induced subgraph H of G, we say that H is (G, I)-*confined* if $I \cap H$ is a maximum independent set of H and every token in $I \cap H$ is $(G, I, V(H))$-confined. In particular, if H is a path (resp., a cycle), we say that it is a (G, I)-*confined path* (resp., *cycle*). For example, in Figure 2.1, the induced cycle containing the tokens t_3 and t_4 is a (G, I)-confined cycle. We denote by $\mathsf{C}(G, I)$ the set of all (G, I)-confined (induced) cycles of G, respectively. For a vertex $v \in V(H)$, we denote by G_H^v the component of G_H containing v, where G_H is obtained from G by removing all edges of H.

Let B, B' be two blocks of a graph G. We say that B is a *neighbor* of B' if $V(B) \cap V(B') \neq \emptyset$. A block B is *safe* if it has at most one cut vertex and at most one neighbor containing more than one cut vertex. For example, the blocks marked with thick edges in Figure 2.2 are safe. A vertex $v \in V(G)$ is *safe* if it is a non-cut vertex of some safe block B of G.

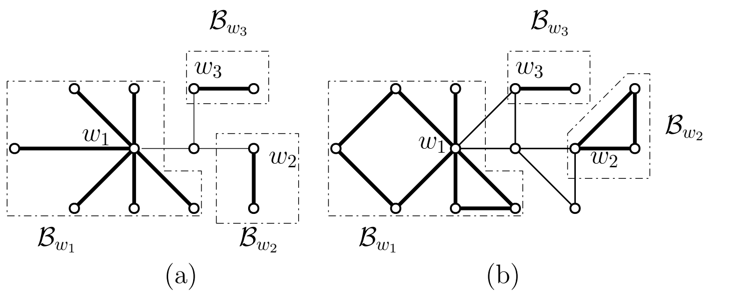


Figure 2.2: Examples of safe blocks (marked with thick edges) in (a) a tree, and (b) a cactus.

For a cut vertex w of G, denote by $\mathcal{B}_w$ the smallest subgraph of G such that $\mathcal{B}_w$ contains all safe blocks of G containing w (see Figure 2.2). $\mathcal{B}_w$ can also be viewed as a collection of safe blocks sharing the same cut vertex w. If no safe block contains w, we define $\mathcal{B}_w = \emptyset$. Observe that for two distinct cut vertices w_1, w_2, $V(\mathcal{B}_{w_1}) \cap V(\mathcal{B}_{w_2}) = \emptyset$.

2.2 Some General Observations

In this section, we prove some general observations regarding the Sliding Token problem. Throughout this section, let I be an independent set of a given graph G. The next proposition is immediate from the definition.

Proposition 2.1. *Let I be an independent set of a graph G. Let t be a token in I. Then, for every J such that $I \overset{G}{\leftrightsquigarrow} J$,*

(i) If t is (G, I, W)-confined for some $W \subseteq V(G)$ then it is also (G, J, W)-confined.

(ii) If t is (G, I)-rigid then it is also (G, J)-rigid.

In the next proposition, we describe a simple property of rigid tokens. We shall employ this property from [31] and therefore omit its proof.

Proposition 2.2 ([31])**.** *Let I be an independent set of a graph G, and let $S \subseteq I$. If for all $w \in N_G(S)$, $|N_G(w) \cap S| > 1$ then $S \subseteq \mathsf{R}(G, I)$.*

The next proposition says that if the given graph G is not connected, then one can deal with each component separately.

Proposition 2.3. *Let I, J be two given independent set of G. Assume that $G_1, \dots, G_k$ are the components of G. Then $I \overset{G}{\leftrightsquigarrow} J$ if and only if $I \cap G_i \overset{G_i}{\leftrightsquigarrow} J \cap G_i$ for $i = 1, 2, \dots, k$.*

Proof. We first show the only-if direction. Assume that $\mathcal{S} = \langle I_1, \dots, I_\ell \rangle$ is a TS-sequence in G that reconfigures $I = I_1$ to $J = I_\ell$. For any $i \in \{1, 2, \dots, k\}$ and any independent set I of G, as $I \cap G_i \subseteq I$, $I \cap G_i$ is also independent. Hence, $\mathcal{S}_i = \langle I_1 \cap G_i, \dots, I_\ell \cap G_i \rangle$ reconfigures $I \cap G_i$ to $J \cap G_i$.

Now, we show the if direction. Assume that for each $i \in \{1, 2, \dots, k\}$, there exists a TS-sequence $\mathcal{S}'_i$ in G_i that reconfigures $I \cap G_i$ to $J \cap G_i$. For any two TS-sequences $\mathcal{S}'_i$ and $\mathcal{S}'_j$ ($i, j \in \{1, 2, \dots, k\}$), if the length of $\mathcal{S}'_i$ is smaller than the length of $\mathcal{S}'_j$ then we can make them equal by appending $\langle I \cap G_i, I \cap G_i, \dots \rangle$ to the end of $\mathcal{S}'_i$. Thus, assume without loss of generality that all $\mathcal{S}'_i$ are of equal length, i.e., any $\mathcal{S}'_i$ can be written in the form $\langle I^i_1 = I \cap G_i, \dots, I^i_l = J \cap G_i \rangle$. Let I^i be an independent set of G_i. Since $G_1, G_2, \dots, G_k$ are components of G, $\bigcup_{i=1}^k I^i$ forms an independent set of G. Thus, we can extend any sequence $\mathcal{S}'_i$ ($i = 1, 2, \dots, k$) to a TS-sequence $\mathcal{S}_i$ in G as follows.

$$\mathcal{S}_i = \langle I^i_1 \cup \bigcup_{j=1}^{i-1} I^j_l \cup \bigcup_{j=i+1}^{k} I^j_1, \dots, I^i_l \cup \bigcup_{j=1}^{i-1} I^j_l \cup \bigcup_{j=i+1}^{k} I^j_1 \rangle. \tag{2.1}$$

Clearly, the sequence $\mathcal{S}$ constructed by first applying $\mathcal{S}_1$, then $\mathcal{S}_2$, and so on is the one that reconfigures I to J in G.

□

Thus, when dealing with Sliding Token, one can assume without loss of generality that the given graph is connected. Next, we claim that in certain conditions, one can “extend” or “restrict” a TS-sequence.

Proposition 2.4. *Let W be a vertex subset of a graph G. Let $\mathcal{S} = \langle I_1, I_2, \dots, I_\ell \rangle$ be a TS-sequence in G such that $W \subseteq I_i$ for every $I_i \in \mathcal{S}$ $(1 \leq i \leq \ell)$. Let $G' = G - N_G[W]$. Then $I_1 \cap G' \overset{G'}{\leftrightsquigarrow} I_\ell \cap G'$. Moreover, for every TS-sequence $\mathcal{S}' = \langle I'_1, \dots, I'_l \rangle$ in G', $I'_1 \cup W \overset{G}{\leftrightsquigarrow} I'_l \cup W$.*

Proof. Since $W \subseteq I$ for every $I \in \mathcal{S}$, the sequence $\mathcal{S}' = \langle I_1 \setminus W, \dots, I_\ell \setminus W \rangle$ clearly reconfigures $I_1 \cap G' = I_1 \setminus W$ to $I_\ell \cap G' = I_\ell \setminus W$. To see that $\mathcal{S}'$ is actually a TS-sequence in G', note that for some $i \in \{2, 3, \dots, \ell\}$ with $I_{i-1} \setminus I_i = \{u\}$ and $I_i \setminus I_{i-1} = \{v\}$, the vertices u and v are either both in $N_G[W]$ or both in $V(G) \setminus N_G[W]$.

Now, let $\mathcal{S}' = \langle I'_1, \dots, I'_l \rangle$ be a TS-sequence in G'. For every independent set I' of G', the set $I' \cup W$ forms an independent set of G. Hence, $\mathcal{S} = \langle I'_1 \cup W, \dots, I'_l \cup W \rangle$ reconfigures $I'_1 \cup W$ to $I'_l \cup W$. □

In the next proposition, we prove some characterization of a (G, I)-confined induced subgraph. Roughly speaking, the structure of a (G, I)-confined induced subgraph H guarantees that the tokens “inside” (resp., “outside”) of H cannot be moved “out” (resp., “in”). Notice that if $I \cap H$ is a maximal independent set of H (instead of a maximum one), it could happen that some token “outside” of H can be moved “in”, even when no token “inside” of H moves “out.”

Proposition 2.5. *Let I be an independent set of a graph G, and let H be an induced subgraph of G. Then the following conditions are equivalent.*

(i) H is (G, I)-confined.

(ii) For every independent set J satisfying $I \overset{G}{\leftrightsquigarrow} J$, the set $J \cap H$ is a maximum independent set of H.

(iii) The set $I \cap H$ is a maximum independent set of H and for every J satisfying $I \overset{G}{\leftrightsquigarrow} J$, the token t_x placed at $x \in J \cap H$ is $(G_H^x, J \cap G_H^x)$-rigid.

Proof. We show that (i) ⇔ (ii) and (ii) ⇔ (iii).

- **(i) ⇔ (ii).** It follows immediately from the definition that (i) ⇒ (ii). We show that (ii) ⇒ (i). Assume that (ii) holds. Since for every J with $I \overset{G}{\leftrightsquigarrow} J$, the set $J \cap H$ is always a maximum independent set of H, no token can be slid from a vertex in H to a vertex in $G - H$, and vice versa. Thus, a token placed at some vertex in $I \cap H$ can only be slid along edges of H, i.e., it is $(G, I, V(H))$-confined. Additionally, as I is reconfigurable to itself, $I \cap H$ is a maximum independent set of H. Thus, (i) holds.

- **(ii) ⇔ (iii).** We first show (ii) ⇒ (iii). Assume that (ii) holds. It follows immediately that $I \cap H$ is a maximum independent set of H. Suppose that there exist an independent set J with $I \overset{G}{\leftrightsquigarrow} J$ and a vertex $x \in J \cap H$ such that the token t_x placed at x is $(G_H^x, J \cap G_H^x)$-movable. Let $\mathcal{S} = \langle I = I_1, \dots, I_\ell = J \rangle$ be a TS-sequence in G that reconfigures I to J. Let $\mathcal{S}' = \langle I'_1 = J \cap G_H^x, I'_2, \dots, I'_k \rangle$ be a TS-sequence in G_H^x such that $\mathcal{S}'$ slides t_x to a vertex $y \in N_{G_H^x}(x)$. By definition of G_H^x, it follows that $y \notin V(H)$. Without loss of generality, assume that no subsequence of $\mathcal{S}'$ moves t_x to y, and $I'_{k-1} \setminus I'_k = \{x\}$ and $I'_k \setminus I'_{k-1} = \{y\}$. For every independent set I of G, $I \cap G_H^x$ is also an independent set of G_H^x. Therefore, one can construct the TS-sequence $\langle I_1 \cap G_H^x, I_2 \cap G_H^x, \dots, I_\ell \cap G_H^x \rangle$ from $\mathcal{S}$. Thus, $I \cap G_H^x \overset{G_H^x}{\leftrightsquigarrow} J \cap G_H^x \overset{G_H^x}{\leftrightsquigarrow} Ip_{k-1}$. Note that for every independent set I' of G_H^x, since $V(G_H^x) \cap (I - G_H^x) = \emptyset$, the set $I' \cup (I - G_H^x)$ is also independent. Therefore, $I \overset{G}{\leftrightsquigarrow} I'_{k-1} \cup (I - G_H^x)$. Let $J' = I'_{k-1} \cup (I - G_H^x)$. By our assumption, $J' \cap H$ is a maximum independent set of H. Let $J'' = I'_k \cup (I - G_H^x)$. Similarly, $J'' \cap H$ must be a maximum independent set of H. Since $J'' \setminus J' = \{y\}$, $J' \setminus J'' = \{x\}$, and $y \notin V(H)$, we obtain a contradiction. It remains to show (iii) $\Rightarrow$ (ii). Suppose that (iii) holds but (ii) does not. Thus, there must be an independent set J such that $I \overset{G}{\leftrightsquigarrow} J$ and $J \cap H$ is not a maximum independent set of H. Let $\mathcal{S} = \langle I_1 = I, I_2, \dots, I_\ell = J \rangle$ be a TS-sequence in G that reconfigures I to J. Without loss of generality, we can assume that for $1 \leq i \leq \ell - 1$, the set $I_i \cap H$ is a maximum independent set of H. Let xy be an edge of G such that $I_{\ell-1} \setminus I_\ell = \{x\}$ and $I_\ell \setminus I_{\ell-1} = \{y\}$. Since $I_\ell \cap H$ is not a maximum independent set of H, $|I_\ell \cap H| < |I_i \cap H|$ for $i = \{1, 2, \dots, \ell - 1\}$. Hence, $y \notin V(H)$. Since $N_G(x) = N_{G_H^x}(x) \cup N_H(x)$ and $N_{G_H^x}(x) \cap N_H(x) = \emptyset$, the vertex y must be in G_H^x. It follows that $\mathcal{S}$ slides a token t_x on x to a vertex $y \in V(G_H^x)$. As in the previous part, one can indeed derive a TS-sequence in G_H^x from $\mathcal{S}$ that slides t_x to y, i.e., it is $(G_H^x, I_{\ell-1} \cap G_H^x)$-movable, which is a contradiction.

$\square$

Chapter 3

Sliding Token for Trees

In this chapter, we prove the following theorem.

Theorem 3.1. *Let T be a tree on n vertices. Let I, J be two independent sets of T. Then, one can decide in $O(n)$ time whether $I \overset{T}{\leftrightsquigarrow} J$. Moreover, if $I \overset{T}{\leftrightsquigarrow} J$, one can construct a* TS*-sequence $\mathcal{S}$ in T between I and J in $O(n^2)$ time.*

Partial results of this chapter have been presented in [41–43]. The rest of this chapter is organized as follows. In Section 3.1, we claim that one can find all structures that forbid the existence of a TS-sequence between two independent sets of a tree (i.e., the *rigid tokens*) in linear time. Then, in Section 3.2, we present a linear-time algorithm for solving Sliding Token for trees. Finally, in Section 3.3, we give an upper bound on the length of a TS-sequence between any two independent sets of a tree.

Throughout this chapter, T is a tree on n vertices, and I, J are independent sets of T. For two vertices u and v of a tree T, let T^u_v be the subtree of T obtained by regarding u as the root of T and then taking the subtree induced by v and its descendants. Note that $u \notin V(T^u_v)$.

3.1 Rigid Tokens in Trees

We first characterize (T, I)-rigid tokens.

Lemma 3.2. *Let I be an independent set of a tree T, and let u be a vertex in I.*

(a) Suppose that $|V(T)| = |\{u\}| = 1$. Then, the token on u is (T, I)-rigid.

(b) Suppose that $|V(T)| \geq 2$. Then, a token on u is (T, I)-rigid if and only if, for all neighbors $v \in N_T(u)$, there exists a vertex $w \in I \cap N_{T^u_v}(v)$ such that the token on w is $(T^v_w, I \cap T^v_w)$-rigid.

Proof. Part (a) is trivial by definition of rigid tokens. Assume that $|V(T)| \geq 2$, we show part (b).

We first show the if direction of (b). Assume that for all $v \in N_T(u)$, there exists $w \in I \cap N_{T^u_v}(v)$ such that the token on w is $(T^v_w, I \cap T^v_w)$-rigid. We show that the token t on u is (T, I)-rigid. Suppose to the contrary that t is (T, I)-movable, which means that t can be slid to a vertex $v \in N_T(u)$. Note that, to slide t to v, we first need to slide the

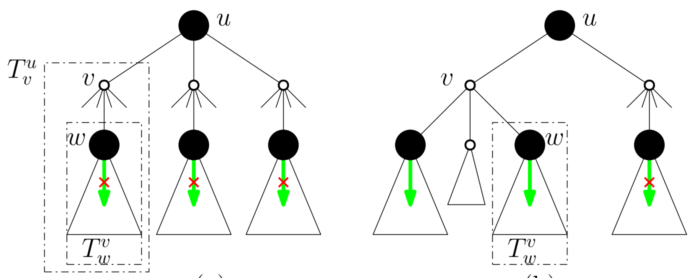


Figure 3.1: (a) A (T, I)-rigid token on u, and (b) a (T, I)-movable token on u.

token t' on w to one of its neighbors other than v, which is a contradiction because t' is $(T_w^v, I \cap T_w^v)$-rigid. Hence, t is (T, I)-rigid.

Next, we show the only-if direction of (b). Suppose that u is (T, I)-rigid, we want to show that for all neighbors $v \in N_T(u)$, there exists a vertex $w \in I \cap N_{T_v^u}(v)$ such that the token on w is $(T_w^v, I \cap T_w^v)$-rigid. We will prove the contrapositive that if either

(i) there exists $v \in N_T(u)$ such that $I \cap N_{T_v^u}(v) = \emptyset$, or

(ii) there exists $v \in N_T(u)$ such that for every $w \in I \cap N_{T_v^u}(v)$, the token on w is $(T_w^v, I \cap T_w^v)$-movable,

then the token t on u is (T, I)-movable. Clearly, if (i) holds, t can be directly slid to v. We now consider the case when (ii) holds. Hence, for each $w \in I \cap N_{T_v^u}(v)$, there exists a TS-sequence $\mathcal{S}_w$ that reconfigures $I \cap T_w^v$ to some independent set $J \subseteq V(T_w^v)$ such that $w \notin J$. Since T_w^v is a subtree of $T - N_T[u]$, Proposition 2.4 implies that the TS-sequence $\mathcal{S}_w$ in T_w^v can be extended to the whole tree T, which implies that, for each w, the token on w is (T, I)-movable and can be slid to one of w's neighbors other than v. Hence, the token on u can finally be slid to v, which means that u is (T, I)-movable.

□

From Lemma 3.2, one can design a $O(n^2)$-time algorithm that computes the set $\mathsf{R}(T, I)$ of all (T, I)-rigid tokens [41, Lemma 2]. However, we can do better. For an independent set I of a bipartite graph G, finding $\mathsf{R}(G, I)$ can be done in $O(n)$ time [31]. As trees is a subclass of bipartite graphs, the algorithm for bipartite graphs can also be applied for trees. However, unlike the case of trees, Sliding Token for bipartite graphs is PSPACE-complete [27]. In this thesis, we employ the following result (without proof).

