Solution Discovery via Reconfiguration for Problems in P

Mario Grobler
University of Bremen, Germany

Stephanie Maaz
University of Waterloo, Canada

Nicole Megow
University of Bremen, Germany

Amer E. Mouawad
American University of Beirut, Lebanon

Vijayaragunathan Ramamoorthi
University of Bremen, Germany

Daniel Schmand
University of Bremen, Germany

Sebastian Siebertz
University of Bremen, Germany

Abstract

In the recently introduced framework of solution discovery via reconfiguration [Fellows et al., ECAI 2023], we are given an initial configuration of k tokens on a graph and the question is whether we can transform this configuration into a feasible solution (for some problem) via a bounded number b of small modification steps. In this work, we study solution discovery variants of polynomial-time solvable problems, namely Spanning Tree Discovery, Shortest Path Discovery, Matching Discovery, and Vertex/Edge Cut Discovery in the unrestricted token addition/removal model, the token jumping model, and the token sliding model. In the unrestricted token addition/removal model, we show that all four discovery variants remain in P. For the token jumping model we also prove containment in P, except for Vertex/Edge Cut Discovery, for which we prove NP-completeness. Finally, in the token sliding model, almost all considered problems become NP-complete, the exception being Spanning Tree Discovery, which remains polynomial-time solvable. We then study the parameterized complexity of the NP-complete problems and provide a full classification of tractability with respect to the parameters solution size (number of tokens) k and transformation budget (number of steps) b. Along the way, we observe strong connections between the solution discovery variants of our base problems and their (weighted) rainbow variants as well as their red-blue variants with cardinality constraints.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Fixed parameter tractability; Mathematics of computing → Combinatorics

Keywords and phrases solution discovery, reconfiguration, spanning tree, shortest path, matching, cut

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.76

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2311.13478

Funding *Vijayaragunathan Ramamoorthi*: Funded by the "Mind, Media, Machines" high-profile area at the University of Bremen, jointly with *Nicole Megow, Daniel Schmand* and *Sebastian Siebertz.*

51st International Colloquium on Automata, Languages, and Programming (ICALP 2024).
Editors: Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson;
Article No. 76; pp. 76:1–76:20

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

In classical optimization problems, we are given a problem instance and the task is to compute an optimal solution. However, in many applications and real-world scenarios we are already provided with a current solution, albeit non-optimal or infeasible for a given instance. Depending on the application, it might be desirable to find an optimal or feasible solution via a bounded number of small modification steps starting from the current solution.

Very prominent examples for such "systems" are typically settings where humans are involved in the system and big changes to the running system are not easily implementable or even accepted. When optimizing public transport lines, shift plans, or when assigning workers to tasks it is clearly desirable to aim for an optimal solution that is as similar as possible to the current state of the system. Fellows et al. [13] recently introduced the *solution discovery via reconfiguration* framework addressing the computational aspects of such problems. In their model, an optimizer is given a problem instance together with a current (possibly) infeasible solution. The aim is to decide whether a feasible solution to the given problem can be constructed by applying only a bounded number of changes to the current state. We extend this line of work and focus on core polynomial-time solvable problems on graphs, namely Spanning Tree, Shortest Path, Matching, and Vertex/Edge Cut. More precisely, for any of the four aforementioned problems, say Π, we consider instances consisting of a graph G, a budget b, and a starting configuration of k tokens, which is not necessarily a feasible solution for Π (and where tokens either occupy vertices or edges of G). The goal is to decide whether one can transform the starting configuration of tokens into a feasible solution for Π by applying at most b "local changes".

The solution discovery framework is inspired by approaches transforming one solution to another such as *local search*, *reoptimization*, and *combinatorial reconfiguration*. *Local search* is an algorithmic paradigm that is based on iterative improvement of solutions in a previously defined neighborhood. In contrast to our setting, *local search* typically improves the current solution in each step, while we allow arbitrary configurations between the starting and ending configurations (our only restriction is that each vertex/edge can be occupied by at most one token). In *reoptimization* the aim is also to compute optimal solutions starting from optimal solutions of "neighboring" instances (distance between instances being usually defined as the number of vertex/edge addition/deletion required to make the two graphs isomorphic). Closely related is the field of *sensitivity analysis*, a very classical area studying how sensitive an optimal solution is (how it reacts) to small changes in the input. In *combinatorial reconfiguration* we are also given a starting solution, but additionally a *target solution*, and very often constraints on the intermediate steps, e. g., that every intermediate step maintains a valid solution. In our setting the target and intermediate steps are not explicitly specified, but we aim for any final configuration satisfying some desired properties.

As an illustrating example application, consider the scenario in which a city experiences severe weather conditions, leading to a rapid increase in river water levels. The city relies on the protection of a dam, but there is a foreseeable risk that the dam may eventually fail. We assume that the dam breaks (continuously) from point A to point B, where the (shortest) distance between A and B in our graph-view of the world is exactly k vertices (including A and B). In anticipation of such emergencies, the city has also placed (at least) k sandbags in fixed locations across the city so that one can move them as fast as possible to avoid greater damage (we here make the simplifying assumption that each unit of the broken dam can be fixed by a single sandbag). When the dam breaks we can easily compute a shortest path between A and B, which also allows us to compute the minimum number of

sandbags required to stop the flowing water. However, computing such a shortest path is not enough in this situation. Instead, we additionally need to account for sandbags having to move to the appropriate locations as quickly as possible. This corresponds exactly to the problem of finding a shortest path (between two fixed vertices) that is quickly reachable from a predefined set of positions (vertices) in the graph. In other words, this motivates the study of discovery variants of the Shortest Path problem.

An alternative perspective at solution discovery problems is as follows. Consider a vertex (resp. edge) selection problem Π on graphs. Assume that each element (vertex or edge) in the solution of size k must be *supported* by one of k support points, which are located at k different elements of the input graph, and which can each support exactly one element of the solution. The cost of supporting an element is measured by the distance to the chosen support point. The problem of deciding whether there exists a solution that can be supported with cost b corresponds to a discovery variant of problem Π.

Before we proceed and state our results we quickly recall the solution discovery framework of Fellows et al. [13] (see preliminaries for formal definitions). Consider an instance of a vertex (resp. edge) selection problem Π on graphs, where some vertices (resp. edges) of the input graph are occupied by (distinct and indistinguishable) tokens. A token may be moved (in a specific way depending on the concrete model) for a cost of 1. In the *unrestricted token addition/removal* model[1], an existing token may be removed, or a new token may be placed on an unoccupied vertex (resp. edge) for a cost of 1. In the *token jumping* model, a token may be moved from one vertex (resp. edge) to an arbitrary unoccupied vertex (resp. edge) for a cost of 1. In the *token sliding* model, a token may be moved to a neighboring unoccupied vertex (resp. edge) for a cost of 1. The goal is to move the tokens such that they form a valid solution for problem Π within the given budget. We remark that these notions of token moves have also been studied in the realm of combinatorial reconfiguration [21, 24].

Fellows et al. [13] considered the solution discovery variants of (computationally hard) fundamental graph problems, namely Vertex Cover, Independent Set, Dominating Set and Vertex Coloring. The complexity of solution discovery for Vertex Coloring in the color flipping model was studied in [16] under the name k-Fix and in the color swapping model in [10] under the name k-Swap. Since these problems, which we call *base problems*, are NP-complete, it is not surprising that their solution discovery variants are also NP-complete in all of the aforementioned models. In this work, we continue the examination of the solution discovery framework and focus on the discovery variants of polynomial-time solvable base problems, namely Spanning Tree Discovery, Shortest Path Discovery, Matching Discovery, and Vertex/Edge Cut Discovery. When a base problem is polynomial-time solvable, one may, given an instance with a partial or infeasible solution, efficiently compute an optimal solution from scratch. However, as previously illustrated, there are situations in which a solution that is close to a currently established configuration is more desirable. As we show in this work, the constraints put on these problems in the solution discovery framework, namely a limited number of changes, can drastically alter their complexities.

