Separating Feasibility and Movement in Solution Discovery: The Case of Path Discovery

Hanno von Bergen
Universität Hamburg, Germany

Enna Gerhard
University of Bremen, Germany

Stephanie Maaz
University of Waterloo, Canada

Roman Rabinovich
Vodafone GmbH, München, Germany

Daniel Schmand
University of Bremen, Germany

Mai Trinh
Universität Hamburg, Germany

Larissa Fastenau
University of Bremen, Germany

Nicola Lorenz
Universität Hamburg, Germany

Amer E. Mouawad
American University of Beirut, Lebanon

Nicole Schirrmacher
University of Bremen, Germany

Sebastian Siebertz
University of Bremen, Germany

Abstract

We study *solution discovery*, where the goal is to obtain a feasible solution to a problem from an initial configuration by a bounded sequence of local moves. In many applications, however, the graph that defines which vertex sets are feasible is not the same as the graph that governs how tokens, agents, or resources may move. Existing models such as token sliding and token jumping typically do not distinguish the problem graph and the movement graph. Motivated by this mismatch, we introduce a directed weighted *two-graph model* that cleanly separates feasibility from movement. A problem graph specifies the desired combinatorial objects, while a movement graph specifies admissible relocations and their costs. This yields a flexible framework that captures asymmetry, heterogeneous movement constraints, and weighted transitions, while subsuming classical discovery models as special cases.

We investigate this model through Path Discovery and Shortest Path Discovery, where the task is to realize a vertex set containing an s-t-path or a shortest s-t-path in the problem graph. These problems are particularly natural in applications, since directed and weighted shortest paths are among the most fundamental algorithmic primitives. At the same time, previous work has already shown that discovery can be computationally hard even when the underlying optimization problem is easy. Our results show that this phenomenon persists, and becomes especially rich, in the two-graph setting. We obtain a detailed complexity picture, identifying tractable cases as well as strong hardness results.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Routing and network design problems; Mathematics of computing → Graph algorithms

Keywords and phrases solution discovery, shortest path discovery, token sliding, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.MFCS.2026.82

Related Version *Full Version*: `https://arxiv.org/abs/2604.27802` [14]

1 Introduction

Combinatorial reconfiguration studies the structure of the solution space of a combinatorial problem under local transformation rules. Typically, one is given two feasible solutions and asks whether one can transform one into the other by a sequence of small local moves while

maintaining feasibility throughout. Over the past years, combinatorial reconfiguration has developed into a rich area at the interface of algorithms, complexity theory, graph theory, and optimization; we refer to the surveys by van den Heuvel and Nishimura for background and further references [11, 13]. Besides its intrinsic mathematical appeal, reconfiguration is motivated by dynamic and incremental settings in which a system state must be adapted while maintaining feasibility.

Recently, Fellows et al. introduced the framework of *solution discovery via reconfiguration* as a framework inspired by combinatorial reconfiguration [5]. Here, one is not given a target solution in advance. Instead, one starts from an initial configuration and asks whether it can be modified by a bounded sequence of local moves to obtain *some* feasible solution. This viewpoint is natural in settings where a system is already partially deployed, resources are already placed, or a previously valid state has become infeasible after local changes. The framework connects reconfiguration with local search, repair, reoptimization, and dynamic adaptation, and it has already led to a systematic study of discovery variants of classical graph problems [2, 5, 6, 7, 12].

A conceptual limitation of the classical token sliding and token jumping models is that they do not distinguish two structures that are often different in applications: the structure that defines *feasibility* of a target solution, and the structure that governs *movement*. For example, consider the task of setting up a wireless communication system by positioning radio-equipped vehicles such that all areas of interest are covered. The coverage depends on the topology of the area, which determines which sets of positions yield a feasible solution. In contrast, the movement of the vehicles is constrained by the road network, which imposes its own structure, directionality, and costs. Thus, the structure governing feasibility and the structure governing movement are inherently different. This phenomenon is common in applications where resources must be arranged to satisfy structural constraints (such as routes, assignments, or coverage), but can only be moved along a different underlying network with its own constraints. Such examples suggest that feasibility and movement should be modeled separately.