Lemma 3.3 ([31]). *Let $G = (A \cup B, E)$ be a bipartite graph (with bipartitions A, B) on n vertices and I be an independent set of G. In $O(n)$ time, $\mathsf{R}(G, I)$ can be computed.*

Proof sketch. The general idea is based on the fact that no TS-sequence moves a (G, I)-rigid token. Thus, starting from any independent set I, it is possible to construct a TS-sequence that moves *all* (G, I)-movable tokens. The construction of such a TS-sequence is based on the property that if $I \setminus \mathsf{R}(G, I) \neq \emptyset$ then there must be some vertex u satisfying $|N_G(u) \cap I| = 1$ (Proposition 2.2). Clearly, if v is such that $N_G(u) \cap I = \{v\}$ then the token on v can be slid to u. □

3.2 Sliding Token for trees

Let (T, I, J) be an instance of Sliding Token where I, J are two independent sets of a tree T on n vertices. The following algorithm decides if $I \overset{T}{\leftrightsquigarrow} J$.

- **Step 1:** If $\mathsf{R}(T, I) \neq \mathsf{R}(T, J)$, return NO. Otherwise, go to **Step 2**.
- **Step 2:** Let F be the forest obtained by removing all vertices in $N_T[\mathsf{R}(T, I)] = N_T[\mathsf{R}(T, J)]$. If $|I \cap F'| = |J \cap F'|$ for every component (tree) F' of F then return YES. Otherwise, return NO.

We first analyze the running time of the described algorithm. Lemma 3.3 ensures that **Step 1** can be checked in $O(n)$ time. **Step 2** can also be done in $O(n)$ time. In total, the algorithm can be executed in $O(n)$ time.

Now, we show that the algorithm above correctly decides if $I \overset{T}{\leftrightsquigarrow} J$. The correctness of **Step 1** is followed from Proposition 2.1. It remains to show the correctness of **Step 2**. Before showing the correctness of **Step 2**, we prove the following useful lemma.

Lemma 3.4. *Let I be an independent set of a tree T such that all tokens are (T, I)-movable, and let v be a vertex such that $v \notin I$. Then, there exists at most one neighbor $w \in I \cap N_T(v)$ such that the token on w is $(T^v_w, I \cap T^v_w)$-rigid.*

Proof. Suppose to the contrary that there are two neighbors $w, w' \in I \cap N_T(v)$ such that the tokens on both w and w' are respectively $(T^v_w, I \cap T^v_w)$-rigid and $(T^v_{w'}, I \cap T^v_{w'})$-rigid. We claim that the token t on w is (T, I)-rigid.

Suppose to the contrary that t is (T, I)-movable. Since t is $(T^v_w, I \cap T^v_w)$-rigid, the only way to move t is sliding it to v. But, to slide t to v, we need to slide the token t' on w' to a vertex of $T^v_{w'}$. This contradicts our assumption that w' is $(T^v_{w'}, I \cap T^v_{w'})$-rigid. □

Next, we show that deleting the vertices where rigid tokens are placed together with their neighbors does not affect the reconfigurability. More precisely,

Lemma 3.5. *Suppose that $\mathsf{R}(T, I) = \mathsf{R}(T, J)$ for two given independent sets I and J of a tree T, and let F be the forest obtained by deleting the vertices in $N_T[\mathsf{R}(T, I)] = N_T[\mathsf{R}(T, J)]$ from T. Then, $I \overset{T}{\leftrightsquigarrow} J$ if and only if $I \cap F \overset{F}{\leftrightsquigarrow} J \cap F$. Furthermore, all tokens in $I \cap F$ are $(F, I \cap F)$-movable, and all tokens in $J \cap F$ are $(F, J \cap F)$-movable.*

Proof. By definition of rigid tokens, for any TS-sequence $\mathcal{S}$ between I and J in T, $\mathsf{R}(T, I) = \mathsf{R}(T, J)$ is a subset of each independent set in $\mathcal{S}$. Thus, Proposition 2.4 implies that $I \overset{T}{\leftrightsquigarrow} J$ if and only if $I \cap F \overset{F}{\leftrightsquigarrow} J \cap F$ (regarding W as $\mathsf{R}(T, I) = \mathsf{R}(T, J)$).

It remains to show that all tokens in $I \cap F$ are $(F, I \cap F)$-movable. A similar argument can be applied for $J \cap F$. Note that each token t on a vertex v in $I \cap F$ is (T, I)-movable; otherwise $t \in \mathsf{R}(T, I)$. Hence, there exists an independent set I' of T such that $I \overset{T}{\leftrightsquigarrow} I'$ and $v \notin I'$. As we have shown before, $I \cap F \overset{F}{\leftrightsquigarrow} I' \cap F$. Therefore, t is $(F, I \cap F)$-movable. □

Next, we show that if $\mathsf{R}(T, I) = \mathsf{R}(T, J) = \emptyset$ then $I \overset{T}{\leftrightsquigarrow} J$ if and only if $|I| = |J|$.

Lemma 3.6. *Let I and J be two independent sets of a tree T such that all tokens in I and J are (T, I)-movable and (T, J)-movable, respectively. Then, $I \overset{T}{\leftrightsquigarrow} J$ if and only if $|I| = |J|$.*

Before proving Lemma 3.6, we show some useful lemmas.

Lemma 3.7. *Let I be an independent set of a tree T such that $\mathsf{R}(T, I) = \emptyset$. Let $v \notin I$ be a safe vertex of T. Then, there exists an independent set I' such that $v \in I'$ and $I \overset{T}{\leftrightsquigarrow} I'$.*

Proof. Let $M = \{w \in I : \mathsf{dist}_T(v, w) = \min_{x \in I} \mathsf{dist}_T(v, x)\}$. Let w be an arbitrary vertex in M, and let $P = p_0 p_1 \dots p_\ell$ ($p_0 = v$, $p_\ell = w$) be the vw-path in T. (See Figure 3.2.)

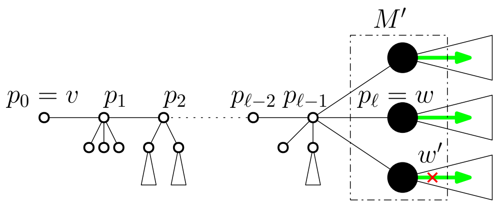


Figure 3.2: Illustration for Lemma 3.7.

If $\ell = 1$ and hence $p_1 \in I$, then we can simply slide the token on p_1 to v.

We now consider the case $\ell \geq 2$. Since the token on w is closest to v, no token in I can be placed on the vertices $p_0, \dots, p_{\ell-1}$ and the neighbors of $p_0, \dots, p_{\ell-2}$. Let $M' = M \cap N_T(p_{\ell-1})$. Since $p_{\ell-1} \notin I$, by Lemma 3.4, there is at most one vertex $w' \in M'$ such that the token on w' is $(T^{p_{\ell-1}}_{w'}, I \cap T^{p_{\ell-1}}_{w'})$-rigid. We choose such a vertex w' if exists, otherwise choose an arbitrary vertex in M' and regard it as w'. Before sliding the token on w' to v, we need to slide all tokens on the vertices w'' in $M' \setminus \{w'\}$ first. Since all tokens on the vertices w'' in $M' \setminus \{w'\}$ are $(T^{p_{\ell-1}}_{w''}, I \cap T^{p_{\ell-1}}_{w''})$-movable, we can slide the tokens on w'' to some vertices in $T^{p_{\ell-1}}_{w''}$. Now, we can slide the token on w' to v along the path P. In this way, we obtain an independent set I' such that $v \in I'$ and $I \overset{T}{\leftrightsquigarrow} I'$. $\square$

We now prove that deleting a safe vertex along with a token on it does not affect the movability of the other tokens. We note that if v is a safe vertex of a tree T then $\deg_T(v) = 1$.

Lemma 3.8. *Let v be a safe vertex of a tree T, and let $\bar{T}$ be the subtree of T obtained by deleting v, its unique neighbor u, and the resulting isolated vertices. Let I be an independent set of T such that $v \in I$ and all tokens are (T, I)-movable. Then, all tokens in $I \setminus \{v\}$ are $(\bar{T}, I \setminus \{v\})$-movable.*

Proof. Since T^u_v consists of a single vertex v, the token on v is $(T^u_v, I \cap T^u_v)$-rigid. Therefore, no token is placed on degree-1 neighbors of u other than v (see Figure 3.3), because otherwise it contradicts to Lemma 3.4; recall that all tokens in I are assumed to be (T, I)-movable.

Let $\bar{I} = I \setminus \{v\}$. Suppose for a contradiction that there exists a token in $\bar{I}$ which is $(\bar{T}, \bar{I})$-rigid. Let $w_p \in \bar{I}$ be such a vertex closest to v, and let z be the vertex on the vw_p-path right before w_p.

- **Case 1:** $z = u$. (See Figure 3.3(a).) Recall that the token on v is (T, I)-movable, but is $(T^u_v, I \cap T^u_v)$-rigid. Therefore, by Lemma 3.4 the token on w_p must be $(T^u_{w_p}, I \cap T^u_{w_p})$-movable. However, this contradicts the assumption that w_p is $(\bar{T}, \bar{I})$-rigid, because $\bar{T} = T^u_{w_p}$ and $\bar{I} = I \cap T^u_{w_p}$ in this case.

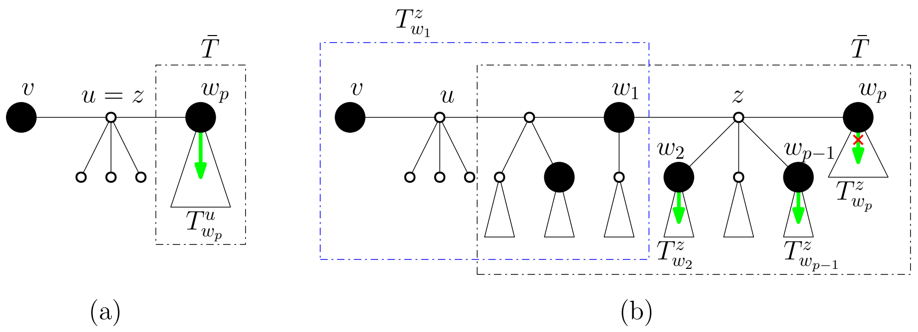


Figure 3.3: Illustration for Lemma 3.8.

- **Case 2:** $z \neq u$. (See Figure 3.3(b).) Let w_1 be the neighbor of z on the vw_p-path other than w_p. Let $N_T(z) = \{w_1, w_2, \ldots, w_p\}$. We note that the subtree $T^z_{w_1}$ contains the deleted star $T \setminus \bar{T}$ centered at u, because only the neighbor w_1 of z is on the vz-path. We first note that the token t_p on w_p is $(\bar{T}^z_{w_p}, \bar{I} \cap \bar{T}^z_{w_p})$-rigid, because otherwise t_p can be slid to some vertex in $\bar{T}^z_{w_p}$ and hence it is $(\bar{T}, \bar{I})$-movable. Since $\bar{T}^z_{w_p} = T^z_{w_p}$ and $\bar{I} \cap \bar{T}^z_{w_p} = I \cap T^z_{w_p}$, the token t_p is also $(T^z_{w_p}, I \cap T^z_{w_p})$-rigid. For each $j \in \{2, 3, \ldots, p-1\}$ with $w_j \in I$, since t_p is $(T^z_{w_p}, I \cap T^z_{w_p})$-rigid, by Lemma 3.4 each token t_j on w_j is $(T^z_{w_j}, I \cap T^z_{w_j})$-movable. Then, since $T^z_{w_j} = \bar{T}^z_{w_j}$ and $I \cap T^z_{w_j} = \bar{I} \cap \bar{T}^z_{w_j}$, the token t_j is $(\bar{T}^z_{w_j}, \bar{I} \cap \bar{T}^z_{w_j})$-movable. Therefore, if $w_1 \notin \bar{I}$ or the token t_1 on w_1 is $(\bar{T}^z_{w_1}, \bar{I} \cap \bar{T}^z_{w_1})$-movable, then we can slide t_p from w_p to z after sliding each token t_j in $\bar{I} \cap \{w_1, w_2, \ldots, w_{p-1}\}$ to some vertex of the subtree $\bar{T}^z_{w_j}$. This contradicts the assumption that t_p is $(\bar{T}, \bar{I})$-rigid. Therefore, we have $w_1 \in \bar{I}$ and a token t_1 on w_1 is $(\bar{T}^z_{w_1}, \bar{I} \cap \bar{T}^z_{w_1})$-rigid. However, since t_p is $(\bar{T}^z_{w_p}, \bar{I} \cap \bar{T}^z_{w_p})$-rigid, this implies that t_1 is $(\bar{T}, \bar{I})$-rigid. Since w_1 is on the vw_p-path in T, this contradicts the assumption that t_p is the $(\bar{T}, \bar{I})$-rigid token closest to v.

□

We are now ready to show the proof of Lemma 3.6.

Proof of Lemma 3.6. The only-if direction of this lemma is trivial. We now prove the if direction.

Suppose that $|I| = |J|$. We claim that there is an independent set I^* such that $I \overset{T}{\leftrightsquigarrow} I^*$ and $J \overset{T}{\leftrightsquigarrow} I^*$. Since a TS-sequence is reversible, $I \overset{T}{\leftrightsquigarrow} I^*$ and $J \overset{T}{\leftrightsquigarrow} I^*$ imply that $I \overset{T}{\leftrightsquigarrow} J$.

We now describe how to construct I^*. Initially, let $I^* = \emptyset$.

- Pick a safe vertex v of T, and add v to I^*.
- Let I' be such that $v \in I'$ and $I \overset{T}{\leftrightsquigarrow} I'$. From Lemma 3.7, such a I' exists.
- Let J' be such that $v \in J'$ and $J \overset{T}{\leftrightsquigarrow} J'$. From Lemma 3.7, such a J' exists.
- Let T' be the tree obtained by deleting v, its unique neighbor u, and the resulting isolated vertices.
- Repeat the above steps with the triple $(T', I' \setminus \{v\}, J' \setminus \{v\})$ instead of (T, I, J) until $|I^*| = |I| = |J|$.

Lemma 3.8 and Proposition 2.1 guarantee that after each iteration, $|I^*|$ increases by 1, and $\mathsf{R}(T', I') = \mathsf{R}(T', J') = \emptyset$. Proposition 2.4 and Lemma 3.7 ensure that $I \overset{T}{\leftrightsquigarrow} J$ if and only if $I' \setminus \{v\} \overset{T'}{\leftrightsquigarrow} J' \setminus \{v\}$. Put everything together, the correctness of this construction is now clear.

□

From Lemmas 3.5 and 3.6, it follows that **Step 2** is correct, which implies the correctness of our described algorithm

3.3 Length of Reconfiguration Sequence

We show that given two independent sets I, J of a tree T such that $I \overset{T}{\leftrightsquigarrow} J$, one can construct an actual TS-sequence between I and J using a polynomial number of token-slides.

Lemma 3.9. *Let (T, I, J) be a* YES*-instance of* SLIDING TOKEN *where I, J are two independent sets of a tree T on n vertices. Then, one can construct a TS-sequence $\mathcal{S}$ of length $O(n^2)$ such that $\mathcal{S}$ reconfigures I to J.*

Before proving Lemma 3.9, we show the following useful lemma.

Lemma 3.10. *Let I be an independent set of a tree T, and let $w \in I$. For a neighbor $z \in N_T(w)$, suppose that the token on w is $(T^z_w, I \cap T^z_w)$-movable. Then, there exists a TS-sequence $\mathcal{S}_w$ of length $O(|V(T^z_w)|)$ from I to an independent set I' of T such that $w \notin I'$ and $J \cap (T \setminus T^z_w) = I \cap (T \setminus T^z_w)$ for all $J \in \mathcal{S}_w$. Furthermore, $\mathcal{S}_w$ can be output in $O(|V(T^z_w)|)$ time.*

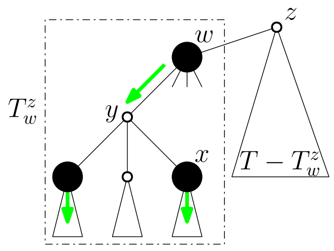


Figure 3.4: Illustration for Lemma 3.10.

Proof. We prove the lemma by induction on the depth of T^z_w (regarding w as root), where the depth of a tree is the longest distance from its root to a leaf. If the depth of T^z_w is zero (and hence T^z_w consists of a single vertex w), then the token on w is $(T^z_w, I \cap T^z_w)$-rigid; this contradicts the assumption. Hence, we may assume that the depth of T^z_w is at least one.

- **Base case:** If the depth of T^z_w is exactly one, then T^z_w is a star centered at w, and no token is placed on any neighbor of w. Thus, we can slide the token on w by 1 ($< |V(T^z_w)|$) token-slides. Then, the lemma holds for trees with depth one.

- **Inductive step:** Assume that the depth of T^z_w is $k \geq 2$, and that the lemma holds for trees with depth at most $k-1$. Since the token on w is $(T^z_w, I \cap T^z_w)$-movable, by Lemma 3.2, there exists $y \in N_{T^z_w}(w)$ such that all tokens on the vertices x in $I \cap N_{T^w_y}(y)$ are $(T^y_x, I \cap T^y_x)$-movable. (See Figure 3.4.)

Then, we can obtain a TS-sequence which (1) first slides all tokens on the vertices x in $I \cap N_{T^w_y}(y)$ to some vertices in T^y_x, and (2) then slide the token on w to the vertex y. By applying the induction hypothesis to each subtree T^y_x, this TS-sequence is of length

$$1 + \sum_{x \in I \cap N_{T^w_y}(y)} |V(T^y_x)| = \big|V(T^w_y)\big|,$$

and can be output in time $O(\big|V(T^w_y)\big|)$. Note that $w \notin I'$ holds for the obtained independent set I' of T. Thus, the lemma holds for trees with depth k.

□

We are now ready to prove Lemma 3.9.

Proof of Lemma 3.9. It is sufficient to show Lemma 3.9 for the case when $\mathsf{R}(T, I) = \mathsf{R}(T, J) = \emptyset$; otherwise, we simply remove $N_G[\mathsf{R}(T, I)] = N_G[\mathsf{R}(T, I)]$ and deal with each component of the resulting graph separately. Let I^* be an independent set described in the proof of Lemma 3.6. We now describe how to construct a TS-sequence $\mathcal{S}_1$ that reconfigures I to I^*. A TS-sequence $\mathcal{S}_2$ between J and I^* can be constructed in the same manner. Clearly, a TS-sequence between I and J in T can be formed by appending $\mathcal{S}_1$ to the TS-sequence obtained by reversing steps in $\mathcal{S}_2$.