We observe strong connections between the solution discovery variants of our base problems and their *weighted rainbow variants* as well as their *red-blue variants with cardinality constraints*. An instance of a weighted rainbow vertex (resp. edge) selection problem consists

[1] We call the model "unrestricted" to differentiate it from the addition/removal model usually considered in reconfiguration problems as the latter imposes a lower or upper bound on the number of tokens in the graph at all times.

of a weighted vertex (resp. edge) colored graph, and the solution of such an instance may not contain two vertices (resp. edges) of the same color, while "collecting" a certain amount of weight. We show that if (the parameterized version of) the weighted rainbow variant of problem Π admits a fixed-parameter tractable (FPT-) algorithm, then this algorithm can be used to design a fixed-parameter tractable algorithm for the solution discovery variant of Π in the token sliding model. Similarly, solving the solution discovery variant of a problem in the token jumping model boils down to solving the red-blue variant of that same problem. An instance of a red-blue vertex (resp. edge) selection problem consists of a graph where every vertex (resp. edge) is either colored red or blue and two integer parameters k and b. The goal is to find a solution of size k that contains at most b blue vertices (resp. edges).

1.1 Our results

We provide a full classification of tractability vs. intractability with respect to the classical as well as the parameterized complexity of the aforementioned solution discovery problems in all three token models (Table 1). Moreover, we prove some results for rainbow problems as well as red-blue problems, which we believe to be of independent interest. Our main results can be summarized as follows:

- Spanning Tree Discovery, Shortest Path Discovery, Matching Discovery, and Vertex/Edge Cut Discovery are polynomial-time solvable in the unrestricted token addition/removal model.
- Spanning Tree Discovery, Shortest Path Discovery and Matching Discovery are polynomial-time solvable, while Vertex/Edge Cut Discovery is NP-complete in the token jumping model.
- Spanning Tree Discovery is polynomial-time solvable, while Shortest Path Discovery, Matching Discovery, and Vertex/Edge Cut Discovery are NP-complete in the token sliding model.

We then consider the parameterized complexity of the NP-complete discovery problems and establish the following connection with their rainbow variants.

- Meta theorem: For an optimization problem Π, if the weighted Rainbow-Π problem (parameterized by solution size k) admits an FPT algorithm, then Π-Discovery (parameterized by solution size k) admits an FPT algorithm in the token sliding model.

FPT algorithms for the Weighted Rainbow Shortest Path problem [1], Weighted Rainbow Matching problem [19], and the Weighted Rainbow Vertex/Edge Cut problem (which we provide in this paper) immediately imply FPT algorithms for the discovery variants parameterized by k (in the token sliding model). We demonstrate the power of our meta theorem by using it to show that Vertex/Edge Cut Discovery is FPT with respect to parameter k in the sliding model. For Shortest Path Discovery and Matching Discovery, we then give more intuitive and direct FPT algorithms, which also achieve better running times. We conclude by studying the parameterized complexity of all hard problems (not covered by the meta-theorem) when parameterized by either k or b, obtaining the following results.

- Shortest Path Discovery, Matching Discovery, and Vertex/Edge Cut Discovery are FPT when parameterized by solution size k in the token sliding model. Furthermore, Vertex/Edge Cut Discovery is FPT when parameterized by k in the token jumping model.

Table 1 Overview of our results.

	Spanning Tree	Shortest Path	Matching	Vertex/Edge Cut
Discovery Add/Rem.	in P	in P	in P	in P
Discovery Jumping	in P	in P	in P	NP-c., FPT[k], W[1]-hard[b]
Discovery Sliding	in P	NP-complete, FPT[k], FPT[b]	NP-c., FPT[k], W[1]-hard[b]	NP-c., FPT[k], W[1]-hard[b]
Rainbow	in P [6]	NP-complete	NP-complete on paths [25]	NP-complete [3], NP-c. on planar
Red-Blue	in P	in P	in P	NP-c., FPT[k], W[1]-hard[b]

- Shortest Path Discovery is FPT, while Matching Discovery and Vertex/Edge Cut Discovery are W[1]-hard when parameterized by the budget b in the token sliding model. Furthermore, Vertex/Edge Cut Discovery is W[1]-hard when parameterized by the budget b in the token jumping model.

1.2 Related work

The solution discovery framework is closely related to the combinatorial reconfiguration framework, introduced by Ito et al. [21] and studied widely since then. In the reconfiguration variant of a problem we are given an initial solution S and a target solution T and the question is whether S can be transformed into T by a sequence of reconfiguration steps (e. g., token additions/removals, token jumps, or token slides) such that each intermediate configuration also constitutes a solution.

The Minimum Spanning Tree Reconfiguration problem by edge exchanges, i.e., token jumps, was first studied by Ito et al. [21]. They showed that the problem is in P by extending the exchange property of matroids to the reconfiguration of weighted matroid bases. Shortest Path Reconfiguration was introduced by Kamiński et al. [23] and shown to be PSPACE-complete by Bonsma [4]. Reconfiguration of perfect matchings was studied by Ito et al. [22]. The Vertex Cut Reconfiguration and Minimum Vertex Cut Reconfiguration problems were studied by Gomes et al. [18, 17]. For further related work on reconfiguration problems we refer the reader to the surveys of van den Heuvel [20], Nishimura [28], and Bousquet et al. [5].

Rainbow spanning trees have been investigated by Broersma and Li [6]. They characterize graphs in which there exists a rainbow spanning tree via matroid intersection. The question to decide whether a graph contains a rainbow path has been studied in the classical and influential work of Alon et al. [1] that introduced the color coding technique. In the related Rainbow s-t-Connectivity problem the question is to decide whether there exists a rainbow path between s and t, that is, a path on which no color repeats [31]. To the best of our knowledge the Rainbow Shortest Path problem has not been studied in the literature. We refer the reader to [7] for more background. The Rainbow Matching problem is NP-complete, even when restricted to properly edge-colored paths [25]. The Rainbow s-t-Cut problem is known to be NP-complete [3] on general graphs. We show that this problem remains NP-complete even if we restrict it to the class of planar graphs.

We furthermore consider red-blue variants of our base problems. Graphs are now vertex (resp. edge) colored with colors red and blue. We are given two integers k and b and the question is whether there exists a solution of size k using at most b blue vertices (resp. edges). To the best of our knowledge these problem variants have not been studied in the literature, however, variants where we have a cardinality constraint on both colors are related, but seem to be more difficult to solve. For example in the Color Constrained Matching problem [29] we are given a 2-edge-colored graph (colors red and blue) and two parameters k and w and we search for a matching of size k with at most w blue and at most w red edges. This problem is known to be at least as hard as the Exact Matching problem [29] (via a logarithmic-space reduction), which was introduced in [30]. Here, where we are given a red-blue edge-colored graph and a parameter b and the question is whether there exists a perfect matching with exactly b blue edges. The complexity of Exact Matching has been open for more than 40 years [26].

2 Preliminaries

We denote the set of non-negative integers by $\mathbb{N}$ and the set of non-negative reals by $\mathbb{R}_+$. For $k \in \mathbb{N}$ we define $[k] = \{1, 2, \ldots, k\}$ with the convention $[0] = \varnothing$.

Graphs. We consider finite and loopless graphs. An undirected simple graph G consists of its vertex set $V(G)$ and edge set $E(G)$, where $E(G)$ is a subset of all two element sets of $V(G)$. Similarly, the edge set $E(G)$ of a directed simple graph is a subset of pairs of its vertices. In a multigraph we allow $E(G)$ to be a multiset. We assume our graphs to be undirected and simple if not stated otherwise. We denote an edge connecting vertices u and v by uv. Observe that $uv = vu$ for every undirected edge $uv \in E(G)$. A sequence $v_1, \ldots, v_q$ of pairwise distinct vertices is a path of length $q-1$ if $v_i v_{i+1} \in E(G)$ for all $1 \leq i < q$. We write P_q for the path of length q. The distance $\mathrm{dist}_G(u, v)$ (or simply $\mathrm{dist}(u, v)$ if G is clear) between two vertices $u, v \in V(G)$ is the length of a shortest path starting in u and ending in v in G. A graph is d-degenerate if it can be reduced to the empty graph by iterative removal of vertices of degree at most d. For example, forests are 1-degenerate. A graph is bipartite if its vertices can be partitioned into two parts A, B such that no edge has both its endpoints in the same part. Equivalently, a graph is bipartite if it does not contain cycles of odd length. For a vertex subset $S \subseteq V(G)$, we denote by $G[S]$ the subgraph of G induced by S, i.e., the graph with vertex set S and edge set $\{uv \in E(G) \mid u, v \in S\}$. Likewise, for an edge subset $M \subseteq E(G)$, we denote by $G[M]$ the graph with edge set M and vertex set $\{u, v \mid uv \in M\}$.