Motivated by this observation, we introduce a directed weighted *two-graph model* for solution discovery. In this model, we are given a directed weighted *problem graph* G, which specifies the target objects whose discovery we are interested in, and a directed weighted *movement graph* M, which specifies admissible token moves and their costs. An instance further contains an initial token configuration and a movement budget. The aim is to move tokens in M so that the finally occupied vertices realize a desired structure in G. Classical models arise as special cases of our framework. The token sliding model is obtained when M coincides with G (or its bi-directed version), so that tokens may only move along edges of the problem graph. The token jumping model is obtained when M is the complete bi-directed graph with unit costs, allowing tokens to move freely between any pair of vertices. Thus, our model strictly generalizes the standard ones while retaining them as natural special cases.

We study this new paradigm through the lens of Path Discovery and Shortest Path Discovery. In Path Discovery, the goal is to realize a vertex set that contains some directed s-t-path in the problem graph G; in Shortest Path Discovery, the goal is to realize a vertex set containing a shortest directed s-t-path in G. This choice is natural for two reasons. First, directed and weighted shortest paths are among the most fundamental problems in classical algorithmics, with countless applications and a long history of efficient algorithms. Second, they provide a clean way to isolate the effect of the discovery process itself: while the underlying shortest-path problem is polynomial-time solvable, its discovery variant is already computationally hard even in the standard undirected token-sliding setting [6]. Thus, Shortest Path Discovery is a particularly well-suited benchmark for understanding how the separation of feasibility and movement affects the complexity landscape.

Shortest Path Reconfiguration in undirected graphs has already been studied in the classical reconfiguration setting, where both the initial and the target shortest path are given and every intermediate state must remain feasible [9], and Shortest Path Discovery in undirected graphs has been studied in [6]. In our study it quickly turns out that Path Discovery is the more fundamental problem in the two-graph model, as Shortest Path Discovery can in most cases be reduced to it by restricting the edge set of the problem graph to those edges that lie on shortest s-t-paths, while keeping the vertex set unchanged. In particular, whenever the considered parameter is not increased by passing to this shortest-path subgraph, our tractability results for Path Discovery also apply to Shortest Path Discovery.

Our results show that the separation of feasibility and movement leads to a robust and expressive framework while preserving substantial algorithmic structure. We obtain a polynomial-time algorithm for the token jumping case, and we show that (Shortest) Path Discovery is fixed-parameter tractable parameterized by the number k of tokens. (Shortest) Path Discovery is fixed-parameter tractable for the structural parameter feedback edge set number. We show that (Shortest) Path Discovery is fixed-parameter tractable when parameterized by any parameter that functionally bounds the number of vertices on all simple directed s-t-paths in G, e.g. the parameter vertex integrity or treedepth of the underlying undirected graph, or for parameter "distance from s to t" for Shortest Path Discovery in case all edge weights are positive. We also obtain XP-algorithms on graph classes of bounded treewidth when treewidth is measured in the undirected union of the problem graph and the movement graph.

On the hardness side, we show that the problem remains difficult already in very restricted settings. In particular, Path Discovery is para-NP-hard for parameter distance from s to t even when all edge weights in the problem graph are positive, while allowing edge weight 0 yields para-NP-hardness for both Path Discovery and Shortest Path Discovery with respect to the parameter distance from s to t. Similarly, if zero-weight edges are allowed in the movement graph, then both problems become para-NP-hard for the budget parameter b. In the classical undirected unweighted setting, we also obtain new hardness results, including NP-hardness on planar graphs and XNLP-hardness for cutwidth (which implies W[i]-hardness for all i for these parameters). Since cutwidth bounds treewidth, the cutwidth hardness also yields hardness for treewidth, and therefore suggests that our XP-algorithm for treewidth cannot be improved to an FPT-algorithm under standard assumptions. Moreover, the same ideas show hardness for the parameter directed feedback vertex set.

Taken together, our results show that separating feasibility and movement is both natural and algorithmically fruitful. The model captures asymmetry, heterogeneity, and weighted movement costs that are unavoidable in realistic applications, while opening a broad new landscape of complexity questions at the interface of shortest paths, solution discovery, and structural parameterized complexity.