- Pick a safe vertex v of T in I^*.
- As in the proof of Lemma 3.7, construct an independent set I' with $v \in I'$ and $I \overset{T}{\leftrightsquigarrow} I'$. Indeed, Lemma 3.10 ensures that a token closest to v can be slid to v in $O(n)$ time.
- Let T' be the tree obtained by deleting v, its unique neighbor u, and the resulting isolated vertices.
- Repeat the above steps with the new triple $(T', I' \setminus \{v\}, I^* \setminus \{v\})$ instead of the triple (T, I, I^*). The process stops when there is no token left to slide.

Lemma 3.6 guarantees that the described algorithm correctly constructs a TS-sequence $\mathcal{S}_1$ between I and I^*. As for each $v \in I^*$, a token closest to v can be slid to v in $O(n)$ time, which then implies that the algorithm takes $O(n) \times |I^*| = O(n^2)$ time.

□

Chapter 4

Sliding Token for Cactus Graphs

In this chapter, we prove the following theorem.

Theorem 4.1. *Let G be a cactus graph on n vertices. Let I, J be two independent sets of G. Then, one can decide in $O(n^2)$ time whether $I \overset{G}{\leftrightsquigarrow} J$. Moreover, if $I \overset{G}{\leftrightsquigarrow} J$, one can construct a TS-sequence $\mathcal{S}$ in G between I and J in $O(n^2)$ time.*

From Proposition 2.3, it is sufficient to consider *connected* cactus graphs. Partial results of this chapter have been presented in [44]. The rest of this chapter is organized as follows. In Section 4.1, we prove some useful observations for tackling Sliding Token for cactus graphs. Then, in Section 4.2, we claim that one can find all structures that forbid the existence of a TS-sequence between any two independent sets of a cactus graph in polynomial time. We then describe a polynomial-time algorithm for solving Sliding Token for a cactus graph and show its correctness in Section 4.3. Finally, in Section 4.4, we give an upper bound on the length of a TS-sequence between any two independent sets of a cactus graph.

4.1 Some Useful Observations

First of all, we consider Sliding Token for cycles—a subclass of cactus graphs. More precisely, we claim that for a given instance (C, I, J) of Sliding Token, where I and J are independent sets of a k-vertex cycle C, if there are no (C, I)-rigid and (C, J)-rigid tokens, one can reconfigure I to J in $O(k^2)$ time if and only if $|I| = |J|$.

Lemma 4.2. *Let I and J be two given independent sets of a k-vertex cycle C. Assume that $\mathsf{R}(C, I) = \mathsf{R}(C, J) = \emptyset$. Then $I \overset{C}{\leftrightsquigarrow} J$ if and only if $|I| = |J|$. Moreover, if $I \overset{C}{\leftrightsquigarrow} J$, one can construct a TS-sequence between them in $O(k^2)$ time.*

Proof. The only-if direction is trivial. We show the if direction, i.e., if $|I| = |J|$, we show that $I \overset{C}{\leftrightsquigarrow} J$. Let $C = v_1 v_2 \dots v_k v_1$. Let I' be an independent set of C such that $|I'| = |I| = |J| \leq \lfloor k/2 \rfloor$ and $v_i \in I'$ if i is odd $(1 \leq i \leq 2|I| - 1)$. We claim that I can be reconfigured to I' in $O(k^2)$ time. Consider the following cases:

- **Case 1:** $|I| = \lfloor k/2 \rfloor$. Since there are no (C, I)-rigid tokens and $|I| = \lfloor k/2 \rfloor$, k must be odd. Let i be the smallest index such that $v_i \in I \setminus I'$ for $2 \leq i \leq k$. Hence, from the definition of I', i must be even. Additionally, $v_j \in I'$ for odd j with

$1 \leq j \leq i-1$, and $v_j \in I$ for even j with $i \leq j \leq k-1$. Hence, one can slide the token on v_i to $v_{i-1} \in I' \setminus I$, then slide the token on v_{i+2} to v_{i+1}, and so on. Let $\mathcal{S}$ be the TS-sequence describing the process above, then clearly it reconfigures I to I', since each sliding step reduces $|I' \setminus I|$. Clearly, this process takes $O(k)$ token-slides.

- **Case 2:** $|I| < \lfloor k/2 \rfloor$. For $2 \leq i \leq k$, let i be the smallest index such that $v_i \in I \setminus I'$. If $i = 2$ and $v_k \in I$, we first slide the token on v_k to v_{k-1} recursively as follows: if $v_{k-2} \notin I$, move the token on v_k to v_{k-1} directly; otherwise, recursively apply the procedure with v_{k-2} and v_{k-3}, instead of v_k and v_{k-1}. As C is finite and $\mathsf{R}(C, I) = \emptyset$, this process eventually moves the token on v_k to v_{k-1} using $O(k)$ token-slides. For convenience, we call the resulting independent set I. Let j be the smallest index such that $v_j \in I' \setminus I$ for $1 \leq j \leq k$. Since $v_i \notin I'$, it follows that $i > j$. Now, one can slide the token on v_i to v_j along the unique v_iv_j-path in C (using $O(k)$ token-slides) and repeat the process. Let $\mathcal{S}$ be the TS-sequence describing the process above, then clearly $I \overset{C}{\leftrightsquigarrow} I'$. Moreover, since each token is moved using $O(k)$ steps, and the "adjustment" in case $i = 2$ and $v_k \in I$ also takes $O(k)$ token-slides, it follows that the length of $\mathcal{S}$ is $O(k^2)$. Similarly, one can also show that $J \overset{C}{\leftrightsquigarrow} I'$. A TS-sequence between I and J can be formed by first reconfiguring I to I', and then from I' to J by reversing the constructed TS-sequence that reconfigures J to I'. Thus, one can indeed construct a TS-sequence of length $O(k^2)$ between I and J.

□

From Proposition 2.2 and Lemma 4.2, it is not hard to desgin an algorithm for solving Sliding Token for cycles. More precisely, the following algorithm decides if $I \overset{C}{\leftrightsquigarrow} J$ where I, J are independent sets of a cycle C on k vertices.

- If $|I| \neq |J|$, return NO.
- Otherwise, if k is even and $|I| = |J| = k/2$ then return NO.
- Otherwise, return YES.

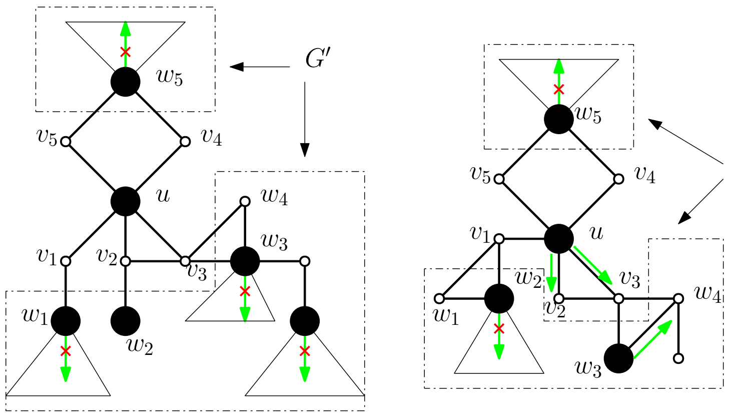
4.2 Rigid Tokens and Confined Cycles in Cactus Graphs

In this subsection, we characterize two non-trivial structures, namely *rigid token* and *confined (induced) cycle*, that forbid the existence of a TS-sequence between two independent sets of a cactus graph. We shall prove that one can find these forbidden structures in polynomial time. Throughout this subsection, unless mentioned otherwise, we always assume that G is a cactus and I is an independent set of G. As we only need to consider induced cycles, from now on, we shall conveniently assume that any cycle of a cactus graph G considered in this subsection and the next ones is an induced cycle of G. First of all, we prove a recursive characterization of (G, I)-rigid tokens in a cactus graph.

Lemma 4.3. *Let I be an independent set of a cactus G. For every vertex $u \in I$, the token t placed at u is (G, I)-rigid (see Figure 4.1(a)) if and only if for every vertex $v \in N_G(u)$, there exists a vertex $w \in \big(N_G(v) \setminus \{u\}\big) \cap I$ satisfying one of the following conditions:*

(i) The token t_w on w is $(G', I \cap G')$-rigid, where $G' = G - N_G[u]$.

(ii) The token t_w on w is $(G', I \cap G')$-movable; and there exists a cycle C in G such that $u \notin V(C)$, $\{v, w\} \subseteq V(C)$, and the path $P = C - v$ is $(G', I \cap G')$-confined.



(a) (b)

Figure 4.1: (a) The token placed at $u \in I$ is (G, I)-rigid. (The tokens placed at w_1, w_2, w_5 satisfy Lemma 4.3(i), while the one placed at w_3 satisfies Lemma 4.3(ii).); (b) The token placed at $u \in I$ is (G, I)-movable. (It can be moved to either v_2 or v_3.)

Proof. First of all, we show the if direction. Let $v \in N_G(u)$. Assume that there exists $w \in \big(N_G(v) \setminus \{u\}\big) \cap I$ such that either (i) or (ii) holds. We claim that in both cases, t cannot be slid to v (see Figure 4.1(a)). Since this claim holds for every $v \in N_G(u)$, it follows that t is (G, I)-rigid. Note that by Proposition 2.4, as long as t is placed at u, every TS-sequence in G' can be extended to a TS-sequence in G and vice versa.

If (i) holds, then clearly no TS-sequence in G' slides t_w to a vertex in $N_{G'}(w) = N_G(w) \setminus \{v\}$. Hence, t cannot be slid to v.

Now, consider the case when (ii) holds. Since t_w is $(G', I \cap G')$-movable, it can be (at least) slid in G' to a vertex $x \in N_{G'}(w)$ by some TS-sequence $\mathcal{S}$. Since P is $(G', I \cap G')$-confined, no TS-sequence in G' slides a token from $G' - P$ to P and vice versa. Clearly, this also holds for $\mathcal{S}$. Let $w' \in N_G(v) \cap V(C)$ such that $w' \neq w$. Hence, if $w' \notin I$ then before sliding any other token in P, $\mathcal{S}$ must move a token in $N_P(w') \cap I$ (because $I \cap P$ is a maximum independent set of P) to w'. It follows that $N_G(v) \cap I' \neq \emptyset$ for every I' such that $I \cap G' \overset{G'}{\rightsquigarrow} I'$. Thus, t cannot be slid to v.

Next, we show the only-if direction by contraposition. More precisely, we claim that if both (i) and (ii) do not hold, then t is (G, I)-movable (see Figure 4.1(b)).

- **Case 1: There exists** $v \in N_G(u)$ **such that** $\big(N_G(v) \setminus \{u\}\big) \cap I = \emptyset$. Clearly, t can be slid to v and hence is (G, I)-movable.

- **Case 2: For all** $v \in N_G(u)$, $\big(N_G(v) \setminus \{u\}\big) \cap I \neq \emptyset$. Let $w \in \big(N_G(v) \setminus \{u\}\big) \cap I$. Since (i) does not hold, we can assume that t_w is $(G', I \cap G')$-movable. Since (ii) does not hold, for every cycle C of G, (at least) one of the following conditions does not hold: (a) $u \notin V(C)$; (b) $\{v, w\} \subseteq V(C)$; (c) P is (G, I)-confined. Note that by definition, $w \neq u$. Additionally, since G is a cactus, there is one cycle C that contains both v and w. Let $H(G', w)$ be the component of G' containing w. We

claim that for such w above, one can slide t_w to a vertex in $N_{H(G',w)}(w)$ without sliding another token to a vertex in $N_G(v)$ beforehand. Eventually, there are no tokens in $N_G(v)$ other than t. Consider the following cases:
Case 2-1: Any cycle C of G contains either v or w but not both of them. Since t_w is (G, I)-movable, it is also $(H(G', w), I \cap H(G', w))$-movable. Suppose that there exists a vertex $x \in N_G(v) \cap H(G', w)$, $x \neq w$. Since $H(G', w)$ is connected, there exists a wx-path Q in $H(G', w)$. Note that Q, vw and vx form a cycle in G that contains both v and w, which contradicts our assumption. Hence, $N_G(v) \cap H(G', w) = \{w\}$. Therefore, one can simply slides t_w to a vertex in $N_{H(G',w)}(w)$ without sliding another token to a vertex in $N_G(v)$ beforehand.
Case 2-2: There is a (unique) cycle C that contains both v and w. We consider the cases when $u \in V(C)$ and $u \notin V(C)$.

- **When $u \in V(C)$ holds.** As before, $N_G(v) \cap H(G', w) = \{w\}$. Otherwise, using the same argument as before, the wx-path Q, vw and vx form a cycle C' in G that contains both v and w, where $x \in N_G(v) \cap H(G', w)$ and $x \neq w$. Because Q (a subgraph of G') does not contain u, it follows that $C' \neq C$, which is a contradiction. Since $N_G(v) \cap H(G', w) = \{w\}$, one can simply slides t_w to a vertex in $N_{H(G',w)}(w)$ without sliding another token to a vertex in $N_G(v)$ beforehand.
- **When $u \notin V(C)$ holds.** Let $w' \in N_C(v)$, $w' \neq w$. By definition of a cactus and our assumption, $N_C(v) \cap H(G', w) = \{w, w'\}$. Since $\{v, w\} \subseteq V(C)$, it must happen that the condition (c) does not hold. By Proposition 2.5, there exists an independent set I' with $I \cap G' \overset{G'}{\leftrightsquigarrow} I'$ such that $|I \cap P| < \lfloor k/2 \rfloor$, where $P = C - v$ and k is the length of C. (A maximum independent set of P must be of size $\lfloor k/2 \rfloor$.) Suppose that both w and w' are in I'. Note that both t_w and $t_{w'}$ are (G', I')-movable. Let $\mathcal{S}_w$ be a TS-sequence in G' that slides t_w to a vertex $x \in N_{H(G',w)}(w)$. Similarly, let $\mathcal{S}_{w'}$ be a TS-sequence in G' that slides $t_{w'}$ to a vertex $y \in N_{H(G',w)}(w')$. Since $|I' \cap P| \leq \lfloor k/2 \rfloor - 1$, $\mathcal{S}_w$ (resp., $\mathcal{S}_{w'}$) does not involve any vertex in $I \cap G^x_C$ where $x \in N_C[w']$ (resp., $x \in N_C[w]$). Note that by Proposition 2.4, $\mathcal{S}_w$ and $\mathcal{S}_{w'}$ can indeed be performed in G. Clearly, after applying both $\mathcal{S}_w$ and $\mathcal{S}_{w'}$, the number of tokens in $N_G(v)$ is reduced. Next, if either w or w' is in I', we can simply perform either $\mathcal{S}_w$ or $\mathcal{S}_{w'}$, respectively. If none of them is in I', nothing needs to be done. We showed that in each case, the number of tokens in $N_G(v)$ is reduced each time we slide the $(G', I \cap G')$-movable token in $w \in \big(N_G(v) \setminus \{u\}\big) \cap I$ to a vertex not in $N_G(v)$, and all such slidings can be performed independently (in each component of G'). Eventually, $N_G(v) \cap I = \{u\}$, and hence we can slide t to v immediately, which implies that t is (G, I)-movable.

□

Let P be an induced path of G of even length k. From Proposition 2.5, P is (G, I)-confined if and only if $I \cap P$ is a maximum independent set of P and for $x \in I \cap P$, the token on x is $(G^x_P, I \cap G^x_P)$-rigid. Additionally, since k is even and $I \cap P$ is a maximum independent set of P, no token can be slid along any edge of P. Hence, P is (G, I)-confined if and only if $I \cap P$ is a maximum independent set of P and for $x \in I \cap P$, the token on x is (G, I)-rigid. Now, we consider the case k is odd.

Lemma 4.4. *Let G be a cactus. Let $P = p_1p_2 \dots p_l$ be an induced path in G. Let I be an independent set of G satisfying that $I \cap P$ is a maximum independent set of P. Assume that for $x \in I \cap P$, the token on x is (G, I)-movable.*

Then, P is (G, I)-confined if and only if l is even (i.e., the length $k = l - 1$ of P is odd) and there exist two independent sets ${I'}_1$ and ${I'}_2$ such that

(i) $I \overset{G}{\leftrightsquigarrow} I'$, where $I' \in \{I, {I'}_1, {I'}_2\}$;

(ii) ${I'}_1 \cap P = \{p_1, p_3, \dots, p_{l-1}\}$, ${I'}_2 \cap P = \{p_2, p_4, \dots, p_l\}$; and

(iii) for every $x \in I' \cap P$, the token placed at x is $(G^x_P, I' \cap G^x_P)$-rigid.

Proof. We first show the if direction. Assume that l is even and the described independent sets ${I'}_1, {I'}_2$ exist. Since $I \cap P$ is a maximum independent set of P, it suffices to show that all tokens in $I \cap P$ are $(G, I, V(P))$-confined. By Proposition 2.5, it is equivalent to saying that for every J satisfying $I \overset{G}{\leftrightsquigarrow} J$ and for $x \in J \cap P$, the token on x is $(G^x_P, J \cap G^x_P)$-rigid. Let $x \in J \cap {I'}_1 \cap P$ for some J such that $I \overset{G}{\leftrightsquigarrow} J$ and suppose that the token t_x placed at x is $(G^x_P, {I'}_1 \cap G^x_P)$-rigid. We claim that it is also $(G^x_P, J \cap G^x_P)$-rigid. Suppose to the contrary that there exists an independent set J' of G^x_P such that $J \cap G^x_P \overset{G^x_P}{\leftrightsquigarrow} J'$ but $x \notin J'$. For every independent set I of G, note that $I \cap G^x_P$ is also independent. Hence, it follows that ${I'}_1 \cap G^x_P \overset{G^x_P}{\leftrightsquigarrow} J \cap G^x_P \overset{G^x_P}{\leftrightsquigarrow} J'$, which then implies that t_x is not $(G^x_P, {I'}_1 \cap G^x_P)$-rigid, a contradiction. Hence, for every independent set J with $I \overset{G}{\leftrightsquigarrow} J$, any token in $J \cap {I'}_1 \cap P$ is $(G^x_P, J \cap G^x_P)$-rigid. Similarly, for every independent set J with $I \overset{G}{\leftrightsquigarrow} J$, any token in $J \cap {I'}_2 \cap P$ is also $(G^x_P, J \cap G^x_P)$-rigid. Moreover, for every J with $I \overset{G}{\leftrightsquigarrow} J$, $J \cap P = \big(J \cap {I'}_1 \cap P\big) \cup \big(J \cap {I'}_2 \cap P\big)$. Hence, every token placed at $x \in J \cap P$ is $(G^x_P, J \cap G^x_P)$-rigid.