An edge coloring $\varphi : E(G) \to \mathcal{C}$ is a function mapping each edge $e \in E(G)$ to a color $\varphi(e) \in \mathcal{C}$. Similarly, a vertex coloring assigns colors to vertices. An edge-weight function is a function $w : E(G) \to \mathbb{R}_+$, and similarly a vertex-weight function assigns weights to vertices. We denote colored weighted graphs by tuples (G, w, φ). The weight of a set of vertices/edges is the sum of the weights of its elements.

Solution discovery. Let G be a graph. A *configuration* of G is either a subset of its vertices or a subset of its edges. We formalize the notions of token moves. In the *unrestricted token addition/removal* model[2], a configuration C' can be obtained (in one step) from C, written $C \vdash C'$, if $C' = C \cup \{x\}$ for an element $x \notin C$, or if $C' = C \setminus \{x\}$ for an element

[2] Recall that this definition differs from the definition in [13].

$x \in C$. In the *token jumping* model, a configuration C' can be obtained (in one step) from C if $C' = (C \setminus \{y\}) \cup \{x\}$ for elements $y \in C$ and $x \notin C$. In the *token sliding* model, a configuration C' can be obtained (in one step) from C if $C' = (C \setminus \{y\}) \cup \{x\}$ for elements $y \in C$ and $x \notin C$ if x and y are neighbors in G, that is, if $x, y \in V(G)$, then $xy \in E(G)$; and if $x, y \in E(G)$, then $x \cap y \neq \varnothing$. If C' can be obtained from C (in any model), we write $C \vdash C'$. A *discovery sequence* of length ℓ in G is a sequence of configurations $C_0C_1 \dots C_\ell$ of G such that $C_i \vdash C_{i+1}$ for all $0 \leq i < \ell$.

Let Π be a vertex (resp. edge) selection problem, i.e., a problem defined on graphs such that a solution consists of a subset of vertices (resp. edges) satisfying certain requirements. The Π-Discovery problem is defined as follows. We are given a graph G, a subset $S \subseteq V(G)$ (resp. $S \subseteq E(G)$) of size k (which at this point is not necessarily a solution for Π), and a budget b (as a non-negative integer). The goal is to decide whether there exists a discovery sequence $C_0C_1 \dots C_\ell$ in G for some $\ell \leq b$ such that $S = C_0$ and C_ℓ is a solution for Π.

Note that for discovery problems in the token sliding model we can always assume that $b \leq kn$, where n is the number of vertices in the input graph. This follows from the fact that each token will have to traverse a path of length at most n to reach its target position. For discovery problems in the token jumping model we can always assume $b \leq k$, as it is sufficient to move every token at most once. Similarly, for the unrestricted token addition/removal model we can always assume that $b \leq n$ for vertex selection problems and $b \leq m$ for edge selection problems, where m is the number of edges in the input graph. As k is trivially upper-bounded by n for vertex selection problems (resp. m for edge selection problems), all solution discovery variants we consider are in NP and thus proving NP-hardness suffices to prove NP-completeness.

Parameterized complexity. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed finite alphabet. For an instance $(x, \kappa) \in \Sigma^* \times \mathbb{N}$, κ is called the *parameter*. The problem L is called *fixed-parameter tractable*, FPT for short, if there exists an algorithm that on input (x, κ) decides in time $f(\kappa) \cdot |(x, \kappa)|^c$ whether $(x, \kappa) \in L$, for a computable function f and constant c.

The *W-hierarchy* is a collection of parameterized complexity classes $\mathsf{FPT} \subseteq \mathsf{W[1]} \subseteq \mathsf{W[2]} \subseteq \dots$. It is standard to assume that the inclusion $\mathsf{FPT} \subseteq \mathsf{W[1]}$ is strict. Therefore, showing intractability in the parameterized setting is usually accomplished by establishing an FPT-reduction from a W-hard problem.

Let $L, L' \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems. A *parameterized reduction* from L to L' is an algorithm that, given an instance (x, κ) of L, outputs an instance (x', κ') of L' such that $(x, \kappa) \in L \Leftrightarrow (x', \kappa') \in L'$, $\kappa' \leq g(\kappa)$ for some computable function g, and the running time of the algorithm is bounded by $f(\kappa) \cdot |(x, \kappa)|^c$ for some computable function f and constant c. We refer to the textbooks [9, 12, 14] for extensive background on parameterized complexity.

3 Spanning trees

A *spanning tree* in a connected graph G is a subset of edges $E' \subseteq E(G)$, where $|E'| = |V(G)| - 1$ and $G[E']$ is a tree containing all vertices of G. In the Spanning Tree problem we are given a graph G and the goal is to compute a spanning tree in G.

3.1 Rainbow minimum spanning trees

We reduce the problem of discovering spanning trees in the sliding model to the problem of finding (weighted) rainbow spanning trees. A spanning tree $T \subseteq E(G)$ in a weighted edge-colored multigraph (G, w, φ) is a *rainbow spanning tree* if every edge in T has a distinct color, i.e., $\forall e, e' \in T$ we have $\varphi(e) = \varphi(e')$ if and only if $e = e'$. In the Rainbow Minimum Spanning Tree (Rainbow MST) problem, we are given (G, w, φ) and the goal is to compute a rainbow spanning tree of minimum total weight in G, or report that no such rainbow spanning tree exists. The Rainbow Spanning Tree (Rainbow ST) problem can be defined similarly, i. e., by dropping the weights or assuming that all weights are uniform.

Rainbow spanning trees and their existence have been discussed by Broersma and Li [6]. In our reduction we will construct an instance of Rainbow Minimum Spanning Tree that trivially guarantees the existence of at least one rainbow spanning tree. We show that we can find a Rainbow MST efficiently, even in multigraphs. The proof of the theorem uses arguments similar to those presented in [6] and builds on matroid intersection.

▶ **Theorem 1.** *The* Rainbow Minimum Spanning Tree *problem in weighted edge-colored multigraphs can be solved in polynomial time.*

3.2 Spanning tree discovery

In the Spanning Tree Discovery (STD) problem in the token sliding model, we are given a graph G, an edge subset $S \subseteq E(G)$ with $|S| = |V(G)| - 1$ as a starting configuration, and a non-negative integer b. The goal is to decide whether there is a spanning tree of G that can be discovered (starting from S) using at most b token slides.

▶ **Theorem 2.** *The* Spanning Tree Discovery *problem in the token sliding model can be solved in polynomial time.*

In the full version of the paper we show that the weighted discovery problem, i.e., where we seek a spanning tree of minimum weight, can also be solved in polynomial time.

4 Shortest paths

A *shortest path* between two vertices of a graph G, say s and t, is a path connecting s and t of minimum length. In the Shortest Path problem we are given a graph G and $s, t \in V(G)$ and the goal is to compute a shortest path between s and t in G.

4.1 Rainbow shortest paths

In the Rainbow Shortest Path (Rainbow SP) problem we are given a vertex-colored graph (G, φ) and two vertices $s, t \in V(G)$. A shortest path P from s to t in G is a *rainbow shortest path* if every vertex in P has a distinct color, i. e., $\forall v, v' \in V(P)$, we have $\varphi(v) = \varphi(v')$ if and only if $v = v'$. The Rainbow Shortest Path problem asks for a rainbow shortest path P from s to t in G (one can similarly define the edge-colored variant of the problem).

Rainbow paths have been studied in the literature before, specifically in relation to the rainbow vertex-connection or edge-connection number of a graph. We refer the reader to [7] for more details. In the following theorem, we show that the extra rainbow requirement makes the problem a lot harder than the base problem, even on very restricted families of graphs. To do so, we describe a reduction from the NP-complete [15] Hamiltonian Path problem to the Rainbow Shortest Path problem.

▶ **Theorem 3.** *The* Rainbow Shortest Path *problem is* NP*-complete on the class of* 2*-degenerate bipartite graphs.*

Proof. The fact that the problem is in NP is immediate. We show NP-hardness by a reduction from the Hamiltonian Path problem, which is known to be NP-complete [15]. Given an instance G of Hamiltonian Path, where G is a graph with n vertices denoted by $V(G) = \{v_1, \ldots, v_n\}$, we construct an instance (H, φ, s, t) of Rainbow Shortest Path as follows. See Figure 1 for an illustration.