2 Preliminaries

For an integer $n \in \mathbb{N}$, we write $[n] \coloneqq \{1, \ldots, n\}$. For a graph G, we denote its vertex and edge sets by $V(G)$ and $E(G) \subseteq V(G) \times V(G)$, respectively. We additionally assume a weight function $w_G \colon E(G) \to \mathbb{Z}$ assigning an integer weight to each edge. For $s, t \in V(G)$, an s-t *path* is a sequence $P = (v_0, v_1, \ldots, v_\ell)$ of pairwise distinct vertices with $v_0 = s$, $v_\ell = t$, and $e_i = (v_{i-1}, v_i) \in E(G)$ for all $i \in [\ell]$. The *weight* of a path P is $w_G(P) \coloneqq \sum_{i=1}^{\ell} w_G(e_i)$, and when G is clear from the context, we simply write $w(e)$ and $w(P)$. The *distance* from u to v

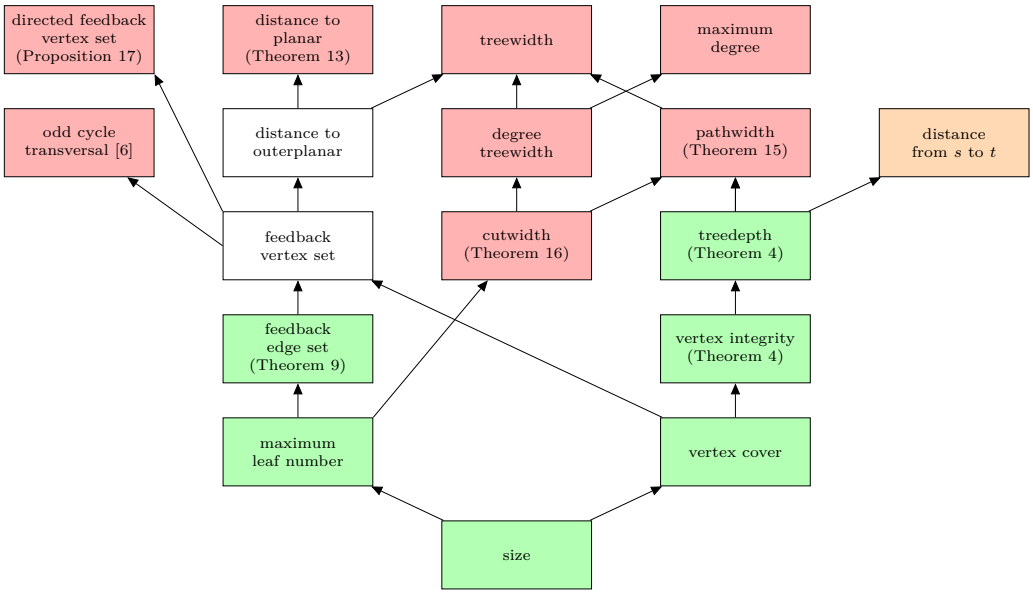

Figure 1 Parameters considered in this paper. Shortest Path Discovery is tractable when parameterized by distance from s to t (in particular by parameter diameter) when all edge weights are positive. The problem is hard when zero weights are allowed. All parameters except for directed feedback vertex set and distance refer to the measure on the underlying undirected movement graph.

in G, denoted $\mathrm{dist}_G(u, v)$, is the minimum weight of a u-v path (if no such path exists, we set $\mathrm{dist}_G(s, t) = \infty$). A *shortest s-t path* is an s-t path P with $w_G(P) = \mathrm{dist}_G(s, t)$. Note that distance is well-defined as we exclude negative cycles and that a shortest walk will always contain a shortest path of the same weight.

Whenever convenient, we view an undirected graph H as a directed graph by replacing each undirected edge $\{x, y\} \in E(H)$ with two directed edges (x, y) and (y, x) of the same weight. Conversely, if G is a graph, we write U for the underlying undirected graph with $V(U) = V(G)$ and $\{x, y\} \in E(U)$ if and only if $(x, y) \in E(G)$ or $(y, x) \in E(G)$.