Now, we show the only-if direction. Assume that P is (G, I)-confined. Since $I \cap P$ is a maximum independent set of P and any token placed at $x \in I \cap P$ is (G, I)-movable, it follows that l must be even. We show how to construct ${I'}_1$ from I using TS rule. A similar process can be applied for ${I'}_2$. Let i be the smallest index such that $p_i \in I \setminus {I'}_1$. From the definition of ${I'}_1 \cap P$, i must be even. Since $I \cap P$ is a maximum independent set of P, it follows that $p_j \in {I'}_1$ for j odd, $j \leq i - 1$, and $p_j \in I \setminus {I'}_1$ for j even, $j \geq i$. By Proposition 2.5, any token placed at $x \in I \cap P$ must be $(G^x_P, I \cap G^x_P)$-rigid. Since the token t_{p_i} on p_i is (G, I)-movable but $(G^{p_i}_P, I \cap G^{p_i}_P)$-rigid, it can only be slid to p_{i-1}. In other words, there exists a TS-sequence $\mathcal{S}_{p_i}$ in G that slides t_{p_i} to p_{i-1}. Note that $\mathcal{S}_{p_i}$ can be constructed recursively as follows. From Lemma 4.3, if $\big(N_G(p_{i-1}) \setminus \{p_i\}\big) \cap I = \emptyset$, $\mathcal{S}_{p_i}$ contains only a single step of sliding t_{p_i} to p_{i-1}. On the other hand, if $\big(N_G(p_{i-1}) \setminus \{p_i\}\big) \cap I \neq \emptyset$, there must be a TS-sequence ${\mathcal{S}'}_{p_i}$ in $G' = G - N_G[p_i]$ that slides any token in $\big(N_G(p_{i-1}) \setminus \{p_i\}\big) \cap I$ to some vertex not in $N_G(p_{i-1}) \setminus \{p_i\}$ without moving a new token to $N_G(p_{i-1}) \setminus \{p_i\}$ beforehand. From Proposition 2.4, ${\mathcal{S}'}_{p_i}$ can be extended to a TS-sequence in G. Hence, $\mathcal{S}_{p_i}$ is constructed by simply performing ${\mathcal{S}'}_{p_i}$ first, then performing a single sliding step which moves t_{p_i} to p_{i-1}. Repeat the described steps, we finally obtain an independent set ${I'}_1$ which satisfies $I \cap G' \overset{G'}{\leftrightsquigarrow} {I'}_1$ and ${I'}_1 \cap P = \{p_1, p_3, \dots, p_{l-1}\}$. □

In the next lemma, we prove that, $\mathsf{R}(G, I)$ can be computed in polynomial time.

Lemma 4.5. *Let I be an independent set of a cactus G. One can check if the token t placed at $u \in I$ is (G, I)-rigid in $O(n)$ time, where $n = |G|$. Consequently, one can determine all (G, I)-rigid tokens in $O(n^2)$ time.*

Proof. Based on Lemma 4.3, we describe a recursive function $\textsc{CheckRigid}(G,\, I,\, u)$ that will output YES if the token t placed at $u \in I$ is (G, I)-rigid. Otherwise, it outputs NO and a TS-sequence $\mathcal{S}_u$ that moves t to one of its neighbors.

Clearly, if $N_G(u) = \emptyset$ then (by definition) t is (G, I)-rigid, and the function outputs YES. We can now assume that $N_G(u) \neq \emptyset$. We analyze the cases when t is not (G, I)-rigid. If there exists $v \in N_G(u)$ such that $\big(N_G(v) \setminus \{u\}\big) \cap I = \emptyset$ then clearly t is not (G, I)-rigid. The function outputs NO and a TS-sequence $\mathcal{S}_u$ contains a single step of sliding t to v. Otherwise, for each $w \in \big(N_G(v) \setminus \{u\}\big) \cap I$, we recursively call $\textsc{CheckRigid}(G',\, I \cap G',\, w)$ to check if the token t_w at w is $(G', I \cap G')$-rigid, where $G' = G - N_G[u]$. It suffices to use $\textsc{CheckRigid}(H(G', w),\, I \cap H(G', w),\, w)$, where $H(G', w)$ is the component of G' containing w. Note that by the definition of a cactus, it must happen that $1 \leq |N_G(v) \cap H(G', w)| \leq 2$. Consider the following cases.

Case 1: $N_G(v) \cap H(G', w) = \{w\}$. In this case, the cycle C mentioned in Lemma 4.3(ii) does not exist. Hence, for every $w \in \big(N_G(v) \setminus \{u\}\big) \cap I$, if $\textsc{CheckRigid}(H(G', w),\, I \cap H(G', w),\, w)$ outputs NO and a TS-sequence $\mathcal{S}_w$ that moves t_w to a vertex in $N_{H(G',w)}(w)$, we can immediately output NO and a TS-sequence $\mathcal{S}_u$ that slides t to v by first applying all such $\mathcal{S}_w$, and then applying a single step of sliding t to v.

Case 2: $N_G(v) \cap H(G', w) = \{w, w'\}$, $(w' \neq w)$. In this case, the cycle C mentioned in Lemma 4.3(ii) does exist. If for all $w \in \big(N_G(v) \setminus \{u\}\big) \cap I$, $\textsc{CheckRigid}(H(G', w),\, I \cap H(G', w),\, w)$ outputs NO, we still need to check if Lemma 4.3(ii) holds. If Lemma 4.3(ii) does not hold for all component $H(G', w)$ satisfying $N_G(v) \cap H(G', w) = \{w, w'\}$, then we can output NO and a TS-sequence $\mathcal{S}_u$ that slides t to v by first moving all t_w to a vertex in $N_{H(G',w)}(w)$ (no token is slid to w' during this process) and slide t to v.

We now describe how to check if Lemma 4.3(ii) holds. Let C be the (unique) cycle in G (of length k) containing v, w (and also w'). Let $P = C - v = p_1 p_2 \dots p_{k-1}$ with $p_1 = w, p_{k-1} = w'$. Clearly, P is an induced path in G'. By the definition of G', it follows that $u \notin V(C) \subseteq V(G') \cup \{v\}$. Note that for $x \in V(C) \setminus \{v\} = V(P)$, the graph G^x_C is a subgraph of $H(G', w)$. If $|I \cap P| < \lfloor k/2 \rfloor$, Lemma 4.3(ii) clearly does not hold. If k is even then it also does not hold, since t_w is not $(H(G', w), I \cap H(G', w))$-rigid. Thus, we now consider the case k is odd and $|I \cap P| = \lfloor k/2 \rfloor$. First, we call $\textsc{CheckRigid}(G^x_C,\, I \cap G^x_C,\, x)$ for every $x \in I \cap P$. If there exists a vertex $x \in I \cap P$ such that the token t_x placed at x is $(G^x_C, I \cap G^x_C)$-movable, we can conclude that Lemma 4.3(ii) does not hold. The reason is that by moving t_x to a vertex in G^x_C, we also obtain an independent set I' satisfying $I \cap G' \overset{G'}{\leftrightsquigarrow} I'$ and $|I' \cap P| < \lfloor k/2 \rfloor$ (see Proposition 2.5). Thus, we can now consider the case when all t_x $(x \in I \cap P)$ are $(G^x_C, I \cap G^x_C)$-rigid. Note that from Lemma 4.3 and the assumption that t_w (and $t_{w'}$ if $w' \in I$) is $(H(G', w), I \cap H(G', w))$-movable, it follows that for every $x \in I \cap P$, t_x must be $(H(G', w), I \cap H(G', w))$-movable, and thus $(G', I \cap G')$-movable. Thus, one can now apply Lemma 4.4. One can construct the independent sets I'_1, I'_2 described in Lemma 4.4 from $I \cap G'$ by sliding tokens in G' (which can also be extended to a TS-sequence in G) as follows. Let i be the smallest index such that $p_i \in I \setminus I'_1$. From the definition of $I'_1 \cap P$, i must be even. Since $I \cap P$ is a maximum independent set of P, it follows that $p_j \in I'_1$ for j odd, $j \leq i - 1$, and $p_j \in I \setminus I'_1$ for j even, $j \geq i$. Since the token t_{p_i} on p_i is $(G', I \cap G')$-movable but

$(G_P^{p_i}, I \cap G_P^{p_i})$-rigid, it can only be slid to p_{i-1}. In other words, there exists a TS-sequence $\mathcal{S}_{p_i}$ in G' that slides t_{p_i} to p_{i-1}. Note that $\mathcal{S}_{p_i}$ can be constructed recursively as follows. From Lemma 4.3, if $\big(N_{G'}(p_{i-1}) \setminus \{p_i\}\big) \cap I = \emptyset$, $\mathcal{S}_{p_i}$ contains only a single step of sliding t_{p_i} to p_{i-1}. On the other hand, if $\big(N_{G'}(p_{i-1}) \setminus \{p_i\}\big) \cap I \neq \emptyset$, there must be a TS-sequence $\mathcal{S}'_{p_i}$ in $G'' = G' - N_{G'}[p_i]$ that slides any token in $\big(N_{G'}(p_{i-1}) \setminus \{p_i\}\big) \cap I$ to some vertex not in $N_{G'}(p_{i-1}) \setminus \{p_i\}$ without moving a new token to $N_{G'}(p_{i-1}) \setminus \{p_i\}$ beforehand. From Proposition 2.4, $\mathcal{S}'_{p_i}$ can be extended to a TS-sequence in G'. Hence, $\mathcal{S}_{p_i}$ is constructed by simply performing $\mathcal{S}'_{p_i}$ first, then performing a single sliding step which moves t_{p_i} to p_{i-1}. Repeat the described steps, we finally obtain an independent set I'_1 which satisfies $I \cap G' \overset{G'}{\leftrightsquigarrow} I'_1$ and $I'_1 \cap P = \{p_1, p_3, \dots, p_k\}$. Note that the recursive construction of $\mathcal{S}_{p_i}$ is indeed included in the results of calling CheckRigid(G', $I \cap G'$, p_i) (must return no and $\mathcal{S}_{p_i}$), and $\mathcal{S}_{p_i}$ only involves t_{p_i} and the tokens in $I \cap G_C^{p_{i-1}}$. A similar procedure can be applied for constructing I'_2. Once we constructed I'_1 and I'_2, by Lemma 4.4, we only need to call CheckRigid(G_C^y, $I'_i \cap G_C^y$, y) for all $y \in P \cap (I'_i \setminus I)$ ($i \in \{1, 2\}$). If all of them outputs yes, we conclude that Lemma 4.3(ii) holds.

Next, we analyze the time complexity of our algorithm. Note that the time complexity of this recursive algorithm is proportional to the number of calls of the CheckRigid function. Observe that for every $u \in V(G)$, the function CheckRigid is called for u at most once during the process of checking Lemma 4.3(i). Now, consider the process of checking Lemma 4.3(ii). For each vertex $v \in V(G_C^x)$ ($x \in V(P) = V(C - v)$), because of the definitions of I'_1, I'_2, CheckRigid is called for v at most twice: at most once for the construction of either I'_1 or I'_2, and at most once for checking the conditions described in Lemma 4.4. Thus, for every vertex $u \in V(G)$, CheckRigid is called for u at most three times. Hence, it takes $O(n)$ time to check if a token is (G, I)-rigid. Therefore, $\mathsf{R}(G, I)$ can be computed in $O(n^2)$ time. ☐

We now characterize (G, I)-confined cycles. Analogously to the case of confined (induced) paths (Lemma 4.4), one can also derive (using Proposition 2.5) that if a cycle C is of even length k, then it is (G, I)-confined if and only if $I \cap C$ is a maximum independent set of C and any token placed at $x \in I \cap C$ is (G, I)-rigid. We now investigate the case when k is odd.

Lemma 4.6. *Let $C = c_1c_2 \dots c_kc_1$ be a cycle of a cactus G. Let I be an independent set of G satisfying that $I \cap C$ is a maximum independent set of C. Assume that for every $x \in I \cap C$, the token placed at x is (G, I)-movable.*

Then, C is (G, I)-confined if and only if k is odd and there exist three independent sets I'_1, I'_2 and I'_3 such that

(i) $I \overset{G}{\leftrightsquigarrow} I'$, where $I' \in \{I, I'_1, I'_2, I'_3\}$;

(ii) $I'_1 \cap C = \{c_1, c_3, \dots, c_{k-2}\}$, $I'_2 \cap C = \{c_2, c_4, \dots, c_{k-1}\}$, and $I'_3 \cap C = \{c_3, c_5, \dots, c_k\}$; and

(iii) for every $x \in I' \cap C$, the token placed at x is $(G_C^x, I' \cap G_C^x)$-rigid.

Proof. First, we show the if direction. Assume that k is odd and the described independent sets I'_1, I'_2, I'_3 exist. As in Lemma 4.4, it suffices to show that for every J with $I \overset{G}{\leftrightsquigarrow} J$ and for $x \in J \cap C$, the token placed at x is $(G_C^x, J \cap G_C^x)$-rigid. For $i \in \{1, 2, 3\}$, let

$x \in J \cap I'_i \cap C$ for some J such that $I \overset{G}{\leftrightsquigarrow} J$ and assume that the token t_x placed at x is $(G^x_C, I'_i \cap G^x_C)$-rigid. Using a similar argument as in the proof of Lemma 4.4, one can show that t_x is also $(G^x_C, J \cap G^x_C)$-rigid. Moreover, for every J with $I \overset{G}{\leftrightsquigarrow} J$, $J \cap C = \bigcup_{i=1}^{3}(J \cap I'_i \cap C)$. Hence, for every $x \in J \cap C$, the token placed at x is $(G^x_C, J \cap G^x_C)$-rigid.

It remains to show the only-if direction. Assume that C is (G, I)-confined. Since $I \cap C$ is a maximum independent set of C and any token placed at $x \in I \cap C$ is (G, I)-movable, it follows that k must be odd. The construction of I'_1 and I'_2 can be done similar as in the proof of Lemma 4.4. For constructing I'_3, instead of starting from I, we start from I'_1 as the only TS-sequence we need is the one that slides the token at c_1 to c_k, which can be obtained from the result of checking if the token placed at c_1 is (G, I'_1)-rigid. $\square$

In the next lemma, we claim that, given $\mathsf{R}(G, I) = \emptyset$, one can compute $\mathsf{C}(G, I)$ in polynomial time.

Lemma 4.7. *Let G be a cactus. Let I be an independent set of G. Assume that $\mathsf{R}(G, I) = \emptyset$. Then for every cycle C in G, one can decide if C is (G, I)-confined in $O(n)$ time, where $n = |G|$. Consequently, computing $\mathsf{C}(G, I)$ takes $O(n^2)$ time.*

Proof. We describe a recursive algorithm that will return YES if C is (G, I)-confined and return NO otherwise.

The idea of our algorithm comes from Lemma 4.6. Let k be the length of C. If k is even or $|I \cap C| < \lfloor k/2 \rfloor$ then clearly we can return NO. Otherwise, for each $x \in I \cap C$, we first check if the token t_x placed at x is $(G^x_C, I \cap G^x_C)$-rigid. If at least one of them is not $(G^x_C, I \cap G^x_C)$-rigid, then we can return NO, because some token t_x can be slid to a vertex in G^x_C. Otherwise, we call the CHECKRIGID(G, I, x) function for each vertex $x \in I \cap C$. Since $\mathsf{R}(G, I) = \emptyset$, it must return NO and a TS-sequence which then can be used for constructing the described sets I'_1, I'_2 and I'_3 in Lemma 4.6. For constructing I'_3, we start from I'_1 instead of I and hence need to check if the token placed at c_1 is (G, I'_1)-rigid or not beforehand. Next, after constructing these three independent sets, we check for all $y \in C \cap (I'_i \setminus I)$ ($i \in \{1, 2, 3\}$) whether the token t_y placed at y is $(G^y_C, I'_i \cap G^y_C)$-rigid. If all of such t_y are $(G^y_C, I'_i \cap G^y_C)$-rigid, by Lemma 4.6, we return YES. Otherwise, we simply return NO.

We now analyze the time complexity of the described algorithm. Note that, for each vertex $u \in V(G)$, the CHECKRIGID function is called for u at most three times: at most once during the process of checking if it is (G, I)-rigid (and should return NO because of our assumption) and constructing I'_1, I'_2; at most once during the process of checking if the token placed at c_1 is (G, I'_1)-rigid and constructing I'_3; and at most once during the process of checking the conditions described in Lemma 4.6. Thus, it takes $O(n)$ time to decide if a cycle C is (G, I)-confined. Consequently, computing $\mathsf{C}(G, I)$ takes $O(n^2)$ time. $\square$

4.3 SLIDING TOKEN for cactus graphs

In this subsection, we design a polynomial-time algorithm for solving SLIDING TOKEN for cactus graphs and prove its correctness. Let (G, I, J) be an instance of SLIDING TOKEN where G is a cactus and I, J are two independent sets of G. The following algorithm decides if $I \overset{G}{\leftrightsquigarrow} J$.

- **Step 1:**
  - **Step 1-1:** If $\mathsf{R}(G, I) \neq \mathsf{R}(G, J)$, return NO.
  - **Step 1-2:** Otherwise, remove all vertices in $N_G[\mathsf{R}(G, I)]$. Let G' be the resulting graph, and go to **Step 2**.
- **Step 2:**
  - **Step 2-1:** If $\mathsf{C}(G', I \cap G') \neq \mathsf{C}(G', J \cap G')$, return NO.
  - **Step 2-2:** Otherwise, remove all cycles in $\mathsf{C}(G', I \cap G')$. Let G'' be the resulting graph, and go to **Step 3**.
- **Step 3:** If $|I \cap F| \neq |J \cap F|$ for some component F of G'' then return NO. Otherwise, return YES.

We now estimate the running time of this algorithm. Lemma 4.5 ensures that **Step 1-1** can be performed in $O(n^2)$ time. **Step 1-2** clearly can be performed in $O(n)$ time. Thus, **Step 1** takes $O(n^2)$ time. **Step 2** also takes $O(n^2)$ time because by Lemma 4.7, **Step 2-1** takes $O(n^2)$ time, and **Step 2-2** can be performed in $O(n)$ time. Finally, **Step 3** clearly runs in $O(n)$ time. In total, the algorithm runs in $O(n^2)$ time.

It remains to show the correctness of our algorithm. An immediate observation is that the correctness of **Step 1-1** and **Step 2-1** are direct consequences of Proposition 2.1 and Proposition 2.5. Thus, it remains to show the correctness of **Step 1-2**, **Step 2-2**, and **Step 3**.