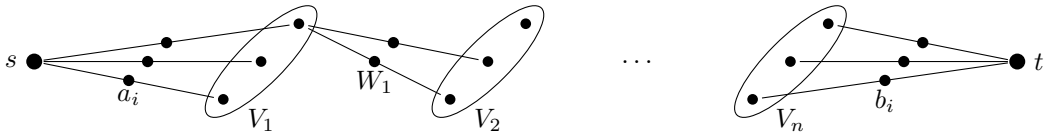

Figure 1 An illustration of the hardness reduction for the Rainbow Shortest Path problem.

First, H consists of two vertices s and t. For every pair $i, j \leq n$ we add a new vertex $v_{i,j}$ in H. We define $V_i = \{v_{i,j} \mid j \leq n\}$. Then, for every $i < n$ and $e \in E(G)$ we add a new vertex $w_{i,e}$ and define $W_i = \{w_{i,e} \mid e \in E(G)\}$. Finally, for every $i \leq n$ we add new vertices a_i and b_i. We define $A = \{a_i \mid i \leq n\}$ and $B = \{b_i \mid i \leq n\}$. We connect the vertices as follows. For every $i \leq n$, we insert the edges $\{s, a_i\}, \{a_i, v_{1,i}\}, \{v_{n,i}, b_i\}$, and $\{b_i, t\}$. Finally, for every $i < n$ and $e = \{v_j, v_\ell\}$ we insert the edges $\{v_{i,j}, w_{i,e}\}$ and $\{w_{i,e}, v_{i+1,\ell}\}$. We assign all vertices in A the same color, all vertices in B the same new color, all vertices $\{v_{i,j} \mid i \leq n\}$ the same new color (this set represents all copies of a vertex $v \in V(G)$), all vertices in W_i the same new color, all vertices in B the same new color, and s and t receive two fresh new colors. This finishes the construction of the instance (H, φ, s, t).

Observe that H is indeed bipartite and 2-degenerate; all vertices of the form $w_{i,j}$ are of degree two. Their removal results in a forest, which is 1-degenerate. Bipartiteness follows from the fact that there are no edges in $G[S]$, $G[T]$, $G[V_i]$, nor $G[W_i]$.

We show that G contains a Hamiltonian path if and only if (H, φ, s, t) is a yes-instance of Rainbow Shortest Path. Observe that every shortest s-t-path in H consists of s, exactly one of the vertices $a_i \in A$, exactly one vertex from every V_i and one from every W_i, exactly one of the vertices $b_i \in B$ and t. Hence, there is no path of length less than $2n + 2$ and every such path of length $2n + 2$ is exactly of this form.

Now assume that G contains a Hamiltonian path $v_{i_1} v_{i_2} \ldots v_{i_n}$. We pick the following vertices in H. We pick a_{i_1} and b_{i_n}, the vertices v_{j,i_j}, and, for every $j < n$, we pick the vertices $w_{j,e}$ where $e = \{v_{i_j}, v_{i_{j+1}}\}$. It is not hard to see that these vertices indeed form a rainbow shortest path from s to t in H.

Conversely, assume that (H, φ, s, t) is a yes-instance of Rainbow Shortest Path. By the above observations, every shortest path must be of the same form (picking vertices from the same sets). As every rainbow shortest path cannot contain two copies of the same vertex from $V(G)$ the claim follows, finishing the proof. ◀

4.2 Shortest path discovery

In the Shortest Path Discovery (SPD) problem in the token sliding model we are given a graph G, two vertices $s, t \in V(G)$, a starting configuration $S \subseteq V(G)$ with $|S| = k$, which is equal to the number of vertices on a shortest path between s and t (including s and t), and a non-negative integer b. The goal is to decide whether we can discover a shortest path between s and t (starting from S) using at most b token slides. We denote an instance of

Shortest Path Discovery by a tuple (G, s, t, S, b). We show that the Shortest Path Discovery problem is NP-complete even when restricted to 2-degenerate bipartite graphs by a minor modification of the reduction from the Hamiltonian Path problem to the Rainbow Shortest Path problem.

▶ **Theorem 4.** *The Shortest Path Discovery problem in the token sliding model is NP-complete on the class of 2-degenerate bipartite graphs.*

In fact, since Hamiltonian Path remains NP-complete on cubic planar graphs, it turns out that the class of all graphs arising in the reduction of Theorem 4 is not only 2-degenerate but also has bounded expansion.

It remains an interesting open problem to determine whether Shortest Path Discovery is polynomial-time solvable on classes of graphs excluding a topological minor, or even on planar graphs. Next, we show that the problem is fixed-parameter tractable with respect to the parameter k as well as the parameter b.

▶ **Theorem 5.** *The Shortest Path Discovery problem in the token sliding model is fixed-parameter tractable with respect to parameter k.*

Proof. Let (G, s, t, S, b) be an instance of SPD. For every $v \in V(G)$ we compute its distance to s and delete v if the distance is larger than k. We enumerate the tokens in S as $s_0, s_1, \ldots, s_{k-1}$ such that token s_i shall slide to a vertex at distance i from s. There are $k!$ such enumerations. Now we orient and assign weights to the edges of G to obtain a weighted directed graph H. If uv is an edge such that $\mathrm{dist}_G(s, u) = i$ and $\mathrm{dist}_G(s, v) = i + 1$, then we orient the edge as (u, v) and assign it weight $\mathrm{dist}_G(s_{i+1}, v)$, that is, the cost of moving token s_{i+1} to vertex v. We delete all edges that did not receive a weight, that is, all edges that connect vertices at the same distance from s.

Since H contains only edges between vertices of distance i to distance $i + 1$ from s, an s-t-path in H corresponds to a shortest s-t-path in G. Furthermore, by definition of the weights, a shortest s-t-path P (with respect to the weights) corresponds exactly to the discovery of a shortest path in G where token s_i slides to the vertex of P which is at distance i from s. We can therefore simply compute a shortest s-t-path in H and verify whether its weight is at most $b - \mathrm{dist}_G(s, s_0)$. Since H is a DAG we can compute such a path in time $\mathcal{O}(n + m)$ and, consequently, the full algorithm runs in time $\mathcal{O}(k!(n + m))$. ◀

▶ **Theorem 6.** *The Shortest Path Discovery problem in the token sliding model is fixed-parameter tractable with respect to parameter b.*

Proof. Let (G, s, t, S, b) be an instance of SPD. Let us call the vertices at distance i from s the vertices at level i. If we can discover a solution with budget b, then it must be the case that all but at most b tokens of S are already in their correct positions. In particular, there can be at most b many levels that do not contain a token from S and at most b many levels containing two or more tokens (and each level can contain at most $b + 1$ many tokens). We call levels not containing exactly one token *very bad* levels. Now, for a level i containing exactly one token on vertex v (not a very bad level), we say that i is a *bad* level whenever one of the following is true:

- $i - 1 \geq 0$, level $i - 1$ contains exactly one token on vertex u, and $uv \notin E(G)$; or
- $i + 1 \leq \mathrm{dist}(s, t) + 1$, level $i + 1$ contains exactly one token on vertex w, and $vw \notin E(G)$.

Note that we can have at most b very bad levels and at most $3b$ bad levels (every 3 bad levels require at least one unit of the budget); otherwise we can reject the instance. Each bad or very bad level contains at most $b+1$ tokens and at most b tokens do not belong to a level between s and t. Hence, the total number of tokens that potentially have to move is so far $5b(b+1) = 5b^2 + 5b$ (located on at most $5b$ levels).

We call a level that is neither bad nor very bad a *good* level. Note that we can group the good levels into at most $5b+1$ groups of consecutive levels. Moreover, the tokens of each group form a path. We denote those paths by $P_1, \ldots, P_q$, for $q \leq 5b+1$. Consider some P_i with at least $2b+1$ many vertices, say $P_i = v_1 \ldots v_\ell$ for $\ell \geq 2b+1$. Let v_x be a vertex of P_i such that $\ell + 1 \leq x \leq \ell - (b+1)$. In other words, there are at least b vertices preceding and at least b vertices succeeding v_x in P_i. In what follows we assume, without loss of generality, that the token on v_x does not move whenever v_x belongs to the final shortest path between s and t. This is a safe assumption because if any other token must pass via v_x to reach its destination we simply swap the roles of both tokens. In other words, getting a token on v_x does not consume any units of the budget. We claim that in a shortest discovery sequence of at most b token slides the token on vertex v_x does not move. This follows from the fact that otherwise it must be the case that either all tokens on v_y, $1 \leq y < x$, or all tokens on v_z, $\ell \geq z > x$, must move. However, as there are at least b tokens to move in either case this contradicts the fact that our discovery sequence slides no more than b tokens. To prove that either every token before or after v_x must move in a shortest discovery sequence we aim towards a contradiction. Assume that the token at level x moves but there are two levels $y < x$ and $z > x$ such that the tokens on level y and z do not move. Since the subpath between levels y and z is already a path connecting v_y and v_z (going through v_x), the movement of the token at level x can be ignored, contradicting our assumption of a shortest discovery sequence. Since at most b tokens can move, and by symmetry, we conclude that at most $2b$ many tokens on the two boundaries of P_i can move. We call the tokens on good levels that are not at distance at most b from some boundary of a P_i *fixed* tokens. Note that the number of tokens that are not fixed is at most $10b^2 + 2b$.