We work with two directed weighted graphs on a common vertex set: the *problem graph* G, which specifies feasibility/optimality of target objects, and the *movement graph* M, which specifies admissible token moves and their costs. Problem-graph edge weights may be arbitrary integers, but we assume that G contains no negative cycle. Movement costs are non-negative integers, $w_M : E(M) \to \mathbb{Z}_{\geq 0}$.

The restriction on the weights of G is standard, as it ensures e.g. that distances are well-defined. The reason for the restriction on the weights of M is that with negative movement edges, unused tokens could be moved along negative-cost paths and then lifted at the end, invalidating some of our observations.

A *configuration* is a multiset $S \subseteq V(M)$. Vertices in S are called *occupied* by tokens and vertices in $V(M) \setminus S$ are *free*. A *(token) move* in configuration S consists of choosing a token vertex $x \in S$ and an edge $(x, y) \in E(M)$, and updating the configuration to

$$S' \;=\; (S \setminus \{x\}) \cup \{y\} \text{ (as a multiset).}$$

The cost of this move is $w_M(x, y)$. A *discovery sequence* from S to S' is a sequence of configurations $S = S_0, S_1, \ldots, S_r = S'$ such that S_{i+1} is obtained from S_i by a single move for each $i \in \{0, \ldots, r-1\}$. Its total cost is $\sum_{i=0}^{r-1} w_M(e_i)$ where e_i is the movement edge used in the ith move. We say that S' is *reachable from S within budget b* if there exists a discovery sequence from S to S' of total cost at most b.

An instance of (Shortest) Path Discovery consists of a problem graph G, a movement graph M (on the same vertex set), two distinct terminals $s, t \in V(G)$, an initial configuration $S \subseteq V(M)$, and a budget $b \in \mathbb{Z}$. The question is whether there exists a configuration $T \subseteq V(M)$ reachable from S within budget b such that T contains the vertex set of some (shortest) s-t-path in G, that is, there exists a (shortest) s-t-path P in G with $V(P) \subseteq T$.

We emphasize that we do *not* require $T = V(P)$ as it was the case in [6] for the unweighted case. The motivation for this requirement in [6] is that in unweighted graphs, the number of vertices on a shortest path is uniquely determined, and we hence know exactly how many tokens are required for a shortest path. This is not the case in the weighted model. Equivalently, we may allow to lift tokens in the end at zero cost, and we will take this intuitive view, when we would like to say that the final token configuration T highlights the discovered path P. We also remark that the stacking of multiple tokens on single vertices was not allowed in previous work, but we allow this just for convenience. If during a sequence two tokens would temporarily "cross" or meet, one may simply swap their roles: we may simply let the other token move on and let the first token take its place.

We assume familiarity with parameterized complexity, and refer e.g. to [3] for more background. A problem is *fixed-parameter tractable* (FPT) with respect to a parameter p if it can be solved in time $f(p) \cdot n^{O(1)}$ for some computable function f. It is XP with respect to p if it can be solved in time $n^{f(p)}$. We use W[i] for the standard W-hierarchy. We also use the class XNLP and XNLP-hardness in the sense of parameterized logspace computation; in particular, XNLP-hardness implies W[i]-hardness for every i.

3 The tractable cases

In this section, we provide our positive results.

3.1 Parameter number of tokens

In this section, we prove that Path Discovery and Shortest Path Discovery are fixed-parameter tractable when parameterized by the number of tokens k.

We begin with a remark that allows us to focus on Path Discovery (in most cases) when proving positive results.

▶ **Observation 1.** *Let G be a graph and let*

$$d_s(v) \coloneqq \mathrm{dist}_G(s, v), \qquad d_t(v) \coloneqq \mathrm{dist}_G(v, t), \qquad D \coloneqq \mathrm{dist}_G(s, t).$$

If $D = \mathrm{dist}_G(s, t) = \infty$, then the instance is immediately negative. Otherwise, define the subgraph $G^\star$ of G by $V(G^\star) = V(G)$, and

$$E(G^\star) = \{(u, v) \in E(G) \mid d_s(u) < \infty,\ d_t(v) < \infty,\ d_s(u) + w_G(u, v) + d_t(v) = D\}.$$

Then the directed s-t-paths in $G^\star$ are exactly the shortest directed s-t-paths in G. Shortest Path Discovery *on G is equivalent to* Path Discovery *on $G^\star$.*

▶ **Theorem 2.** (Shortest) Path Discovery *is fixed-parameter tractable parameterized by k.*

Proof sketch. By Observation 1, it suffices to consider Path Discovery. Let k be the number of tokens. We try $r = 2, \ldots, k$ in increasing order. For a fixed value of r, we guess the r tokens that will occupy the discovered path, together with their order along the path.