First, we prove an useful lemma.

Lemma 4.8. *Let I be an independent set of a cactus G. Let $v \notin I$. Assume that $\mathsf{R}(G, I) = \emptyset$, and $N_G(v) \cap I \neq \emptyset$. Then, there is at most one $(G', I \cap G')$-rigid token in $N_G(v) \cap I$, where $G' = G - v$. On the other hand, if there exists a cycle C containing v such that the path $P = C - v$ is $(G', I \cap G')$-confined, then all tokens in $N_G(v) \cap I$ must be $(G', I \cap G')$-movable. Moreover, if $\mathsf{C}(G, I) = \emptyset$ then there is at most one cycle C with the described property.*

Proof. Suppose to the contrary that there are two vertices w and w' in $N_G(v) \cap I$ such that the tokens t_w and $t_{w'}$ placed at w and w', respectively, are both $(G', I \cap G')$-rigid (see Figure 4.2(a)). From the assumption, t_w and $t_{w'}$ must be (G, I)-movable. Therefore, t_w (at least) can be slid to v. However, this can happen only when $t_{w'}$ can be slid to a vertex in $N_{G'}(w')$, i.e., $t_{w'}$ is $(G', I \cap G')$-movable, which contradicts our assumption. Hence, there is at most one $(G', I \cap G')$-rigid token in $N_G(v) \cap I$.

Assume that there exists a cycle C containing v such that the path $P = C - v$ is $(G', I \cap G')$-confined. By Proposition 2.5, for every independent set I' with $I \cap G' \overset{G'}{\leftrightsquigarrow} I'$, $|I \cap P| = \lfloor k/2 \rfloor$, where k is the length of C. Hence, for every $x \in I \cap C$, the token on x is at least $(G^x_C, I \cap G^x_C)$-rigid. Hence, if k is even, it follows that no token can be slid (in G) along edges of C, i.e., all tokens in $I \cap C$ are (G, I)-rigid, which is a contradiction. Therefore, k must be odd. It follows that the tokens in $N_G(v) \cap I \cap C$ must be $(G', I \cap G')$-movable. Now, suppose to the contrary that the token $t_{w'}$ at some vertex $w' \in (N_G(v) \cap I) - C$ is $(G', I \cap G')$-rigid. Since $t_{w'}$ is (G, I)-movable, it can at least be slid to v, which contradicts Lemma 4.3(ii). Hence, all tokens in $N_G(v) \cap I$ must be $(G', I \cap G')$-movable.

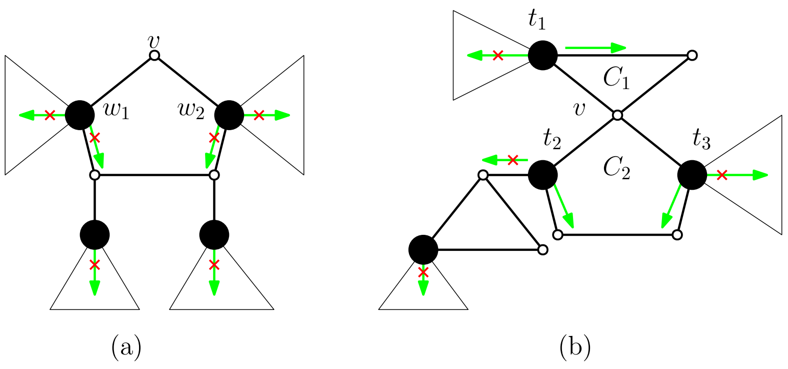


Figure 4.2: Illustration for Lemma 4.8.

Finally, we claim that if $\mathsf{C}(G, I) = \emptyset$ then there are at most one cycle C containing v such that the path $P = C - v$ is $(G', I \cap G')$-confined. Suppose to the contrary that there are two cycles C_1 and C_2 satisfy the described property (see Figure 4.2(b)). For $i \in \{1, 2\}$, since $v \notin I$ and $I \cap (C_i - v)$ is a maximum independent set of $C_i - v$, it follows that $I \cap C_i$ is a maximum independent set of C_i. Additionally, note that $\mathsf{C}(G, I) = \emptyset$. Thus, no $(G, I, V(C_i))$-confined token ($i \in \{1, 2\}$) is placed at any vertex of $I \cap C_i$. From the assumption, all tokens in $I \cap (C_i - v) = I \cap C_i$ are $(G, I, V(C_i - v))$-confined. On the other hand, since $I \cap C_1$ is a maximum independent set of C_1, there exists a token t_1 at some vertex $v_1 \in N_{C_1}(v)$. As before, t_1 must be $(G, I, V(C_1 - v))$-confined and not $(G, I, V(C_1))$-confined. Therefore, it can be slid to v. Similarly, there exists a token t_2 at some vertex $v_2 \in N_{C_2}(v)$ such that t_2 is $(G, I, V(C_2 - v))$-confined and not $(G, I, V(C_2))$-confined. Clearly, t_2 must also be slid to v, but this is a contradiction since one needs to slide t_1 to a vertex not in $N_G(v)$ first, which can be done (at least) when t_2 has been moved. Note that since $I \cap C_2$ is a maximum independent set of C_2, there always exists some token in $N_{C_2}(v)$ while no token in $I \cap C_2$ is moved to a vertex not in $V(C_2)$. Therefore, there is at most one cycle C containing v such that the path $P = C - v$ is $(G', I \cap G')$-confined. $\square$

The next lemma ensures the correctness of **Step 1-2** and **Step 2-2**.

Lemma 4.9. *Suppose that $\mathsf{R}(G, I) = \mathsf{R}(G, J)$ for two given independent sets I and J of a cactus G, and let G' be the graph obtained from G by deleting the vertices in $N_G[\mathsf{R}(G, I)] = N_G[\mathsf{R}(G, J)]$. Then $I \overset{G}{\leftrightsquigarrow} J$ if and only if $I \cap G' \overset{G'}{\leftrightsquigarrow} J \cap G'$. Furthermore, $\mathsf{R}(G', I \cap G') = \mathsf{R}(G', J \cap G') = \emptyset$.*

Suppose that $\mathsf{C}(G', I \cap G') = \mathsf{C}(G', I \cap G') \neq \emptyset$. Let G'' be the graph obtained by removing all cycles in $\mathsf{C}(G', I \cap G')$. Then $I \cap G' \overset{G'}{\leftrightsquigarrow} J \cap G'$ if and only if $I \cap G'' \overset{G''}{\leftrightsquigarrow} J \cap G''$. Furthermore, $\mathsf{R}(G'', I \cap G'') = \mathsf{R}(G'', J \cap G'') = \emptyset$ and $\mathsf{C}(G'', I \cap G'') = \mathsf{C}(G'', J \cap G'') = \emptyset$.

Proof. Assume that there exists a TS-sequence $\mathcal{S} = \langle I = I_1, I_2, \dots, I_r = J \rangle$ in G that reconfigures I to J, and $\mathsf{R}(G, I) = \mathsf{R}(G, J)$. We show that $I \cap G' \overset{G'}{\leftrightsquigarrow} J \cap G'$. Since no tokens can be placed at any neighbor of $\mathsf{R}(G, I) = \mathsf{R}(G, J) = \mathsf{R}(G, I_i)$ ($i \in \{1, 2, \dots, r\}$), for every independent set I of G, $I \setminus \mathsf{R}(G, I)$ is indeed an independent set of G'. For $i \in \{2, \dots, r\}$, let $I_{i-1} \setminus I_i = \{u\}$ and $I_i \setminus I_{i-1} = \{v\}$, for some edge $uv \in E(G)$. Since $u \notin I_i$ and $v \notin I_{i-1}$, both u and v are not in $\mathsf{R}(G, I)$. Thus, $uv \in E(G')$. Therefore, $\mathcal{S}' = \langle I_1 \setminus \mathsf{R}(G, I), I_2 \setminus \mathsf{R}(G, I), \dots, J \setminus \mathsf{R}(G, I) \rangle$ is a TS-sequence in G' that reconfigures $I \setminus \mathsf{R}(G, I) = I \cap G'$ to $J \setminus \mathsf{R}(G, I) = J \cap G'$.

Assume that there exists a TS-sequence $\mathcal{S}' = \langle I'_1 = I \cap G', I'_2, \dots, I'_s = J \cap G'\rangle$ in G' that reconfigures $I \cap G'$ to $J \cap G'$. By definition of G', it follows that for every independent set I' of G', $I' \cup \mathsf{R}(G, I)$ forms an independent set of G. Hence, $\mathcal{S} = \langle I'_1 \cup \mathsf{R}(G, I), I'_2 \cup \mathsf{R}(G, I), \dots, I'_s \cup \mathsf{R}(G, I)\rangle$ is a TS-sequence that reconfigures $I'_1 \cup \mathsf{R}(G, I) = I$ to $I_s \cup \mathsf{R}(G, I) = J$.

We now show that $\mathsf{R}(G', I \cap G') = \emptyset$. Let $v \in I \cap G'$. Then, the token t_v placed at v is (G, I)-movable, because otherwise $v \in \mathsf{R}(G, I)$. Hence, there exists a TS-sequence $\mathcal{S}$ in G that slides t_v to a vertex $w \in N_G(v)$. Note that $w \in V(G')$. As before, from $\mathcal{S}$, one can construct a TS-sequence $\mathcal{S}'$ in G' that slides t_v to w, hence it implies t_v is $(G', I \cap G')$-movable. Therefore, $\mathsf{R}(G', I \cap G') = \emptyset$. Similarly, one can also show that $\mathsf{R}(G', J \cap G') = \emptyset$.

Suppose that $\mathsf{C}(G', I \cap G') = \mathsf{C}(G', I \cap G') \neq \emptyset$ and there exists a TS-sequence $\mathcal{S}' = \langle I'_1 = I \cap G', I'_2, \dots, I'_s = J \cap G'\rangle$ in G' that reconfigures $I \cap G'$ to $J \cap G'$. For $j \in \{2, \dots, s\}$, let $I'_{j-1} \setminus I'_j = \{u\}$ and $I'_j \setminus I'_{j-1} = \{v\}$ for some edge $uv \in E(G')$. Since all tokens in $I \cap C$ are $(G', I \cap G', V(C))$-confined, u and v must be either both in G'' or both in some cycle $C \in \mathsf{C}(G', I \cap G')$. Hence, $\mathcal{S}'' = \langle I'_1 \cap G'' = I \cap G'', I'_2 \cap G'', \dots, I'_s \cap G'' = J \cap G''\rangle$ is a TS-sequence in G'' that reconfigures $I \cap G''$ to $J \cap G''$.

Assume that there exists a TS-sequence $\mathcal{S}'' = \langle I''_1 = I \cap G'', I''_2, \dots, I''_t = J \cap G''\rangle$ in G'' that reconfigures $I \cap G''$ to $J \cap G''$. We claim that one can construct a TS-sequence $\mathcal{S}'$ in G' that reconfigures $I \cap G' = (I \cap G'') \cup (I \cap \mathsf{C}(G', I \cap G'))$ to $J \cap G' = (J \cap G'') \cup (J \cap \mathsf{C}(G', I \cap G'))$. Note that for a given independent set I'' of G'' and a cycle $C \in \mathsf{C}(G', I \cap G')$, $I'' \cup \big(I \cap C\big)$ may not be an independent set of G'. The same observation holds for every independent set reconfigurable from I. Let $\mathcal{F}$ be the set of all components of G''. From the previous part, one can construct a TS-sequence $\mathcal{S}''_F = \langle I''_1 \cap F, I''_2 \cap F, \dots, I''_t \cap F\rangle$ for each component $F \in \mathcal{F}$. Let $A = \bigcup_{C \in \mathsf{C}(G', I \cap G')} \bigcup_{x \in I \cap C} \big(N_{G'}(x) \setminus V(C)\big)$. For a given component F of G'', consider the following cases.

Case 1: $\mathcal{S}''_F$ involves no vertex in A. For an independent set $I_F \in \mathcal{S}''_F$ and a cycle C of G', $I_F \cup (I \cap C)$ forms an independent set of G'. It follows that $\mathcal{S}''_F$ can be extended to a TS-sequence in G'.

Case 2: $\mathcal{S}''_F$ involves vertices in $A' = A \cap F$ (see Figure 4.3).

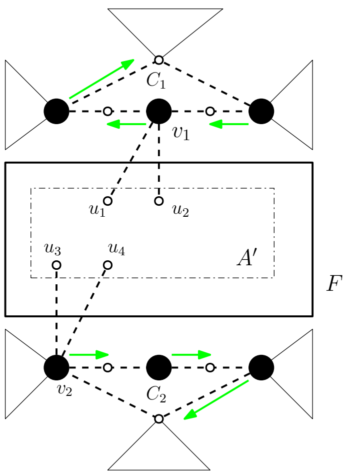


Figure 4.3: $\mathcal{S}''_F$ involves vertices in $A' \subseteq A$ (Lemma 4.9).

Let $C \in \mathsf{C}(G', I \cap G')$. Since G' is a cactus, there is at most one vertex $v \in I \cap C$ such that $N_{G'}(v) \cap V(F) \neq \emptyset$. Moreover, if there are two vertices $u_1, u_2 \in V(F)$ such that $N_{G'}(u_i) \cap V(C) \neq \emptyset$ ($i \in \{1, 2\}$) then they must both adjacent to v. By definition of $(G', I \cap G', V(C))$-confined tokens, for such a cycle C above, there exists a TS-sequence $\mathcal{S}(C, v)$ that slides the token t_v at $v \in I \cap C$ ($N_{G'}(v) \cap V(F) \neq \emptyset$) to some vertex w in $N_C(v)$. Now, if there are two of such cycle C, say C_1 and C_2, let v_1 (resp., v_2) be a vertex in $I \cap C_1$ (resp., $I \cap C_2$) such that $N_{G'}(v_1) \cap V(F) \neq \emptyset$ (resp., $N_{G'}(v_2) \cap V(F) \neq \emptyset$). Since G is a cactus, $V(G^x_{C_1}) \cap V(G^y_{C_2}) = \emptyset$, where $x \in V(C_1) \setminus \{v_1\}$ and $y \in V(C_2) \setminus \{v_2\}$. It follows that $\mathcal{S}(C_1, v_1)$ and $\mathcal{S}(C_2, v_2)$ can be performed independently.

The TS-sequence $\mathcal{S}'$ thus can be constructed as follows. First of all, we perform any sequence $\mathcal{S}''_F$ that does not involve vertices of A. Next, for a component F such that $\mathcal{S}''_F$ involves some vertex of A, let $C \in \mathsf{C}(G', I \cap G')$ be such that there exists a vertex $v \in I \cap C$ satisfying $N_G(v) \cap V(F) \subseteq A$. As observed before, such a vertex v is uniquely determined. Then, we perform $\mathcal{S}(C, v)$, then perform $\mathcal{S}''_F$, and then perform $\mathcal{S}(C, v)$ in reverse order. If the vertex $w \in N_C(v)$ where the token t_v is slid to after performing $\mathcal{S}(C, v)$ is also in J then in the step of reversing $\mathcal{S}(C, v)$, we do not reverse the step of sliding t_v to w. At this moment, we have reconfigured $I \cap G''$ to $J \cap G''$ in G'. The remaining problem is to reconfigure $I \cap C$ to $J \cap C$ in G' for every cycle $C \in \mathsf{C}(G', I \cap G')$. This can be done using Lemma 4.2 and the observation that for every vertex $v \in V(C)$, if $v \in J$ then $N_G(v) \cap J = \emptyset$.

Using a similar argument as before (based on the fact that if I' is an independent set of G' then $I' \cap G''$ is also an independent set of G''), one can show that $\mathsf{R}(G'', I \cap G'') = \mathsf{R}(G'', J \cap G'') = \emptyset$, and $\mathsf{C}(G'', I \cap G'') = \mathsf{C}(G'', J \cap G'') = \emptyset$. $\square$

Before proving the correctness of **Step 3**, we need some extra definitions. Let w be a cut vertex of a cactus G such that $\mathcal{B}_w \neq \emptyset$. For every block $B \in \mathcal{B}_w$, since each block of G is either K_2 or a simple cycle and all blocks in $\mathcal{B}_w$ share the same (unique) cut vertex w, without loss of generality, assume that the vertices of B are labeled as $v_0[B], v_1[B], \dots, v_{|B|-1}[B]$ so that $v_0[B] = w$; $v_i[B]$ is adjacent to $v_{i+1}[B]$, $i \in \{1, 2, \dots, |B| - 2\}$; and $v_0[B]$ is adjacent to $v_{|B|-1}[B]$.

Lemma 4.10. *Let I be an independent set of a given cactus G. Assume that $\mathsf{R}(G, I) = \emptyset$ and $\mathsf{C}(G, I) = \emptyset$. Let w be a cut vertex of G such that $\mathcal{B}_w \neq \emptyset$. Assume that $|I| \geq \sum_{B \in \mathcal{B}_w} \left(\lfloor |B|/2 \rfloor - 1\right)$.*

(i) If $\sum_{B \in \mathcal{B}_w} \left(\lfloor |B|/2 \rfloor - 1\right) = 0$, then there is an independent set I' satisfying that $I \overset{G}{\leftrightsquigarrow} I'$ and $v \in I'$, where $v \in V(\mathcal{B}_w)$ is some safe vertex of G.

(ii) If $\sum_{B \in \mathcal{B}_w} \left(\lfloor |B|/2 \rfloor - 1\right) \geq 1$, then there is an independent set I' satisfying that $I \overset{G}{\leftrightsquigarrow} I'$, $N_{\mathcal{B}_w}(w) \cap I' = \emptyset$, and $|I' \cap (\mathcal{B}_w - w)| = \sum_{B \in \mathcal{B}_w} \left(\lfloor |B|/2 \rfloor - 1\right)$.

Proof. We first prove several useful claims.