Putting it all together, we conclude that the set of tokens that can potentially move has size at most $15b^2 + 7b$. We can now proceed just as in the proof of Theorem 5 to obtain the required algorithm for parameter b; we can now guess the subset of tokens that move as well as the level each moving token will occupy. ◀

5 Matchings

A *matching* in a graph G is a set of edges $M \subseteq E(G)$ such that each vertex $v \in V(G)$ appears in at most one edge in M. In the Matching problem we are given a graph G and an integer k and the goal is to compute a matching of size k in G.

5.1 Rainbow matchings

We show NP-hardness of the Matching Discovery problem via a reduction from the Rainbow Matching problem. Let (G, φ) be an edge-colored graph. A matching $M \subseteq E(G)$ is said to be a *rainbow matching* if all edges in M have pairwise distinct colors. Formally, the Rainbow Matching problem is defined as follows. Given an edge-colored graph (G, φ) and an integer k, decide whether there exists a rainbow matching of size k in G.

▶ **Theorem 7** ([25]). *The Rainbow Matching problem is NP-complete, even when restricted to properly edge-colored paths.*

5.2 Matching discovery

In the Matching Discovery problem in the token sliding model we are given a graph G, a starting configuration $S \subseteq E(G)$ of size k, and a non-negative integer b. The goal is to decide whether we can discover a k-sized matching M (starting from S) using at most b token slides. Recall that a token on some edge e can only slide (in one step) to some edge e' such that e and e' are adjacent, i.e., share a vertex. We denote an instance of Matching Discovery by a tuple (G, S, b). We first show that the Matching Discovery problem is NP-complete even on restricted graph classes.

▶ **Theorem 8.** *The Matching Discovery problem in the token sliding model is NP-hard on 2-degenerate bipartite graphs.*

Furthermore, we show intractability of the problem when parameterized by the budget. This result is established by a parameterized reduction from the W[1]-hard Multicolored Clique problem and is the technically most demanding contribution of the paper.

▶ **Theorem 9.** *The Matching Discovery problem in the token sliding model is W[1]-hard when parameterized by b, even on the class of 3-degenerate graphs.*

On the positive side, we next show that the Matching Discovery problem is fixed-parameter tractable with respect to parameter $k + b$. Then we show that for any instance of the problem one can bound b by a (quadratic) function of k, which implies fixed-parameter tractability of the problem parameterized by k alone. We start with some relevant definitions.

Let $d(t, w)$ denote the minimum number of slides token t has to slide in order to become incident to vertex w. For an integer $i \in \{1, \ldots, b\}$ and a token $t \in S$, let $Y_t^i = \{w \in V(G) \mid d(t, w) = i - 1\}$ and $Z_t^i = \{w \in V(G) \mid d(t, w) = i\}$. Let G_t^i denote the graph $G[Y_t^i \cup Z_t^i] - E(G[Z_t^i])$.

▶ **Theorem 10.** *The Matching Discovery problem in the token sliding model parameterized by both the number of tokens k and the budget b is fixed-parameter tractable.*

Proof Sketch. Let (G, S, b) be an instance of the Matching Discovery problem, where $|S| = k$. The algorithm proceeds as follows. We first guess a set $S' \subseteq S$ of tokens that will slide form their original positions. If such a guess leaves any overloaded vertices, i.e., vertices incident to more than one token, we ignore the guess and proceed to the next one. Note that this *guessing* procedure requires $\mathcal{O}(2^k)$ time in the worst case. Next, for each $r \in \{0, 1, \ldots, b\}$ and for a fixed S', we guess a partition of r over the tokens in S'. In other words, we try all possible ways of distributing the budget r over the tokens in S'.

Now, for a fixed subset S', a fixed r, and a fixed distribution D of r over the tokens of S', let r_i be the budget allocated to a token $t_i \in S'$ under D. The algorithm computes the sets $Y_{t_i}^{r_i}$ and $Z_{t_i}^{r_i}$ for each $t_i \in S'$. The next step is to produce a set of candidate target edges for each t_i, which we denote by T_i. To do so, we run a maximum matching algorithm in the graph $G_{t_i}^{r_i} - (S \setminus S')$. Roughly speaking, we show that it is enough to retain $\mathcal{O}(k^2)$ edges of the matching in $G_{t_i}^{r_i} - (S \setminus S')$; which also implies a bound on the size of T_i. Once the candidate sets have been constructed, the algorithm exhaustively checks whether it can reach a valid matching in the graph G where each token $t_i \in S'$ moves to some edge in T_i. The algorithm declares a no-instance if no valid matching is found during the exhaustive search; otherwise we have a yes-instance. ◀

Finally, using the notion of augmenting paths, we show that the algorithm of Theorem 10 is also a fixed-parameter tractable algorithm for parameter k alone by proving that k upper bounds the parameter b.

▶ **Lemma 11.** *In any yes-instance (G, S, b) of the* Matching Discovery *problem in the sliding model, b can be upper bounded by $2(k^2 + k)$, where $k = |S|$.*

Combining Theorem 10 with Lemma 11 we get the following result.

▶ **Corollary 12.** *The* Matching Discovery *problem in the token sliding model is fixed-parameter tractable with respect to parameter k.*

6 Vertex/edge cuts

Let G be a graph and s and t vertices of G. A *vertex s-t-cut* is a set of vertices $C \subseteq V(G)$ such that every s-t-path contains a vertex of C. Likewise, an *edge s-t-cut* is a set of edges $C \subseteq E(G)$ such that every s-t-path contains an edge of C. In the Vertex Cut (resp. Edge Cut) problem we are given a graph G, vertices s, t and an integer k and the goal is to compute a vertex s-t-cut (resp. edge s-t-cut) of size k.

6.1 Rainbow vertex/edge cut

Given an edge-colored graph (G, φ) with vertices $s, t \in V(G)$, the Rainbow Vertex Cut problem asks whether there exists a vertex s-t-cut $C \subseteq V(G)$ such that all vertices in C have pairwise different colors. Such a cut is called a rainbow vertex s-t-cut. Likewise, the Rainbow Edge Cut problem asks whether there exists an edge s-t-cut $C \subseteq E(G)$ such that all edges in C have pairwise different colors. The Rainbow Edge Cut problem is known to be NP-complete [3, Theorem 5.5]. We start by showing that the problem remains NP-complete on planar graphs.

▶ **Theorem 13.** *The Rainbow Edge Cut problem is NP-complete on planar graphs.*

Proof. Containment in NP is clear as Rainbow Edge Cut on general graphs is in NP. Hence we focus on the hardness proof. We present a reduction from Rainbow Matching on paths. Let (P, κ) be an instance of Rainbow Matching where P is a path on n vertices denoted by $v_1, \dots v_n$ and the edges are colored with colors from a color set $\mathcal{C}$.

We construct an instance $(G, \psi : E(G) \to \mathcal{C}')$ of Rainbow s-t-Cut as follows. The new color set is $\mathcal{C}' = \mathcal{C} \cup \{\mathsf{black}\} \cup \{c_i \mid i \leq n-2\}$, that is, $\mathcal{C}'$ uses the colors from $\mathcal{C}$ as well as $n-2$ fresh colors and the color black.

Let us describe the construction of G in detail; see Figure 2 for an illustration. In the first step, G consists of κ disjoint copies of P, which we call $P_1, \dots, P_\kappa$. Let the vertices of P_j be called $v_{j,1} \dots v_{j,n}$. Additionally, we add two fresh vertices s and t. For every $j < \kappa$, insert a set L_j of $n-1$ fresh vertices, and call them $u_{j,1}, \dots, u_{j,n-1}$. Hence, $V(G) = \bigcup_{j \leq \kappa} V(P_j) \cup \bigcup_{j < \kappa} V(L_j) \cup \{s, t\}$.