For a fixed ordered list $Y = (y_1, \ldots, y_r)$, we build a layered auxiliary digraph with source vertex α and sink vertex ω, whose i-th layer contains a copy (v, i) of every vertex $v \in V(G)$. An arc from layer i to layer $i+1$ represents choosing a problem-graph edge of G, and its weight is the movement cost of token y_{i+1} to the new vertex. Thus, an α-ω path in this auxiliary graph represents an s-t-walk in G, with total weight equal to the cost of moving the guessed tokens to the chosen vertices. For every guess of tokens, we compute a shortest α-ω path in the auxiliary graph, and we accept as soon as such a path has weight at most b.

It remains to justify that the accepted walk can be taken to be simple. Consider the first value of r for which the algorithm accepts, and let $v_1 = s, \ldots, v_r = t$ be the projected walk. If $v_i = v_j$ for some $i < j$, then deleting the closed subwalk $v_i, \ldots, v_j$, together with the corresponding tokens $y_{i+1}, \ldots, y_j$, gives a shorter s-t-walk. Since movement distances are non-negative, the cost does not increase. Hence, the algorithm would already have accepted for the smaller value $r - (j - i)$, a contradiction. Therefore, the first accepted walk is a simple s-t-path. Conversely, any valid solution is found when we guess its number of vertices and the tokens in their order on the path. Since there are at most

$$\sum_{r=2}^{k} \frac{k!}{(k-r)!} \leq 2^k k!$$

token-order guesses and each auxiliary shortest-path computation is polynomial, the total running time is $f(k) \cdot |V|^{\mathcal{O}(1)}$. ◀

3.2 Tractability of token jumping

By using that in the token jumping model, we have unit costs to move a token anywhere, we do not have to guess the order of tokens as in Theorem 2. Since the target object is a simple s-t-path and every extra repeated cycle in a projected walk can be deleted without increasing the number of newly occupied vertices, we can avoid the color-coding machinery in the token-jumping case.

▶ **Theorem 3.** (Shortest) Path Discovery *in the token jumping model is solvable in polynomial time.*

3.3 Parameter bounded solution size

We next treat the case where the size of the target path is bounded, while the number of available tokens may be large. The difficulty is that we do not know in advance which of the tokens of S will be used on the final path, so we cannot simply restrict to a bounded number of tokens. To resolve this, we apply color-coding simultaneously to the path vertices and to the used tokens, and then invoke the bounded-token algorithm from Theorem 2 on a compressed instance.

▶ **Theorem 4.** (Shortest) Path Discovery *is fixed-parameter tractable when parameterized by any parameter that functionally bounds the number of vertices on all simple directed s-t-paths in G.*

We obtain the following two corollaries.

▶ **Corollary 5.** (Shortest) Path Discovery *is fixed-parameter tractable when parameterized by vertex integrity or treedepth of the underlying undirected graph.*

▶ **Corollary 6.** Shortest Path Discovery *when all edge weights of G are positive is fixed-parameter tractable when parameterized by the distance from s to t in G.*

Proof. Let (G, M, s, t, S, b) be an instance of Shortest Path Discovery, and let $\delta := \mathrm{dist}_G(s, t)$. By Observation 1, it suffices to solve Path Discovery on the shortest-path subgraph $G^\star$ of G. Since all edge weights of G are positive integers, every shortest s-t-path in G has at most $\delta + 1$ vertices. Hence, every directed s-t-path in $G^\star$ has at most $\delta + 1$ vertices. Applying Theorem 4 to $G^\star$ therefore yields an $f(\delta) \cdot |V(G)|^{\mathcal{O}(1)}$-time algorithm. ◀

3.4 Parameter feedback edge set

We now consider the parameter feedback edge set. Here, by feedback edge set we mean a feedback edge set of the underlying undirected graph U of G. We begin with a standard auxiliary lemma: if the target configuration is fixed, then its reachability under a budget can be decided in polynomial time. This subroutine was implicitly used already in tractability proofs for solution discovery; see, the proof of Theorem 3 in [5]. We adapt the lemma for the two-graph setting and make it explicit. The proof follows by solving the described matching problem.