Claim 4.10.1. *If $N_{\mathcal{B}_w}(w) \cap I = \emptyset$ then one can slide a closest token in G^* to w, where G^* is the graph obtained from G by removing all vertices in $\mathcal{B}_w - w$. In other words, there exists an independent set J such that $I \overset{G}{\leftrightsquigarrow} J$ and $w \in J$.*

Proof. If $w \in I$ then we are done. Thus, we can assume that $w \notin I$. Let $w' \in I \cap G^*$ be a vertex with $\mathsf{dist}_{G^*}(w, w') = \min_{w'' \in I \cap G^*} \mathsf{dist}_{G^*}(w, w'')$. Let $P = w_1 \dots w_p$ ($p \geq 3$) be a shortest ww'-path with $w_1 = w$ and $w_p = w'$. Let $M = N_{G^*}(w_{p-1}) \cap I$. Since $N_{\mathcal{B}_w}(w) \cap I = \emptyset$, it follows that $M = N_{G^*}(w_{p-1}) \cap I = N_G(w_{p-1}) \cap I$ for $p \geq 3$. The definition of w' implies that no tokens are placed at $N_G[w_i]$ for $i \in \{1, 2, \dots, p-2\}$. We claim that a token on some vertex of M can be slid to w. If $|M| = 1$, i.e., M contains only w', then one can slide (in G) the token on w' to w directly. If $|M| \geq 2$, then by Lemma 4.8, there exists at most one vertex z in M such that the token on z is $(G', I \cap G')$-rigid, where $G' = G - w_{p-1}$ (see Figure 4.4(a)). On the other hand, if there exists a cycle D containing w_{p-1} such that the path $Q = D - w_{p-1}$ is $(G', I \cap G')$-confined, then all tokens in M must be $(G', I \cap G')$-movable (see Figure 4.4(b)). Note that because $\mathsf{C}(G, I) = \emptyset$, such a cycle D above (if exists) must be unique. Also note that by Lemma 4.3 and the assumption that $\mathsf{R}(G, I) = \emptyset$, both z and D cannot exist at the same time. If both of them do not exist, we can slide the token $t_{w'}$ placed at w' to w by first sliding all tokens in $M - w'$ (which are clearly $(G', I \cap G')$-movable) to some vertices in G', and then slide $t_{w'}$ to w. If z exists, we first reduce the number of tokens in M by sliding all tokens in $M - z$ (which are clearly $(G', I \cap G')$-movable) to some vertices in G' (using Lemma 4.5), and then slide the token t_z on z to w. On the other hand, if D exists (uniquely), then one can slide a token $t_{z'}$ on $z' \in M \cap D$ to w by first sliding all tokens in $M - C$ (which are clearly $(G', I \cap G')$-confined) to some vertices in G' (using Lemma 4.7), then sliding $t_{z'}$ to w_{p-1} (which, by Lemma 4.8, is the only way of moving $t_{z'}$ "out of" D), and finally to w.

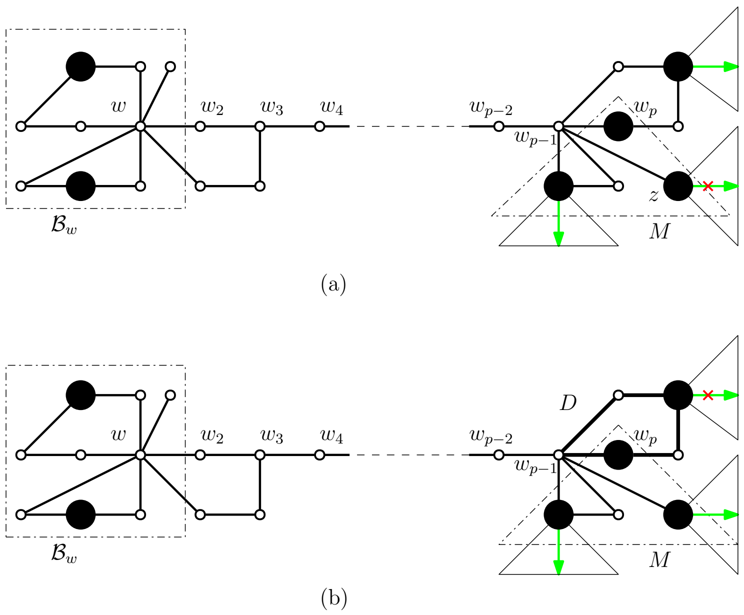


Figure 4.4: (a) The token t_z at z is $(G', I \cap G')$-rigid; (b) The cycle D containing w_{p-1} such that the path $Q = D - w_{p-1}$ is $(G', I \cap G')$-confined.

□

Claim 4.10.2. *The maximum number of tokens that can be placed at vertices of $\mathcal{B}_w$ is $\sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big) + 1$.*

Proof. Observe that for every block $B \in \mathcal{B}_w$, since B is either K_2 or a cycle, $B - w$ is indeed a path. Moreover, the path $P = B - w$ satisfies that any token t_x placed at $x \in I \cap P$ is $(G_P^x, I \cap G_P^x, V(P))$-confined, simply because in this case G_P^x is the graph contains a single vertex x. By Lemma 4.8, there is at most one block $B \in \mathcal{B}_w$ that contains $\lfloor |B|/2 \rfloor$ token(s), while all other blocks $B' \neq B$ must contain at most $\lfloor |B'|/2 \rfloor - 1$ token(s). Thus, $|I \cap \mathcal{B}_w| \leq \sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big) + 1$. □

Claim 4.10.3. *If $|I \cap \mathcal{B}_w| \leq \sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big)$, then one can "arrange" the token(s) in $I \cap \mathcal{B}_w$ such that no token is placed at any vertex in $N_{\mathcal{B}_w}[w]$. More formally, there exists an independent set J such that $I \overset{G}{\leftrightsquigarrow} J$ and $N_{\mathcal{B}_w}[w] \cap J = \emptyset$.*

Proof. If there exists a block $B \in \mathcal{B}_w$ such that $|I \cap B| = \lfloor |B|/2 \rfloor$ then since $|I \cap \mathcal{B}_w| \leq \sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big)$, there must be another block $B' \in \mathcal{B}_w$ where $|B' \cap I| < \lfloor |B'|/2 \rfloor - 1$. Since $\mathsf{R}(G, I) = \emptyset$ and $\mathsf{C}(G, I) = \emptyset$, if $w \notin I \cap B$, one can slide a token from B to w and then slide it to a vertex in B'. On the other hand, if $w \in I \cap B$, we slide the token on w to a vertex in B' (other than w) directly. Since $\mathsf{C}(G, I) = \emptyset$, by Lemma 4.8, at most one such block B exists. Thus, we can now assume that $|I \cap B| \leq \lfloor |B|/2 \rfloor - 1$ for every block $B \in \mathcal{B}_w$. Clearly, a block $B \in \mathcal{B}_w$ contains a token only when $|B| \geq 4$, i.e., it is a cycle of length at least 4. Thus, using Lemma 4.2 and note that all blocks $B \in \mathcal{B}_w$ are safe, one can "arrange" the tokens in each $B \in \mathcal{B}_w$ so that no token is placed at $N_B[w]$. The resulting independent set is our desired set J. □

We now prove Lemma 4.10.

(i) Assume that $\sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big) = 0$. Since $|B| \geq 2$ for every block B of G, it follows that for all $B \in \mathcal{B}_w$, $2 \leq |B| \leq 3$, i.e., B is either K_2 or a cycle of length 3. Clearly, $N_{\mathcal{B}_w}(w) = V(\mathcal{B}_w) \setminus \{w\}$. Now, for a safe vertex $v \in V(\mathcal{B}_w)$, one must have that $v \in N_{\mathcal{B}_w}(w) \subseteq N_G(w)$. If $v \in I$ then we are done. Therefore, assume that $v \notin I$. Note that in this case $|I \cap \mathcal{B}_w| \leq 1$. If $|I \cap \mathcal{B}_w| = 0$ then by Claim 4.10.1, one can slide a token to w, and then to v. Otherwise, if $w \in I$, then clearly the token placed at w can be slid to v. On the other hand, if there is a vertex $v' \notin \{v, w\}$ where $v' \in I \cap \mathcal{B}_w$ then since $\mathsf{R}(G, I) = \emptyset$ and $\mathsf{C}(G, I) = \emptyset$, it follows that the token placed at v' can be slid to a vertex outside the block containing v' and w, therefore must be slid to w (which is the unique cut vertex of G in $\mathcal{B}_w$), and then can be slid to v from w.

(ii) Assume that $\sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big) \geq 1$. If $|I \cap \mathcal{B}_w| = \sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big)$ then we can just simply use Claim 4.10.3 to "arrange" the tokens in $I \cap \mathcal{B}_w$. If $|I \cap \mathcal{B}_w| = \sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big) + 1$ then there must exist a unique token t in $N_{\mathcal{B}_w}[w]$ which cannot be "arranged" using Claim 4.10.3. Note that in this case $|I \cap (\mathcal{B}_w - w)| = \sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big)$. If t is placed at w then $N_{\mathcal{B}_w}(w) \cap I = \emptyset$ and we are done. If t is placed at some vertex in $N_{\mathcal{B}_w}(w)$ then it can be slid to w because $\mathsf{R}(G, I) = \emptyset$ and $\mathsf{C}(G, I) = \emptyset$. By sliding t to w, there is now no token placed at any vertex in $N_{\mathcal{B}_w}(w)$, and the resulting independent set is the set I' we need. It remains to consider the case $|I \cap \mathcal{B}_w| < \sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big)$. We claim that one can construct an independent set I' such that $I \overset{G}{\leftrightsquigarrow} I'$, $N_{\mathcal{B}_w}(w) \cap I' = \emptyset$, and $|I' \cap (\mathcal{B}_w - w)| = \sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big)$. Using Claim 4.10.3, we can assume without loss of generality that $N_{\mathcal{B}_w}[w] \cap I = \emptyset$. We construct the set I' using TS rule as follows. While the number of tokens in $\mathcal{B}_w - w$ is smaller than $\sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big)$, we use Claim 4.10.1 to move some token t not in $\mathcal{B}_w - w$ to w, then move t to some block $B \in \mathcal{B}_w$ which contains less than $\lfloor |B|/2 \rfloor - 1$ token(s), then using Claim 4.10.3 to "arrange" the set of tokens in $\mathcal{B}_w$ so that $N_{\mathcal{B}_w}[w]$ contains no token. Repeat the steps above until the number of tokens in $\mathcal{B}_w$ is equal to $\sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big)$, we finally obtain I'.

□

Lemma 4.11. *Let I be an independent set of a given cactus G. Assume that $\mathsf{R}(G, I) = \emptyset$, and $\mathsf{C}(G, I) = \emptyset$. Let w be a cut vertex of G such that $\mathcal{B}_w \neq \emptyset$.*

(i) When $\sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big) = 0$. Let $v \in V(\mathcal{B}_w)$ be a safe vertex of G. Suppose that $v \in I$. Then, $\mathsf{R}(G^*, I^*) = \emptyset$, where G^* is the graph obtained from G by removing all vertices in $\mathcal{B}_w$ and $I^* = I \cap G^*$. Moreover, $\mathsf{C}(G^*, I^*) = \emptyset$.

(ii) When $\sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big) \geq 1$. Assume that $I \cap (\mathcal{B}_w - w) = I \cap \bigcup_{B \in \mathcal{B}_w} \{v_i[B] : 2 \leq i \leq |B| - 2$, and i is even$\}$. Let G^* be the graph obtained from G by removing all vertices in $N_G[I \cap (\mathcal{B}_w - w)]$ and $I^* = I \cap G^*$. Then $\mathsf{R}(G^*, I^*) = \emptyset$ and $\mathsf{C}(G^*, I^*) = \emptyset$.

Proof. We prove the lemma using case-analysis.

(i) First of all, we claim that $\mathsf{R}(G^*, I^*) = \emptyset$. Suppose to the contrary that $\mathsf{R}(G^*, I^*) \neq \emptyset$. Let $w' \in I^*$ be a vertex where a (G^*, I^*)-rigid token is placed. Let $P = w_1 w_2 \dots w_p$ be a vw'-path with $w_1 = v$, $w_2 = w$ and $w_p = w'$.

Case (i-1): $w_{p-1} = w$. (See Figure 4.5.) In this case, it is clear that $\mathsf{dist}_G(w, w_p) = 1$. From Lemma 4.10, any block $B \in \mathcal{B}_w$ is either K_2 or a cycle of length 3. Let B be the safe block containing v. If B is K_2 then clearly the token t_v placed at v is $(G - w, I \cap (G - w))$-rigid. On the other hand, if B is a cycle of length 3 then the path $B - w$ is clearly $(G - w, I \cap (G - w))$-confined. By Lemma 4.8, in any of these two cases, the token t_{w_p} placed at $w_p = w_3 \in N_G(w)$ must be $(G - w, I \cap (G - w))$-movable. By definition, G^* is indeed a connected component of $G - w$ and $I^* = I \cap G^* = (I - v) \cap (G - w)$. Hence, t_{w_p} must be $(G^*, I \cap G^*)$-movable, which is a contradiction.

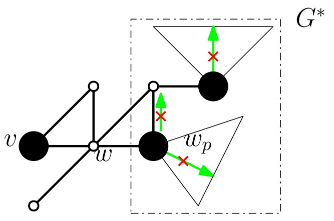


Figure 4.5: Illsutration of **Case (i-1)** of Lemma 4.11(i).

Case (i-2): $w_{p-2} = w$. (See Figure 4.6.) In this case, we can assume that any (G^*, I^*)-rigid token is of distance (in G) at least 2 from w (which then implies $\mathsf{dist}_G(w, w_p) = 2$ in this case); otherwise, we back to **Case (i-1)** and show that there must be some contradiction.

Before analyzing **Case (i-2)**, we show some useful claims.

Claim 4.11.1. *There exists a vertex w_p above such that there is no cycle C_1 in G^* such that $w_{p-1} \in V(C_1)$, $w_p \notin V(C_1)$, and the path $P_1 = C_1 - w_{p-1}$ is $(G^* - N_{G^*}[w_p], I^* \cap (G^* - N_{G^*}[w_p]))$-confined.*

Proof. Suppose that such C_1 exists. Let $H(G^* - N_{G^*}[w_p], P_1)$ be the component of $G^* - N_{G^*}[w_p]$ containing P_1. Since G is a cactus, it follows that $N_G(w) \cap H(G^* - N_{G^*}[w_p], P_1) = \emptyset$. Hence, $H(G^* - N_{G^*}[w_p], P_1)$ must also be a component of $G - N_G[w_p]$. Therefore, C_1 satisfies that $w_{p-1} \in V(C_1)$, $w_p \notin V(C_1)$, and the path $P_1 = C_1 - w_{p-1}$ is $(G - N_G[w_p], I \cap (G - N_G[w_p]))$-confined. It follows that the token t_{w_p} placed at w_p cannot be slid in

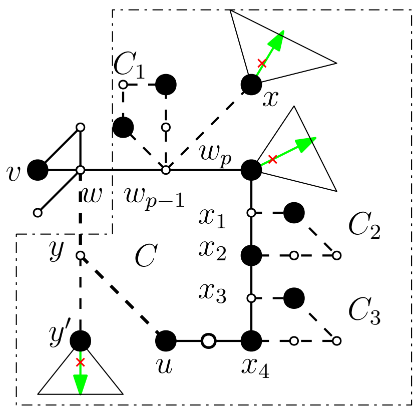


Figure 4.6: Illsutration of **Case (i-2)** of Lemma 4.11(i).

G to w_{p-1}. Note that Lemma 4.8 implies that C_1 is uniquely determined. Since t_{w_p} is (G, I)-movable, it follows that there exists a vertex $x_1 \in N_G(w_p) \setminus \{w_{p-1}\}$ such that t_{w_p} can be slid in G to x_1. Since t_{w_p} is (G^*, I^*)-rigid, it follows that $\big(N_{G^*}(x_1) \setminus \{w_p\}\big) \cap I^* = \big(N_G(x_1) \setminus \{w_p\}\big) \cap I \neq \emptyset$.

Let $x_2 \in N_{G^*}(x_1) \setminus \{w_p\}\big) \cap I^*$. Now, if there exists a cycle C_2 in G^* such that $\{x_1, x_2\} \subseteq V(C_2)$, $w_p \notin V(C_2)$, and the path $P_2 = C_2 - x_1$ is $(G^* - N_{G^*}[w_p], I^* \cap (G^* - N_{G^*}[w_p]))$-confined, then using the same argument as with P_1, it follows that t_{w_p} cannot be slid in G to x_1, which contradicts our assumption. Therefore, for $x_2 \in N_{G^*}(x) \setminus \{w_p\}\big) \cap I^*$, such a cycle C_2 does not exist.

Hence, there must be some $x_2 \in N_{G^*}(x) \setminus \{w_p\}\big) \cap I^*$ such that the token t_{x_2} placed at x_2 must be $(G^* - N_{G^*}[w_p], I^* \cap (G^* N_{G^*}[w_p]))$-rigid, and hence also (G^*, I^*)-rigid since t_{w_p} is also (G^*, I^*)-rigid. On the other hand, since t_{x_2} is (G, I)-movable, it follows that the component $H(G^* - N_{G^*}[w_p], x_2)$ of $G^* - N_{G^*}[w_p]$ containing x_2 must not be a component of $G - N_G[w_p]$, which then implies that $w \in V(H(G - N_G[w_p], x_2))$, where $H(G - N_G[w_p], x_2)$ is the component of $G - N_G[w_p]$ containing x_2. Hence, there exists a cycle C in G containing w, w_{p-1}, w_p, x_1 and x_2. As G is a cactus, the cycle C is unique.