Now we connect s and t with a black edge, which enforces that this black edge must be part of every (rainbow) s-t-cut. Hence, any other black edge may not be part of any rainbow s-t-cut of G. We connect s and t to the vertices in P_j and L_j as follows. For every $j \leq \kappa$ we insert the edges $\{s, v_{j,1}\}$ and $\{v_{j,n}, t\}$, which are colored black. Likewise, for every $j < \kappa$ we insert the edges $\{s, u_{j,1}\}$ and $\{u_{j,n-1}, t\}$, which are also colored black. Finally, for every $j < \kappa$ and $i \leq n-2$, we insert the edge $\{v_{j,i}, u_{j,i}\}$ which is colored black and the edges $\{u_{j,i}, v_{j,i+1}\}$ and $\{u_{j,i}, v_{j+1,i+2}\}$, which are colored c_i. This finishes the construction.

We claim that P has a rainbow matching of size κ if and only if G admits a rainbow s-t-cut. Assume that P has a rainbow matching $M = \{e_1, \dots, e_\kappa\}$ of size κ. Without loss of generality, we assume that the edges in M are ordered with respect to the v_i, i.e., if

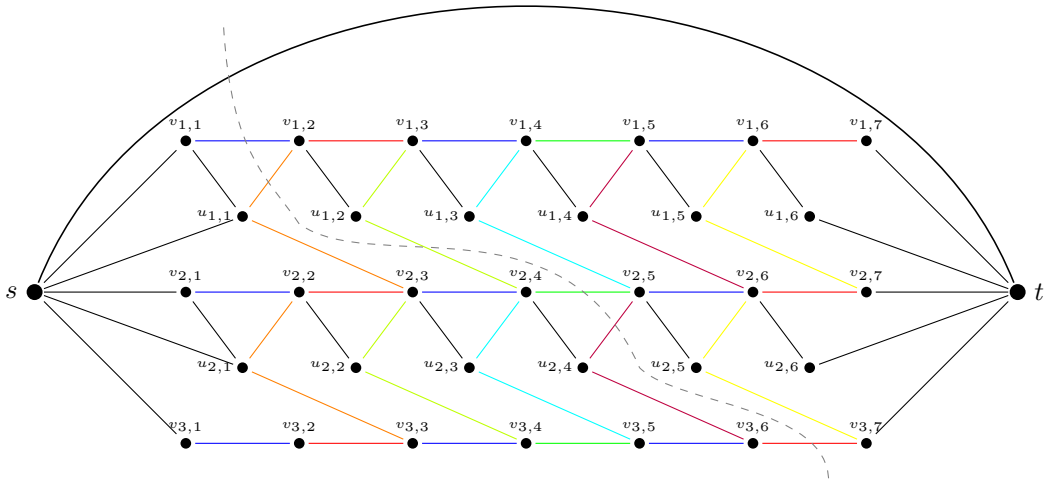

Figure 2 Illustration of the hardness reduction for Rainbow s-t-Cut on planar graphs.

$e_i = \{v_{\ell_i}, v_{\ell_{i+1}}\}$ and $i < j$, then $\ell_i < \ell_j$. We claim that the set C that consists of the black edge $\{s,t\}$, the copy of e_i in P_i, and the obvious edges connecting the P_i and L_j is a rainbow s-t-cut of G. To be precise, we have

$$C = \{s,t\} \cup \{\{v_{i,\ell_i}, v_{i,\ell_i+1}\}, \{u_{i,\ell_i}, v_{i,\ell_i+1}\} \mid i \leq \kappa\} \cup \bigcup_{i \leq \kappa} \{\{u_{i,j}, v_{i,j+2}\} \mid \ell_i < j \leq \ell_{i+1} - 2\}.$$

Observe that C is a cut by construction (see Figure 2), and that no two edges in C have the same color, as M is a rainbow matching, and for every $j < \kappa$ and $i \leq n-2$ at most one of the edges $\{v_{j,i}, u_{j_i}\}$ and $\{u_{j_i}, v_{j+1,i+2}\}$ is contained in C.

Now assume that G admits a rainbow s-t-cut. As s and t are directly connected, every rainbow s-t-cut C for G must contain $\{s,t\}$. Furthermore, by construction C contains exactly one edge from every P_j, say the edge $\{v_{j,\ell_i}, v_{j,\ell_{i+1}}\}$, as no other black edge is part of C. We claim that $M = \{\{v_{\ell_i}, v_{\ell_{i+1}}\} \mid \{v_{j,\ell_i}, v_{j,\ell_{i+1}}\} \in C \text{ for some } j \leq \kappa\}$ is a rainbow matching of P. Obviously, M is rainbow, as C is rainbow. To show that M is indeed a matching, observe that for all $\ell_i \neq \ell_j$ we have $|\ell_i - \ell_j| \geq 2$, that is, M does not contain two (copies of) consecutive edges of P. To see this, assume for the sake of contradiction that there are $i \neq j$ such that $|\ell_i - \ell_j| \leq 1$. Let $j_1 < j_2$ be such that $\{v_{j_1,\ell_i}, v_{j_1,\ell_{i+1}}\}$ and $\{v_{j_2,\ell_j}, v_{j_2,\ell_{j+1}}\}$ are contained in C. By construction, C must also contain $\{u_{j_1,\ell_i}, v_{j_1,\ell_{i+1}}\}$ and can hence not contain $\{u_{j_1,\ell_i}, v_{j_1+1,\ell_i+2}\}$, as they share the same color. This however implies that $\{v_{j_2,\ell_j}, v_{j_2,\ell_{i+1}}\}$ cannot be contained in C, a contradiction. This finishes the proof. ◀

We reuse the ideas of the proof of Theorem 13 combined with standard tricks for edge/vertex cuts, e.g., subdivision of edges, to obtain the following corollary.

▶ **Corollary 14.** *The Rainbow Vertex Cut problem is NP-complete on planar graphs.*

The following theorem is one of two main ingredients required for establishing the fixed-parameter tractability of the discovery variants of cut problems parameterized by k, the other being the aforementioned meta-theorem, which is formally stated in Section 7. In what follows, we consider weighted variants of the rainbow problems, where in addition to finding a rainbow cut of size k it is also required that it has weight at most b.

▶ **Theorem 15.** *The* Weighted Rainbow Vertex/Edge Cut *problem is fixed-parameter tractable when parameterized by k.*

Proof Sketch. We present an FPT algorithm for the Weighted Rainbow Vertex Cut problem. The tractability of Weighted Rainbow Edge Cut follows by considering the vertex cut version in the line graph of the input graph.

We use the *treewidth reduction theorem* [27, Theorem 2.15], which essentially states that for every colored and weighted graph G and integer k we can efficiently compute a graph H of low treewidth that preserves all minimal vertex s-t-cuts of size at most k. Hence, it is sufficient to find a minimum weight rainbow vertex s-t-cut of size at most k in H.

To do so, we apply the optimization version of Courcelle's Theorem as presented in [2, 8]. In our case we conclude a running time of $f(k) \cdot n^2$ where f is some computable function and $n = |V(H)| \leq |V(G)|$. ◀

6.2 Vertex/edge cut discovery

In the Vertex Cut Discovery problem in the token sliding model we are given a graph G, vertices $s, t \in V(G)$, a starting configuration $S \subseteq V(G)$ of size k and a non-negative integer b. The goal is to decide whether we can discover a vertex cut separating s and t in G (starting from S) using at most b token slides. Similarly, in the Edge Cut Discovery problem, we are given a graph G, vertices $s, t \in V(G)$, a starting configuration $S \subseteq E(G)$ of size k and a non-negative integer b. The goal is again to decide whether we can discover an edge cut separating s and t in G (starting from S) using at most b token slides. We denote an instance of Vertex Cut Discovery resp. Edge Cut Discovery by a tuple (G, s, t, S, b). We always use k to denote the size of S.

We prove that Vertex Cut Discovery in the token sliding model is NP-hard, fixed-parameter tractable with respect to parameter k and W[1]-hard with respect to parameter b. We show that these observations translate to Edge Cut Discovery as well. We start by proving hardness via a reduction from the Clique problem, which is NP-hard and its parameterized variant is W[1]-hard with respect to the solution size [11].