▶ **Lemma 7.** *Let G and M be graphs on the same vertex set V, let $S, T \subseteq V$ (as multisets) be two token configurations with $|T| \leq |S|$, and let $b \in \mathbb{N}$. Assume that lifting unused tokens is allowed. Then one can decide in polynomial time whether the target configuration T can be realized from S within total movement budget at most b.*

More precisely, let H be the complete bipartite graph with bipartition (S, T), and assign to every edge $uv \in S \times T$ the weight $w(u, v) := \mathrm{dist}_M(u, v)$. Then T can be realized from S within budget at most b if and only if H admits a matching that saturates T and has total weight at most b. This problem can be solved in polynomial time.

Next we show that graphs with small feedback edge set contain only few simple s-t-paths.

▶ **Lemma 8.** *Let G be a graph, let $s, t \in V(G)$, and let $F \subseteq E(U)$ be a feedback edge set of the underlying undirected graph U of G of size $f := |F|$. Then the number of simple s-t-paths in G is at most*

$$N(f) \;:=\; \sum_{r=0}^{f} \binom{f}{r} r!\, 2^r \;\leq\; (2f+1)^f.$$

Moreover, every simple s-t-path is uniquely determined by the ordered list of feedback edges it uses together with the directions in which it traverses them.

As a corollary, graphs of small feedback edge set admit fixed-parameter algorithms for both Path Discovery and Shortest Path Discovery, as shown next.

▶ **Theorem 9.** (Shortest) Path Discovery *is fixed-parameter tractable parameterized by the feedback edge set number of the underlying undirected graph of the problem graph G.*

3.5 Parameter treewidth

Finally, we show that Path Discovery and Shortest Path Discovery are XP when parameterized by treewidth. Here, for an instance $(G, M, w_G, w_M, s, t, S, b)$, we measure the treewidth in the graph $U := \big(V(G), E^{\mathrm{und}}(G) \cup E^{\mathrm{und}}(M)\big)$, which is the undirected union of G and M. Our proof is a variation of the proof for Vertex Cover Discovery parameterized by treewidth [5].

▶ **Theorem 10.** (Shortest) Path Discovery *can be solved in time* $n^{\mathcal{O}(tw)}$*, hence, it is in* XP *when parameterized by* tw*.*

Proof Sketch. The algorithm performs a bottom-up dynamic program over a nice tree decomposition of the undirected union of the problem graph and the movement graph. At each bag, the state records, first, the interface of the partial directed s-t-path with the separator, including local path roles and the pairing of open path endpoints, and second, a balance function describing how many tokens must still enter or may leave the processed part through each separator vertex. Since both the path interface and the token balance have only $n^{\mathcal{O}(t)}$ possibilities per bag and all introduce, forget, and join transitions are local, the dynamic program runs in time $n^{\mathcal{O}(t)}$, yielding an XP algorithm. ◀

4 Hardness results

In this section, we collect our hardness results. We first give a direct NP-hardness reduction that simultaneously works for Path Discovery and Shortest Path Discovery even in the classical undirected unweighted model, and then explain how the same construction yields hardness on several restricted graph classes and for several structural parameters. After that, we incorporate the weighted two-graph hardness constructions for the diameter and budget parameters.

4.1 Source problem: Circulating Orientation

We reduce from Circulating Orientation, which is defined as follows.