Let $x_3 \neq x_1$ be another neighbor of x_2 in C. Using a similar argument as with C_1, one can show that there does not exist any cycle C_3 in G^* such that $x_3 \in V(C_3)$, $x_2 \notin V(C_3)$, and the path $P_3 = C_3 - x_3$ is $(G^* - N_{G^*}[y], I^* \cap (G^* - N_{G^*}[x_2]))$-confined. Note that in such cycle C_3 above, $V(C_3) \cap V(C) = \{x_3\}$. Hence, there must be some $x_4 \in \big(N_{G^*}(x_3) \setminus \{x_2\}\big) \cap I^*$ such that the token t_{x_4} placed at x_4 is $(G^* - N_{G^*}[x_2], I^* \cap (G^* - N_{G^*}[x_2]))$-rigid, and hence (G^*, I^*)-rigid as t_{x_2} is also (G^*, I^*)-rigid. On the other hand, since t_{x_4} is (G, I)-movable, it follows that the component $H(G^* - N_{G^*}[x_2], x_4)$ of $G^* - N_{G^*}[x_2]$ containing x_4 must not be a component of $G - N_G[x_2]$, which then implies that $w \in V(H(G - N_G[x_2], x_4))$, where $H(G - N_G[x_2], x_4)$ is the component of $G - N_G[x_2]$ containing x_4. Since G is a cactus, it must happen that $x_4 \in V(C)$. Repeat the arguments with vertices of C, we finally obtain that there must be some (G^*, I^*)-rigid token placed at a vertex $u \in V(C)$ of distance 1 or 2 from w (in G). Since $\mathsf{dist}_G(w, w_p) = 2$ and t_{w_p} is a closest (G^*, I^*)-rigid token to w, no (G^*, I^*)-rigid token can be placed at some vertex of distance 1 from w. Thus, $\mathsf{dist}_G(w, u) = 2$. Therefore, we can now simply regard u as w_p. $\square$

Claim 4.11.2. *Assume that w_p satisfies Claim 4.11.1. Then, there exists a (unique) cycle C in G containing w and w_p.*

Proof. Since t_{w_p} is (G^*, I^*)-rigid and C_1 does not exist, there must be some vertex $x \in \big(N_{G^*}(w_{p-1}) \setminus \{w_p\}\big) \cap I^*$ such that the token t_x placed at x is $(G^* - N_{G^*}[w_p], I^* \cap (G^* - N_{G^*}[w_p]))$-rigid, and hence also (G^*, I^*)-rigid as t_{w_p} is (G^*, I^*)-rigid. Thus, both t_{w_p} and t_x are $(G^* - w_{p-1}, I^* \cap (G^* - w_{p-1}))$-rigid. Since all tokens in I are (G, I)-movable and $w_{p-1} \notin I$, Lemma 4.8 implies that at most one of the two tokens t_{w_p} and t_x is $(G - w_{p-1}, I \cap (G - w_{p-1}))$-rigid. Without loss of generality, assume t_{w_p} is not $(G - w_{p-1}, I \cap (G - w_{p-1}))$-rigid. Hence, it must happen that $w \in V(H(G - w_{p-1}, w_p))$, where $H(G - w_{p-1}, w_p)$ is the component of $G - w_{p-1}$ containing w_p. Thus, there exists a (unique) cycle C in G containing w and w_p. □

We now consider **Case (i-2)**. Let $H(G^* - w_{p-1}, x)$ and $H(G^* - w_{p-1}, w_p)$ be the components of $G^* - w_{p-1}$ containing x and w_{p-1}, respectively. As $H(G^* - w_{p-1}, w_p)$ is not a component of $G - w_{p-1}$, it follows that $H(G^* - w_{p-1}, x)$ is a component of $G - w_{p-1}$, i.e., $H(G^* - w_{p-1}, x) = H(G - w_{p-1}, x)$ because if otherwise, $w \in V(H(G - w_{p-1}, x))$, which contradicts to the fact that G is a cactus. Hence, t_x is indeed $(G - w_{p-1}, I \cap (G - w_{p-1}))$-rigid, which means that t_{w_p} cannot be slid in G to w_{p-1}.

Let $x_1 \in N_G(w_p) \setminus \{w_{p-1}\}$ be a neighbor of w_p such that t_{w_p} can be slid in G to x_1. If $x_1 \notin V(C)$ then since t_{w_p} is (G^*, I^*)-rigid and (G, I)-movable, it must happen that $w \in H(G - w_p, x_1)$, which is a contradiction because G is a cactus. Hence, $x_1 \in V(C)$. As before, one can show that there exists a vertex $x_2 \in \big(N_{G^*}(x_1) \setminus \{w_p\}\big) \cap I^*$ which is (G^*, I^*)-rigid and (G, I)-movable, and hence must be in $V(C)$. Repeat the arguments, we finally obtain that there must be some (G^*, I^*)-rigid token placed at some vertex, say u, in $V(C)$ of distance 2 (in G) from w which is different from w_p and x. Now, let y be the common neighbor of w and u. As the token t_u placed at u is (G^*, I^*)-rigid, there exists some vertex $y' \in \big(N_{G^*}(y) \setminus \{u\}\big) \cap I^*$ such that the token $t_{y'}$ placed at y' is $(G^* - N_{G^*}[u], I^* \cap (G^* - N_{G^*}[u]))$-rigid, and hence (G^*, I^*)-rigid as t_u is (G^*, I^*)-rigid. Let $H(G^* - N_{G^*}[u], y')$ be the component of $G^* - N_{G^*}[u]$ containing y'. Since $t_{y'}$ is (G, I)-movable, $H(G^* - N_{G^*}[u], y')$ is not a component of $G - N_G[u]$, which means that $w \in H(G - N_G[u], y')$. But this is a contradiction because G is a cactus.

Case (i-3): $w_{p-1} \neq w$ and $w_{p-2} \neq w$. (See Figure 4.7.) As before, one can assume that any (G^*, I^*)-rigid token is of distance (in G) at least 3 from w. Before analyzing **Case (i-3)**, we prove some useful claims.

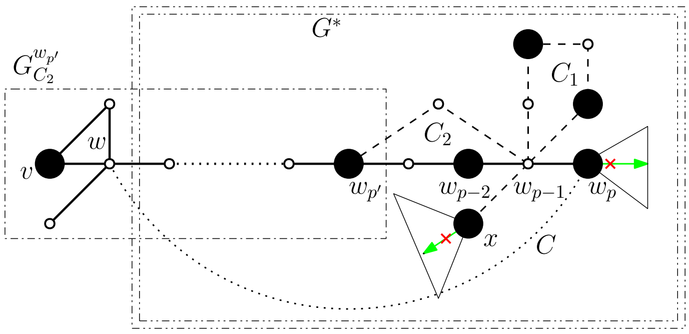


Figure 4.7: Illsutration of **Case (i-3)** of Lemma 4.11(i).

Claim 4.11.3. *There does not exist a cycle C_1 such that $w_{p-1} \in V(C_1)$, $w_p \notin V(C_1)$, $w_{p-2} \notin V(C_1)$, and the path $P_1 = C_1 - w_{p-1}$ is $(G^* - N_{G^*}[w_p], I^* \cap (G^* - N_{G^*}[w_p]))$-confined.*

Proof. Assume C_1 exists. As in **Case (i-2)**, one can show that there must be a (G^*, I^*)-rigid token placed at some vertex of distance 1 or 2 (in G) from w, which then leads to a contradiction. Hence, such a cycle C_1 does not exist. $\square$

Claim 4.11.4. *Assume that Claim 4.11.3 holds. There is a (unique) cycle C_2 such that $\{w_{p-1}, w_{p-2}\} \subseteq V(C_2)$, $w_p \notin V(C_2)$, and the path $P_2 = C_2 - w_{p-1}$ is $(G^* - N_{G^*}[w_p], I^* \cap (G^* - N_{G^*}[w_p]))$-confined.*

Proof. Assume that C_2 does not exist. Since t_{w_p} is (G^*, I^*)-rigid, there must be some vertex $x \in \big(N_{G^*}(w_{p-1}) \setminus \{w_p\}\big) \cap I^*$ such that the token t_x placed at x is $(G^* - N_{G^*}[w_p], I^* \cap (G^* - N_{G^*}[w_p]))$-rigid, and hence also (G^*, I^*)-rigid as t_{w_p} is (G^*, I^*)-rigid. As before, at most one of the two tokens t_{w_p} and t_x is $(G - w_{p-1}, I \cap (G - w_{p-1}))$-rigid. Without loss of generality, assume that t_{w_p} is not $(G - w_{p-1}, I \cap (G - w_{p-1}))$-rigid. Hence, it must happen that $w \in V(H(G - w_{p-1}, w_p))$, where $H(G - w_{p-1}, w_p)$ is the component of $G - w_{p-1}$ containing w_p. Thus, there exists a (unique) cycle C in G containing w and w_p. Using a similar argument as in the previous part, one can show that this will lead to a contradiction. $\square$

We now consider **Case (i-3)**. Let C_2 be the cycle described in Claim 4.11.4. Let p' be the smallest index $(1 \leq p' \leq p-1)$ such that $w_{p'} \in V(C_2) \cap V(P)$. Using Lemma 4.6 and the fact that for $x \in V(C_2) \setminus \{w_{p'}\}$, $G^{*x}_{C_2} = G^x_{C_2}$ (i.e., $w \in G^{w_{p'}}_{C_2}$), we can thus assume that $w_{p'} \in I$ and the token $t_{w_{p'}}$ placed at $w_{p'}$ is $(G^{*w_{p'}}_{C_2}, I^* \cap G^{*w_{p'}}_{C_2})$-rigid and $(G^{w_{p'}}_{C_2}, I \cap G^{w_{p'}}_{C_2})$-movable. Replace G by $G^{w_{p'}}_{C_2}$, the independent set I by $I \cap G^{w_{p'}}_{C_2}$, and w_p by $w_{p'}$ in the previous arguments, one can then either obtain a contradiction (when $\mathsf{dist}_G(w, w_{p'}) \leq 2$) or repeat the arguments one more time (when $\mathsf{dist}_G(w, w_{p'}) \geq 3$). Hence, we can now conclude that $\mathsf{R}(G^*, I^*) = \emptyset$.

Next, we claim that $\mathsf{C}(G^*, I^*) = \emptyset$. Suppose that it is not empty, i.e., there exists a cycle $C^* \in \mathsf{C}(G^*, I^*)$. Note that C^* is also a cycle of G, and $I \cap C^* = I^* \cap C^*$, which means that $I \cap C^*$ is also a maximum independent set of C^*. Without loss of generality, using Lemma 4.6, we can assume that there is some token t_x placed at a vertex $x \in I \cap C^*$ such that t_x is $(G^x_{C^*}, I \cap G^x_{C^*})$-movable but $(G^{*x}_{C^*}, I^* \cap G^{*x}_{C^*})$-rigid. It follows that $w \in V(G^x_{C^*})$. Since any TS-sequence in $G^x_{C^*}$ can indeed be extended to a TS-sequence in G (see the proof of Proposition 2.5), it follows that $\mathsf{R}(G^x_{C^*}, I \cap G^x_{C^*}) = \emptyset$. Additionally, using the previous part, one can show that the removal of vertices in $\mathcal{B}_w$ from $G^x_{C^*}$ does not result any new rigid token in the obtained graph $G^{*x}_{C^*}$, which clearly contradicts the assumption that t_x is $(G^{*x}_{C^*}, I^* \cap G^{*x}_{C^*})$-rigid.

(ii) We first show that $\mathsf{R}(G^*, I^*) = \emptyset$. Note that, from the assumption, it follows that $|I \cap (\mathcal{B}_w - w)| = \sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big)$ and $N_{\mathcal{B}_w}(w) \cap I = \emptyset$. Toward a contradiction, suppose that $\mathsf{R}(G^*, I^*) \neq \emptyset$. Let $w' \in I^*$ be a vertex where a (G^*, I^*)-rigid token is placed. Let $Q = w_1 w_2 \dots w_q$ be a ww'-path with $w_1 = w$ and $w_q = w'$ $(q \geq 1)$.

Case (ii-1): $w_q = w$. First, consider the case $N_{\mathcal{B}_w}(w) \subseteq N_G[I \cap (\mathcal{B}_w - w)]$. Also note that in this case $|I \cap \mathcal{B}_w| = \sum_{B \in \mathcal{B}_w} \big(\lfloor |B|/2 \rfloor - 1\big) + 1$. It follows that the token t_w placed at w cannot be slid (in G) to any vertex in $N_{\mathcal{B}_w}(w)$. Since t_w is (G, I)-movable, there must be some TS-sequence $\mathcal{S} = \langle I_1 = I, I_2, \dots, I_\ell \rangle$ in G that slides t_w to some vertex in $N_{G^*}(w)$. Since w is the unique cut vertex in $\mathcal{B}_w$ and $|I \cap \mathcal{B}_w|$ is maximum, $\mathcal{S}$ does not involve any vertex in $I \cap (\mathcal{B}_w - w)$, i.e., for every $J \in \mathcal{S}$, $(I \cap (\mathcal{B}_w - w)) \subseteq$

J. (Roughly speaking, no token in $\mathcal{B}_w$ can "move out" while t_w "stay" in w.) Hence, $\mathcal{S}' = \langle I_1 \setminus (I \cap (\mathcal{B}_w - w)), I_2 \setminus (I \cap (\mathcal{B}_w - w)), \dots, I_\ell \setminus (I \cap (\mathcal{B}_w - w))\rangle$ is a TS-sequence in G^* that slides t_w to a vertex in $N_{G^*}(w)$, which is clearly a contradiction. Hence, $N_{\mathcal{B}_w}(w) \nsubseteq N_G[I \cap (\mathcal{B}_w - w)]$. It follows that there exists some vertex $x \in N_{\mathcal{B}_w}(w) \cap V(G^*)$. From the definition of G^* and $I \cap N_{\mathcal{B}_w}(w) = \emptyset$, we must have $N_{G^*}(x) \cap I = \{w\}$, i.e., t_w can be directly slid to x in G^*, which is a contradiction.

Case (ii-2): $w_{q-1} = w$. Without loss of generality, we assume that no (G^*, I^*)-rigid token is placed at w. Assume that there exists a cycle C_1 in G^* such that $w_q \notin V(C_1)$, $w_{q-1} \in V(C_1)$, and the path $P_1 = C_1 - w_{q-1}$ is $(G^* - N_{G^*}[w_q], I \cap (G^* - N_{G^*}[w_q]))$-confined. Let $H(G^* - N_{G^*}[w_q], P_1)$ be the component of $G^* - N_{G^*}[w_q]$ containing P_1. Since all vertices in $N_G[I \cap (\mathcal{B}_w - w)]$ are non-cut, $H(G^* - N_{G^*}[w_q], P_1)$ is also a component of $G - N_G[w_q]$, i.e., the token t_{w_q} placed at w_q cannot be slid to w in G. Using a similar argument as in **Case (i-2)**, one can indeed assume that such cycle C_1 does not exist and then derive some contradiction.

Case (ii-3): $w_{q-2} = w$. Similar as in **Case (i-3)**, one can argue that there does not exist any cycle C_1 such that $w_{q-1} \in V(C_1)$, $w_q \notin V(C_1)$, $w_{q-2} \notin V(C_1)$, and the path $P_1 = C_1 - w_{q-1}$ is $(G^* - N_{G^*}[w_q], I \cap (G^* - N_{G^*}[w_q]))$-confined. On the other hand, there must be some C_2 with $\{w_{q-1}, w_{q-2}\} \subseteq V(C_2)$, $w_q \notin V(C_2)$ and the path $P_2 = C_2 - w_{q-1}$ is $(G^* - N_{G^*}[w_q], I \cap (G^* - N_{G^*}[w_q]))$-confined. As in **Case (i-3)**, we assume that $\mathsf{R}(G^w_{C_2}, I \cap G^w_{C_2}) = \emptyset$ and argue with the triple $(G^w_{C_2}, I \cap G^w_{C_2}, w)$ instead of (G, I, w_q) and immediately derive the contradiction because of **Case (ii-1)**.

Case (ii-4): $w_{q-1} \neq w$ and $w_{q-2} \neq w$. One can use a similar argument as in **Case (i-3)** to claim that some contradiction must happen.

Using a similar argument as in part (i), one can also show that $\mathsf{C}(G^*, I^*) = \emptyset$. □

The next lemma ensures the correctness of **Step 3**.

Lemma 4.12. *Let G be a cactus. Let I and J be two given independent sets of G. Assume that $\mathsf{R}(G, I) = \mathsf{R}(G, J) = \emptyset$ and $\mathsf{C}(G, I) = \mathsf{C}(G, J) = \emptyset$. Then $I \overset{G}{\leftrightsquigarrow} J$ if and only if $|I| = |J|$.*

Proof. The only-if direction is trivial. We claim the if direction, i.e., if $|I| = |J|$ then $I \overset{G}{\leftrightsquigarrow} J$. It suffices to show that there is some independent set I^* such that $I \overset{G}{\leftrightsquigarrow} I^*$ and $J \overset{G}{\leftrightsquigarrow} I^*$. The following algorithm constructs such I^*. The same process can be applied for J. Initially, let $I^* = \emptyset$.

- Pick a cut vertex w with $\mathcal{B}_w \neq \emptyset$. It follows from the definition of a cactus that such w always exists.

- If $\sum_{B \in \mathcal{B}_w} \left(\lfloor |B|/2 \rfloor - 1\right) = 0$, pick a safe vertex $v \in V(\mathcal{B}_w)$, slide a token in I to v using Lemma 4.10(i). Let I_1 and J_1 be the resulting independent sets. Let $I' = I_1 \setminus \{v\}$ and $J' = J_1 \setminus \{v\}$. Add v to I^*. Remove all vertices in $\mathcal{B}_w$ and let G' be the resulting graph.

- If $\sum_{B \in \mathcal{B}_w} \left(\lfloor |B|/2 \rfloor - 1\right) \geq 1$, we use Lemma 4.10(ii) and Lemma 4.2 to slide at most $\sum_{B \in \mathcal{B}_w} \left(\lfloor |B|/2 \rfloor - 1\right)$ tokens of I to vertices of $\mathcal{B}_w$ so that for every block $B \in \mathcal{B}_w$, the tokens are exhaustively placed at the vertices in $\{v_i[B] : 2 \leq i \leq |B| - 2, \text{and } i \text{ is even}\}$. The same procedure is applied for J. Let I_1 and J_1 be the resulting independent sets. Let $I' = I_1 \setminus (\mathcal{B}_w - w)$ and $J' = J_1 \setminus (\mathcal{B}_w - w)$. Add the vertices in $\mathcal{B}_w$ where tokens are placed to I^*. Remove all vertices in $N_G[I^* \cap (\mathcal{B}_w - w)]$ and let G' be the resulting graph.

- Repeat the steps above with the new triple (G', I', J'). Note that Proposition 2.4 implies that a TS-sequence in G' can be extended to a TS-sequence in G. On the other hand, Lemma 4.11 guarantees that $\mathsf{R}(G', I') = \mathsf{R}(G', J') = \emptyset$ and $\mathsf{C}(G', I') = \mathsf{C}(G', J') = \emptyset$. The algorithm stops when there are no tokens to move.

□

4.4 Length of Reconfiguration Sequence

In this subsection, we show an upper bound on the length of a TS-sequence (if exists) between any two independent sets of a cactus.

Lemma 4.13. *Let (G, I, J) be a* YES*-instance of* Sliding Token *for cactus graphs. Then, one can reconfigure I to J (and vice versa) using $O(n^2)$ token-slides, where $n = |G|$.*

Proof. By Lemma 4.9, it is sufficient to show Lemma 4.13 for the case $\mathsf{R}(G, I) = \mathsf{R}(G, J) = \emptyset$ and $\mathsf{C}(G, I) = \mathsf{C}(G, J) = \emptyset$. The idea of constructing a TS-sequence $\mathcal{S}$ between I and J comes from Lemma 4.12. More precisely, the outline of this construction is as follows.