▶ **Theorem 16.** *The* Vertex Cut Discovery *problem in the sliding model is NP-hard and W[1]-hard with respect to parameter b on 2-degenerate bipartite graphs.*

Proof Sketch. We show NP-hardness by a reduction from the Clique problem. The reduction is both a polynomial time reduction as well as a parameterized reduction, showing both claimed results. Let (G, κ) be an instance of the Clique problem.

We construct the following graph H (see Figure 3 for an illustration). We add two vertices s and t in H where s has κ pendent vertices, which we collect in a set named Z. For every vertex $u \in V(G)$, we add a vertex x_u in H, which we connect with s, and for every $e \in E(G)$ we add a vertex y_e in H, which we connect with t. Let X denote the set of the x_u and Y denote the set of the y_e. For every vertex $u \in V(G)$ and for each edge $e \in E(G)$ incident with u, connect x_u and y_e via a 1-subdivided edge (that is, a path with one interval vertex). Furthermore, we add $\binom{\kappa}{2}$ disjoint paths $P_1, \ldots, P_{\binom{\kappa}{2}}$, each with a single internal vertex, connecting s and t. We denote the internal vertex of P_i by p_i. This completes the construction of H. We define the initial configuration S as $Z \cup Y$ and $b = 2\kappa + 2\binom{\kappa}{2}$. ◀

▶ **Corollary 17.** *The* Edge Cut Discovery *problem in the token sliding model is NP-hard and W[1]-hard with respect to parameter b on 2-degenerate bipartite graphs.*

Finally, combining Theorem 15 and Theorem 19 (Section 7), we get the following result:

▶ **Theorem 18.** *The* Vertex/Edge Cut Discovery *problem in the token sliding model is fixed-parameter tractable with respect to parameter k.*

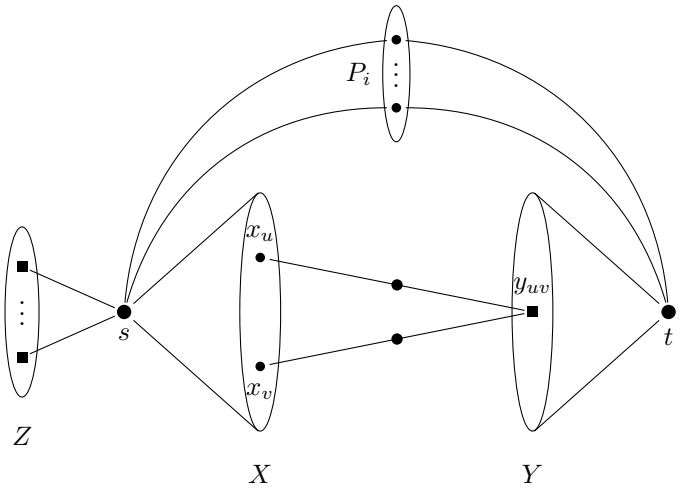

Figure 3 An illustration of the hardness reduction for the Vertex Cut Discovery problem.

7 Tractability via rainbow problems

In this section, we establish an algorithmic meta-theorem showing tractability of discovery problems in the token sliding model (when parameterized by k) via the tractability of (weighted) rainbow variants of optimization problems. We utilize the color-coding technique introduced by Alon et al. [1] along with FPT algorithms for the rainbow problems to design FPT algorithms for the discovery problems parameterized by k.

Let Π be an optimization problem and (G, k) be an instance of Π. In the Rainbow-Π problem we assume that G is either a vertex-colored graph or an edge-colored graph. The parameter k refers to the solution size to be optimized. We consider problems that seek to optimize the selection of edges or vertices, but not both. For instance, the rainbow variants for shortest paths and vertex cuts are vertex selection problems whereas for spanning trees, matchings and edge cuts we have edge selection problems. For a vertex (or edge) selection problem, the Weighted Rainbow-Π problem refers to the weighted variant where the input graph G additionally has weights on the edges (or vertices) and we seek a solution of weight at most b (amongst all solutions satisfying the cardinality constraint k).

▶ **Theorem 19.** *For an optimization problem Π, if the Weighted Rainbow-Π problem parameterized by k admits an FPT algorithm, then the Π-Discovery problem in the token sliding model parameterized by k admits an FPT algorithm.*

Proof Sketch. Without loss of generality, assume that the problem Π is an edge selection problem. Let $(G, S \subseteq E(G), b)$ be an instance of the Π-Discovery problem. Let $\mathcal{C}$ be a palette of k colors, and $\pi : \mathcal{C} \to S$ be a bijection. We color the edges $E(G) \setminus S$ uniformly at random using $\mathcal{C}$, yielding an edge coloring $\varphi : E(G) \to \mathcal{C}$. Now we define a weight function $w : E(G) \to \mathbb{R}_+$ such that for each $e \in E(G)$ we have $w(e) = \text{dist}(e, \pi(\varphi(e)))$. Intuitively, the weight function denotes the cost of moving a token from the initial configuration to an edge with the same color. Observe that the weights of the edges in the initial configuration are zero and they are colored using piece-wise distinct colors. Now we have an edge-colored graph (G, φ). We claim that if (G, φ, k, b) is a yes-instance of the Weighted Rainbow-Π problem, then (G, S, b) is a yes-instance of the Π-Discovery problem in the token sliding model. Intuitively speaking, the color-coding step allows us to transforms the discovery problem into a weighted rainbow problem.

The rest of the proof consists of showing that if (G, S, b) is a yes-instance of the Π-Discovery problem, then the probability that (G, φ, k, b) is a yes-instance of the Weighted Rainbow-Π problem is at least e^{-k}. We then derandomize the algorithm using the technique introduced by Alon et al. [1]. Instead of a random coloring φ, we construct an $(m-k, k)$-perfect hash family $\mathcal{F}$ such that for every $X \subseteq E(G) \setminus S$ with $|X| = k$, there is a function $f \in F$ such that every element of X is mapped to a different element in $\mathcal{C}$ (color palette as a k-sized set). The family $\mathcal{F}$ can be constructed deterministically in time $O(2^{O(k)} \log m)$ [1]. ◀

8 The jumping and addition/removal models

We now turn to the *token jumping* and *unrestricted token addition/removal* models. Recall that in these models we are no longer restricted to sliding tokens along edges.

8.1 Token jumping

We first define the *red-blue* variant of a vertex (resp. edge) selection problem and show that it is always at least as hard as the solution discovery variant in the token jumping model.

Let Π be an arbitrary vertex (resp. edge) selection problem. An instance of the Red-Blue-Π problem consists of a graph G whose vertices (resp. edges) are either colored red or blue, as well as two non-negative integers k and b. If Π is the decision variant of a minimization/maximization problem, the goal is to decide whether there exists a solution $X \subseteq V(G)$ (resp. $X \subseteq E(G)$) of Π of size k such that the number of blue elements in X is at most b. We denote an instance of Red-Blue-Π by (G, k, b).

▶ **Corollary 20.** *Let Π be an arbitrary vertex (resp. edge) selection problem. Then the following results hold in the token jumping model:*

1. *If Red-Blue-Π is in P, then so is Π-Discovery.*
2. *If Red-Blue-Π is in FPT with respect to k or b, then so is Π-Discovery.*
3. *If Π-Discovery is NP-hard, then so is Red-Blue-Π.*
4. *If Π-Discovery is W[1]-hard with respect to k or b, then so is Red-Blue-Π.*

We remark that the other direction does not trivially hold, i. e., we can in general not consider an instance of Red-Blue-Π as an instance of Π-Discovery, as the number of red vertices/edges might exceed the bound k on the solution size. Nevertheless, we show that Red-Blue Spanning Tree, Red-Blue Shortest Path and Red-Blue Matching are in P, implying that their discovery variants in the token jumping model are in P as well, whereas Vertex/Edge Cut Discovery is NP-hard in the token jumping model, implying that Red-Blue Vertex/Edge Cut Discovery is NP-hard as well.

▶ **Proposition 21.** *Red-Blue Spanning Tree, Red-Blue Shortest Path, and Red-Blue Matching can be solved in polynomial time. Hence, Spanning Tree Discovery, Shortest Path Discovery, and Matching Discovery in the token jumping model can be solved in polynomial time.*

▶ **Proposition 22.** *The Vertex/Edge Cut Discovery problem in the token jumping model is NP-complete. Hence, the Red-Blue Vertex/Edge Cut problem is NP-complete.*

▶ **Theorem 23.** *The Weighted Red-Blue Vertex/Edge Cut problem is fixed-parameter tractable when parameterized by k.*

8.2 Unrestricted token addition/removal

Obviously, every spanning tree of a connected graph G has size $|V(G)| - 1$ by definition. Hence, the Spanning Tree Discovery problem in the unrestricted token addition/removal model can easily be reduced to the token jumping model by halving the budget b (rounded down). To see this, note that any solution to the Spanning Tree Discovery problem in the unrestricted token addition/removal model adds and removes the same number of tokens. Furthermore, the order in which we add or remove tokens from the initial configuration S does not matter. Thus, we assume that the first modification step to S is a token removal, followed by a token addition, followed by removals/additions in alternating order. This can easily be simulated by jumps.

Similarly, the length of a shortest s-t-path in a graph G is a fixed number for fixed s and t. Hence, the same argument as above boils the Shortest Path Discovery problem in the unrestricted token addition/removal model down to the Shortest Path Discovery problem in the token jumping model. When it comes to Matching Discovery and Vertex/Edge Cut Discovery, the unrestricted token addition/removal model allows for two natural variations on the problems. On one hand, and in tune with the rest of the paper, if we impose solutions of size exactly $|S|$ then it is not hard to see that the addition/removal model becomes again equivalent to the jumping model.

▶ **Corollary 24.** *The Spanning Tree Discovery, Shortest Path Discovery, Matching Discovery, and Vertex/Edge Cut Discovery problems in the unrestricted token addition/removal model can be solved in polynomial time.*

Alternatively, if we drop the constraints on solution size and allow solutions of any size then the problems become considerably different. In particular, the Matching Discovery problem becomes equivalent to asking for a smallest subset of S whose removal (from S) results in a matching. Similarly, the Vertex/Edge Cut Discovery problem becomes equivalent to the problem of computing a smallest subset of vertices whose addition to S results in a cut between s and t. We show that, even in the relaxed setting, both problems remain polynomial-time solvable.

▶ **Proposition 25.** *The (Relaxed) Matching Discovery problem in the unrestricted token addition/removal model can be solved in polynomial time.*

▶ **Proposition 26.** *The (Relaxed) Vertex/Edge Cut Discovery problem can be solved in the unrestricted token addition/removal model in polynomial time.*

Conclusion and future work

We contribute to the new framework of solution discovery via reconfiguration with complexity theoretic and algorithmic results for solution discovery variants of *polynomial-time solvable* problems. While we can employ the efficient machinery of Weighted Matroid Intersection for the Spanning Tree Discovery problem, all other problems under consideration are shown to be NP-complete, namely, Shortest Path Discovery, Matching Discovery, and Vertex/Edge Cut Discovery. For the latter problems, we provide a full classification of tractability with respect to the parameters solution size k and transformation budget b.

We expect further research on this new model capturing the dynamics of real-world situations as well as constraints on the adaptability of solutions. It seems particularly interesting to investigate directed and/or weighted versions of the studied problems. Notice that all base problems that we studied remain polynomial-time solvable in the presence of

edge weights and for cost functions summing the total weight of the solution (spanning tree, path, matching, cut). Interestingly, our result on the Spanning Tree Discovery problem can be generalized to the weighted setting with a clever adaptation of the reduction. For other problems the hardness results clearly hold, but we leave the existence of FPT algorithms for future work. Another challenge is the design of efficient algorithms that can compute approximate solutions with respect to the solution size/value or with respect to the allowed transformation budget. We note, without proof, that the hardness reduction for the Vertex Cut Discovery problem (Theorem 16) can be adjusted to give a $n^{1-\varepsilon}$-inapproximability of the optimal transformation budget.

References

1 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995.

2 Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.

3 Xuqing Bai, Renying Chang, and Xueliang Li. More on rainbow disconnection in graphs. *Discussiones Mathematicae Graph Theory*, 42:1185–1204, 2020.

4 Paul Bonsma. The complexity of rerouting shortest paths. *Theoretical computer science*, 510:1–12, 2013.

5 Nicolas Bousquet, Amer E. Mouawad, Naomi Nishimura, and Sebastian Siebertz. A survey on the parameterized complexity of the independent set and (connected) dominating set reconfiguration problems. *arXiv preprint*, 2022. arXiv:2204.10526.

6 Hajo Broersma and Xueliang Li. Spanning trees with many or few colors in edge-colored graphs. *Discuss. Math. Graph Theory*, 17(2):259–269, 1997.

7 Lily Chen, Xueliang Li, and Yongtang Shi. The complexity of determining the rainbow vertex-connection of a graph. *Theoretical Computer Science*, 412(35):4531–4535, 2011.

8 Bruno Courcelle and Mohamed Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theoretical Computer Science*, 109(1-2):49–82, 1993.

9 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

10 Marzio De Biasi and Juho Lauri. On the complexity of restoring corrupted colorings. *Journal of Combinatorial Optimization*, 37(4):1150–1169, 2019.

11 Rod G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141(1-2):109–131, 1995.

12 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

13 Michael R. Fellows, Mario Grobler, Nicole Megow, Amer E. Mouawad, Vijayaragunathan Ramamoorthi, Frances A. Rosamond, Daniel Schmand, and Sebastian Siebertz. On solution discovery via reconfiguration. In *ECAI 2023 - 26th European Conference on Artificial Intelligence*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 700–707. IOS Press, 2023.

14 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

15 Michael R. Garey, David S. Johnson, and R Endre Tarjan. The planar hamiltonian circuit problem is NP-complete. *SIAM Journal on Computing*, 5(4):704–714, 1976.

16 Valentin Garnero, Konstanty Junosza-Szaniawski, Mathieu Liedloff, Pedro Montealegre, and Paweł Rzazewski. Fixing improper colorings of graphs. *Theoretical Computer Science*, 711:66–78, 2018.

17 Guilherme Gomes, Sérgio H. Nogueira, and Vinicius F. dos Santos. Some results on vertex separator reconfiguration. *arXiv preprint*, 2020. arXiv:2004.10873.

18 Guilherme C. M. Gomes, Clément Legrand-Duchesne, Reem Mahmoud, Amer E. Mouawad, Yoshio Okamoto, Vinícius Fernandes dos Santos, and Tom C. van der Zanden. Minimum separator reconfiguration. In *18th International Symposium on Parameterized and Exact Computation, IPEC 2023*, volume 285 of *LIPIcs*, pages 9:1–9:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

19 Sushmita Gupta, Sanjukta Roy, Saket Saurabh, and Meirav Zehavi. Parameterized algorithms and kernels for rainbow matching. *Algorithmica*, 81(4):1684–1698, 2019.

20 Jan van den Heuvel. The complexity of change. *Surveys in Combinatorics*, 409(2013):127–160, 2013.

21 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.

22 Takehiro Ito, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, and Yoshio Okamoto. Shortest reconfiguration of perfect matchings via alternating cycles. *SIAM Journal on Discrete Mathematics*, 36(2):1102–1123, 2022.

23 Marcin Kamiński, Paul Medvedev, and Martin Milanič. Shortest paths between shortest paths. *Theoretical Computer Science*, 412(39):5205–5210, 2011.

24 Marcin Kaminski, Paul Medvedev, and Martin Milanic. Complexity of independent set reconfigurability problems. *Theoretical Computer Science*, 439:9–15, 2012.

25 Van Bang Le and Florian Pfender. Complexity results for rainbow matchings. *Theoretical Computer Science*, 524:27–33, 2014. doi:10.1016/J.TCS.2013.12.013.

26 Nicolas El Maalouly and Raphael Steiner. Exact matching in graphs of bounded independence number. In *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022*, volume 241 of *LIPIcs*, pages 46:1–46:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

27 Dániel Marx, Barry O'Sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms (TALG)*, 9(4):1–35, 2013.

28 Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018.

29 Christos Nomikos, Aris Pagourtzis, and Stathis Zachos. Randomized and approximation algorithms for blue-red matching. In *32nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2022*, pages 715–725. Springer, 2007.

30 Christos H. Papadimitriou and Mihalis Yannakakis. The complexity of restricted spanning tree problems. *Journal of the ACM (JACM)*, 29(2):285–309, 1982.

31 Kei Uchizawa, Takanori Aoki, Takehiro Ito, Akira Suzuki, and Xiao Zhou. On the rainbow connectivity of graphs: complexity and fpt algorithms. *Algorithmica*, 67:161–179, 2013.