- Construct a TS-sequence $\mathcal{S}_1$ from I to I^*, and $\mathcal{S}_2$ from J to I^*, as described in Lemma 4.12.
- The TS-sequence $\mathcal{S}$ can be formed by performing $\mathcal{S}_1$ first, and then perform $\mathcal{S}_2$ in reverse order.

Clearly, $\mathcal{S}$ reconfigures I to J. It suffices to show that $\mathcal{S}_1$ (as well as $\mathcal{S}_2$, and hence $\mathcal{S}$) uses $O(n^2)$ token-slides. We note that in Lemma 4.10, each time a (chosen) token t is moved from the original vertex to some vertex a safe block of G, it performs $O(n)$ steps of token sliding. In case the set M described in Claim 4.10.1 is of size at least 2, the process of "moving away" all tokens in M other than t (and then we can move t) uses $O(n)$ steps, since the number of steps of moving a token $t' \neq t$ in M is bounded by the time of either checking the rigidity of t' itself or checking the confining of a path in a component of a subgraph of G (namely the graph G' described in Claim 4.10.1), which is $O(n)$. In total, t can be moved to some vertex of a safe block using $O(n)$ token-slides. The "arrangement" (if necessary) using Claim 4.10.3 can be done with $O(n^2)$ token-slides in total, because for a safe cycle C, the arrangement takes $O(|V(C)|^2)$ token-slides (Lemma 4.2). Hence, the construction of $\mathcal{S}_1$, and then $\mathcal{S}_2$ and $\mathcal{S}$, uses $O(n^2)$ token-slides. □

Chapter 5

Conclusion and Future Works

In this thesis, we have shown that SLIDING TOKEN can be solved efficiently for trees, and cactus graphs (whose treewidth is at most 2). Since INDEPENDENT SET RECONFIGURATION under TJ/TAR can be solved efficiently for trees [20] and cactus graphs [28], it follows that under TS, TJ, and TAR rules, INDEPENDENT SET RECONFIGURATION for cactus graphs can be solved efficiently. Moreover, for a YES-instance, one can construct in polynomial time a TS-sequence between two given independent sets. The results presented in this thesis partially answered the open question of whether efficient algorithms exist for SLIDING TOKEN (and more generally, for INDEPENDENT SET RECONFIGURATION) when the input graph is of small bandwidth/treewidth/pathwidth/cliquewidth. For investigating this open question, the next interesting targets may be series-parallel graphs [46] (graphs of treewidth at most 2), distance-hereditary graphs [47] (whose cliquewidth is at most 3) and its subclasses, and bandwidth-2 graphs [48, 49]. Especially, from the structure of a bandwidth-2 biconnected graph [48], we conjecture that the algorithm for solving SLIDING TOKEN for cycles (see Section 4.1) can be used as a basis for designing a polynomial-time algorithm for solving SLIDING TOKEN for banwidth-2 biconnected graphs. Consequently, one may use a similar approach as in the case of cactus graphs for solving SLIDING TOKEN for banwidth-2 graphs. In a bandwidth-2 graph G, a key structure that forbids the existence of a TS-sequence between two independent sets of G seems to be its *confined biconnected components*, where a *biconnected component* of G is a maximal subgraph of G that is biconnected. Thus, we conjecture that for bandwidth-2 graphs, SLIDING TOKEN can be solved in polynomial time.

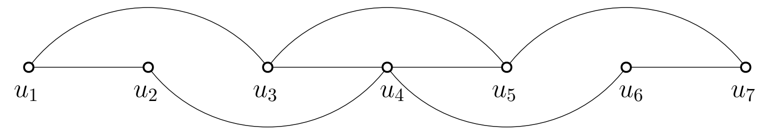


Figure 5.1: An example of a banwidth-2 biconnected graph.

Another problem that is closely related to SLIDING TOKEN is the SHORTEST SLIDING TOKEN problem. Given a YES-instance (G, I, J) of SLIDING TOKEN, SHORTEST SLIDING TOKEN asks whether there exists a TS-sequence of shortest length between I and J. SHORTEST SLIDING TOKEN remains open even for trees. To the best of our knowledge, the first positive result regarding this problem is achieved when the input graph is a caterpillar [35] (a subclass of trees). Given a YES-instance (T, I, J) of SLIDING TOKEN where I, J are independent sets of a tree T, one can define an auxiliary directed graph

$A(T, I, J)$ with vertex set $V(A(T, I, J)) = V(T)$ and there is a directed egde between $x, y \in V(A(T, I, J))$ if $xy \in E(T)$ and $\left|I \cap T^x_y\right| \leq \left|J \cap T^x_y\right|$. Recall that T^x_y is the subtree of T induced by y and its descendants when regarding x as the root. Intuitively, if there is a directed edge between x and y then a TS-sequence between I and J may, at some point, move a token from x to y. We conjecture that the structure of $A(T, I, J)$ can be used to design a polynomial-time algorithm for finding a TS-sequence of smallest length between two independent sets I, J of a tree T.

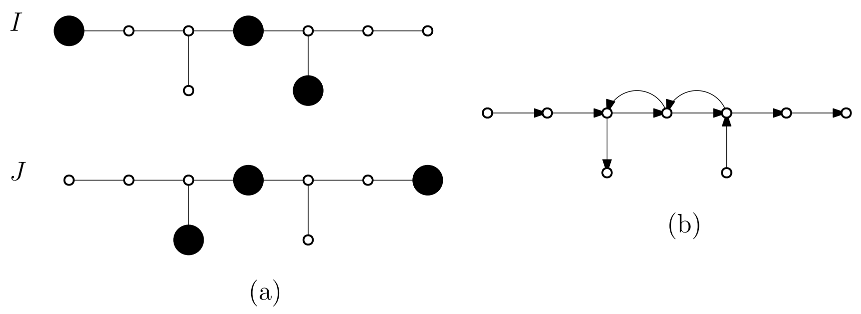


Figure 5.2: (a) A YES-instance (T, I, J) of SLIDING TOKEN for trees and (b) the corresponding auxiliary graph $A(T, I, J)$.

In a more general viewpoint, one may investigate the structural properties of the corresponding reconfiguration graph of SLIDING TOKEN. A fundamental question is given a graph G, which graph is isomorphic to the corresponding reconfiguration graph of SLIDING TOKEN with input G. Fatehi et al. [36] investigated the same question for INDEPENDENT SET RECONFIGURATION under TJ/TAR with different input graphs. To the best of our knowledge, up to present, determining which graph can be a reconfiguration graph of SLIDING TOKEN remains open.

Bibliography

[1] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W.H. Freeman & Company, 1979. `isbn:978-0-716-71044-8`.

[2] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, 4th edition, 2010. `isbn:978-3-642-14278-9`.

[3] Wm. Woolsey Johnson and William E. Story. Notes on the "15" puzzle. *American Journal of Mathematics*, 2(4):397–404, 1879. `doi:10.2307/2369492`.

[4] D. Kornhauser, G. Miller, and P. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science, SFCS 1984*, pages 241–250. IEEE Computer Society, 1984. `doi:10.1109/SFCS.1984.715921`.

[5] Daniel Ratner and Manfred Warmuth. The $(n^2 - 1)$-puzzle and related relocation problems. *Journal of Symbolic Computation*, 10(2):111–137, 1990. `doi:10.1016/S0747-7171(08)80001-6`.

[6] Erik D Demaine and Mikhail Rudoy. A simple proof that the $(n^2 - 1)$-puzzle is hard. *arXiv preprint*, 2017. `arXiv:1707.03146`.

[7] Pavel Surynek. An application of pebble motion on graphs to abstract multi-robot path planning. In *Proceedings of the 21st International Conference on Tools with Artificial Intelligence, ICTAI 2009*, pages 151–158. IEEE, 2009. `doi:10.1109/ICTAI.2009.62`.

[8] Jan van den Heuvel. *The complexity of change*, pages 127–160. London Mathematical Society Lecture Note Series. Cambridge University Press, 2013. `doi:10.1017/CBO9781139506748.005`.

[9] Aaron F. Archer. A modern treatment of the 15 puzzle. *The American mathematical monthly*, 106(9):793–799, 1999. `doi:10.2307/2589612`.

[10] Richard M. Wilson. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, Series B*, 16(1):86–96, 1974. `doi:10.1016/0095-8956(74)90098-7`.

[11] Oded Goldreich. *Finding the Shortest Move-Sequence in the Graph-Generalized* 15-*Puzzle Is NP-Hard*, volume 6650 of *Lecture Notes in Computer Science*, pages 1–5. Springer, 2011. `doi:10.1007/978-3-642-22670-0_1`.

[12] Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4), 2018. doi:10.3390/a11040052.

[13] Takehiro Ito, Erik D. Demaine, Nicholas J.A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12-14):1054–1065, 2011. doi:10.1016/j.tcs.2010.12.005.

[14] Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Finding paths between 3-colorings. *Journal of Graph Theory*, 67(1):69–82, 2011. doi:10.1002/jgt.20514.

[15] Paul Bonsma. The complexity of rerouting shortest paths. *Theoretical Computer Science*, 510:1–12, 2013. doi:10.1016/j.tcs.2013.09.012.

[16] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959. doi:10.1007/BF01386390.

[17] Julie Beier, Janet Fierson, Ruth Haas, Heather M Russell, and Kara Shavo. Classifying coloring graphs. *Discrete Mathematics*, 339(8):2100–2112, 2016. doi:10.1016/j.disc.2016.03.003.

[18] Saeid Alikhani, Davood Fatehi, and Sandi Klavžar. On the structure of dominating graphs. *Graphs and Combinatorics*, 33(4):665–672, 2017. doi:10.1007/s00373-017-1792-5.

[19] Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005. doi:10.1016/j.tcs.2005.05.008.

[20] Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical Computer Science*, 439:9–15, 2012. doi:10.1016/j.tcs.2012.03.004.

[21] Mark de Berg, Bart M.P. Jansen, and Debankur Mukherjee. Independent-set reconfiguration thresholds of hereditary graph classes. In Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen, editors, *Proceedings of the 36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016*, volume 65 of *Leibniz International Proceedings in Informatics*, pages 34:1–34:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.FSTTCS.2016.34.

[22] Paul Bonsma, Marcin Kamiński, and Marcin Wrochna. Reconfiguring independent sets in claw-free graphs. In R. Ravi and Inge Li Gørtz, editors, *Proceedings of the 14th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2014*, volume 8503 of *Lecture Notes in Computer Science*, pages 86–97. Springer, 2014. doi:10.1007/978-3-319-08404-6_8.

[23] Paul Bonsma and Luis Cereceda. Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. In Luděk Kučera and Antonín

Kučera, editors, *Proceedings of the 32nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2007*, pages 738–749. Springer, 2007. doi:10.1007/978-3-540-74456-6_65.

[24] Marcin Wrochna. Reconfiguration in bounded bandwidth and tree-depth. *Journal of Computer and System Sciences*, 93:1–10, 2018. doi:10.1016/j.jcss.2017.11.003.

[25] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1):77–114, 2000. doi:10.1016/S0166-218X(99)00184-5.

[26] Tom C. van der Zanden. Parameterized complexity of graph constraint logic. In Thore Husfeldt and Iyad Kanj, editors, *Proceedings of the 10th International Symposium on Parameterized and Exact Computation, IPEC 2015*, volume 43 of *Leibniz International Proceedings in Informatics*, pages 282–293. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.IPEC.2015.282.

[27] Daniel Lokshtanov and Amer E. Mouawad. The complexity of independent set reconfiguration on bipartite graphs. In Artur Czumaj, editor, *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 185–195. SIAM, 2018. doi:10.1137/1.9781611975031.13.

[28] Amer E Mouawad, Naomi Nishimura, and Venkatesh Raman. Vertex cover reconfiguration and beyond. In Hee-Kap Ahn and Chan-Su Shin, editors, *Proceedings of the 25th International Symposium on Algorithms and Computation, ISAAC 2014*, volume 8889 of *Lecture Notes in Computer Science*, pages 452–463. Springer, 2014. doi:10.1007/978-3-319-13075-0_36.

[29] Takehiro Ito, Hiroyuki Nooka, and Xiao Zhou. Reconfiguration of vertex covers in a graph. *IEICE Transactions on Information and Systems*, E99.D(3):598–606, 2016. doi:10.1587/transinf.2015FCP0010.

[30] Paul Bonsma. Independent set reconfiguration in cographs and their generalizations. *Journal of Graph Theory*, 83(2):164–195, 2016. doi:10.1002/jgt.21992.

[31] Eli Fox-Epstein, Duc A. Hoang, Yota Otachi, and Ryuhei Uehara. Sliding token on bipartite permutation graphs. In K. Elbassioni and K. Makino, editors, *Proceedings of the 26th International Symposium on Algorithms and Computation, ISAAC 2015*, volume 9472 of *Lecture Notes in Computer Science*, pages 237–247. Springer, 2015. doi:10.1007/978-3-662-48971-0_21.

[32] Marthe Bonamy and Nicolas Bousquet. Token sliding on chordal graphs. In H. Bodlaender and G. Woeginger, editors, *Proceedings of the 43rd International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2017*, volume 10520 of *Lecture Notes in Computer Science*, pages 136–149. Springer, 2017. doi:10.1007/978-3-319-68705-6_10.

[33] Marcin Kamiński, Paul Medvedev, and Martin Milanič. Shortest paths between shortest paths and independent sets. In Costas S. Iliopoulos and William F. Smyth, editors, *Proceedings of the 21st International Workshop on Combinatorial Algorithms,*

IWOCA 2010, volume 6460 of *Lecture Notes in Computer Science*, pages 56–67. Springer, 2010. `doi:10.1007/978-3-642-19222-7_7`.

[34] Marthe Bonamy and Nicolas Bousquet. Reconfiguring independent sets in cographs. *arXiv preprint*, 2014. `arXiv:1406.1433`.

[35] Takeshi Yamada and Ryuhei Uehara. Shortest reconfiguration of sliding tokens on a caterpillar. In Mohammad Kaykobad and Rossella Petreschi, editors, *The 10th International Workshop on Algorithms and Computation, WALCOM 2016*, volume 9627 of *Lecture Notes in Computer Science*, pages 236–248. Springer, 2016. `doi:10.1007/978-3-319-30139-6_19`.

[36] Davood Fatehi, Saeid Alikhania, and Abdul Jalil M. Khalaf. The k-independent graph of a graph. *Advances and Applications in Discrete Mathematics*, 18(1):45–56, 2017. `doi:10.17654/DM018010045`.

[37] Rodney G. Downey and Michael Ralph Fellows. *Parameterized complexity*. Monographs in Computer Science. Springer, 2009. `doi:10.1007/978-1-4612-0515-9`.

[38] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, 2015. `doi:10.1007/978-3-319-21275-3`.

[39] Paul Bonsma and Luis Cereceda. Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. *Theoretical Computer Science*, 410(50): 5215–5226, 2009. `doi:10.1016/j.tcs.2009.08.023`.

[40] Takehiro Ito, Hirotaka Ono, and Yota Otachi. Reconfiguration of cliques in a graph. In Rahul Jain, Sanjay Jain, and Frank Stephan, editors, *Proceedings of the 12th Annual Conference on Theory and Applications of Models of Computation, TAMC 2015*, volume 9076 of *Lecture Notes in Computer Science*, pages 212–223. Springer, 2015. `doi:10.1007/978-3-319-17142-5_19`.

[41] Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Polynomial-time algorithm for sliding tokens on trees. In Hee-Kap Ahn and Chan-Su Shin, editors, *Proceedings of the 25th International Symposium on Algorithms and Computation, ISAAC 2014*, volume 8889 of *Lecture Notes in Computer Science*, pages 389–400. Springer, 2014. `doi:10.1007/978-3-319-13075-0_31`.

[42] Duc A. Hoang. The independent set reconfiguration problem on some restricted graphs. Master's thesis, Japan Advanced Institute of Science and Technology, 2015. URL `http://hdl.handle.net/10119/12643`.

[43] Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Linear-time algorithm for sliding tokens on trees. *Theoretical Computer Science*, 600:132–142, 2015. `doi:10.1016/j.tcs.2015.07.037`.

[44] Duc A. Hoang and Ryuhei Uehara. Sliding tokens on a cactus. In Seok-Hee Hong, editor, *Proceedings of the 27th International Symposium on Algorithms and Computation, ISAAC 2016*, volume 64, pages 37:1–37:26. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016. `doi:10.4230/LIPIcs.ISAAC.2016.37`.

[45] Moritz Mühlenthaler. Degree-constrained subgraph reconfiguration is in P. In *Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science, MFCS 2015*, pages 505–516. Springer, 2015. `doi:10.1007/978-3-662-48054-0_42`.

[46] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes: A Survey*. SIAM, 1999. `doi:10.1137/1.9780898719796`.

[47] Hans-Jürgen Bandelt and Henry Martyn Mulder. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41(2):182 – 208, 1986. `doi:10.1016/0095-8956(86)90043-2`.

[48] Fillia Makedon, Dafna Sheinwald, and Yaron Wolfsthal. A simple linear-time algorithm for the recognition of bandwidth-2 biconnected graphs. *Information Processing Letters*, 46(2):103–107, 1993. `doi:10.1016/0020-0190(93)90206-O`.

[49] Alberto Caprara, Federico Malucelli, and Daniele Pretolani. On bandwidth-2 graphs. *Discrete Applied Mathematics*, 117(1):1–13, 2002. `doi:10.1016/S0166-218X(01)00196-2`.

Publications

Journal

[1] Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada: "Linear-Time Algorithm for Sliding Tokens on Trees," *Theoretical Computer Science*, Vol. 600, pp. 132–142 (Jul. 2015).

International Conference

[2] Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada: "Polynomial-Time Algorithm for Sliding Tokens on Trees," Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC 2014), LNCS 8889, pp. 389–400 (Dec. 2014).

[3] Duc A. Hoang, and Ryuhei Uehara: "Sliding Tokens on a Cactus," Proceedings of the 27th International Symposium on Algorithms and Computation (ISAAC 2016), LIPIcs 64, pp. 37:1–37:26 (Dec. 2016).

Others (not included in this thesis)

[4] Duc A. Hoang, Eli Fox-Epstein, and Ryuhei Uehara: "Sliding Token on Block Graphs," Proceedings of the 11th International Conference and Workshops on Algorithms and Computation (WALCOM 2017), LNCS 10167, pp. 460–471 (Mar. 2017).

[5] Eli Fox-Epstein, Duc A. Hoang, Yota Otachi, and Ryuhei Uehara: "Sliding Token on Bipartite Permutation Graphs," Proceedings of the 26th International Symposium on Algorithms and Computation (ISAAC 2015), LNCS 9472, pp. 237–247 (Dec. 2015).