▶ **Definition 11** (Circulating Orientation)**.** *Let* $G = (V, E)$ *be an undirected multigraph and let* $w\colon E \to \mathbb{N}$ *be an edge-weight function. For an orientation* D *of* G*, let* $A(D)$ *denote the corresponding set of directed arcs. For a vertex* $v \in V$*, define the* weighted indegree *of* v *by*

$$\mathrm{in}_w^D(v) \coloneqq \sum_{(u,v)\in A(D)} w(\{u,v\})$$

and similarly the weighted outdegree *of* v *by*

$$\mathrm{out}_w^D(v) \coloneqq \sum_{(v,u)\in A(D)} w(\{v,u\}).$$

A circulating orientation *of* (G, w) *is an orientation* D *of* G *such that*

$$\mathrm{in}_w^D(v) = \mathrm{out}_w^D(v) = \frac{1}{2}\sum_{e\ni v} w(e) \qquad \text{for every } v \in V.$$

The problem Circulating Orientation *asks whether a given weighted multigraph* (G, w) *admits a circulating orientation.*

The problem is also known as Flow Orientation. It is NP-hard on planar graphs [4]. Moreover, the reduction of Jansen et al. [8] shows W[1]-hardness parameterized by pathwidth even for planar instances. Both hardness results still apply if every edge capacity is bounded by a polynomial in $|V(G)|$.

4.2 NP-hardness on planar graphs even in the undirected one-graph model

We now give an NP-hardness reduction for (Shortest) Path Discovery even in the classical undirected unweighted token-sliding model from Circulating Orientation on planar graphs. Thus, throughout this subsection, the problem graph and the movement graph coincide, the edge relation is symmetric, and all edges have unit cost. For simplicity, we treat the graphs as undirected graphs.

Intuitively, we will create a path that crosses all edges of our original graph twice. We introduce gadgets that force the choice of subpaths in a way that is equivalent to choosing an edge orientation for the original problem. All this is done while preserving planarity.

We need the following auxiliary lemma.

▶ **Lemma 12.** *Let G be a connected plane graph. Then one can compute in polynomial time a Jordan curve γ such that*

1. *for every edge $e \in E(G)$, the curve γ intersects the relative interior of e in exactly two points;*
2. *γ is disjoint from $V(G)$; and*
3. *γ has no other intersection with the drawing of G.*

Consequently, after subdividing every edge of G at the two intersection points with γ, adding edges between consecutive subdivision vertices along γ preserves planarity and makes the new subdivision vertices induce a cycle. Deleting one edge of this cycle yields a path P that crosses every edge exactly twice.

The construction is illustrated in Figure 2.

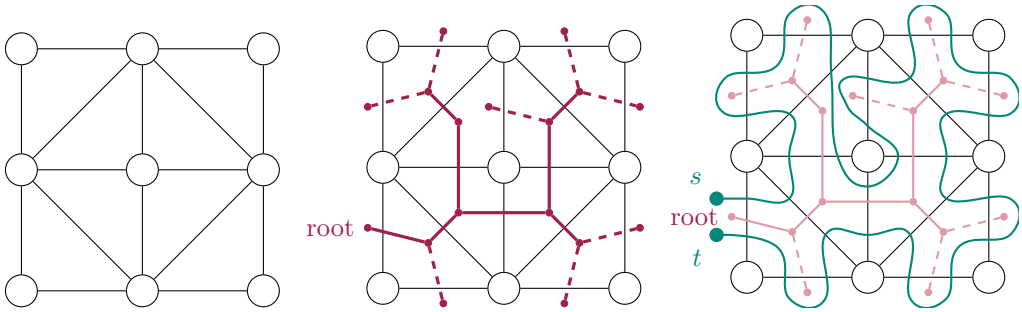

Figure 2 The original graph, the spanning tree of its dual graph (red) with detours (dotted) and constructing a path (green) crossing each edge of a planar graph twice while moving around the spanning tree including its detours of its dual graph on a Jordan curve.

▶ **Theorem 13.** (Shortest) Path Discovery *in the classical undirected unweighted token-sliding model on planar graphs is NP-hard.*

Proof. We start the proof by describing our construction. Let (H, w) be an instance of Circulating Orientation, where H is a planar graph allowing multi-edges with fixed plane embedding and $w\colon E \to \mathbb{N}$ is an edge-weight function. For a vertex $v \in V(H)$, let $W(v) \coloneqq \sum_{e \ni v} w(e)$ denote the total of its incident weights. If $W(v)$ is odd for some vertex v, then the instance is trivially negative. Hence, we may assume throughout that $W(v)$ is even for every vertex v. Set $W \coloneqq \sum_{e \in E(H)} w(e)$ and $L \coloneqq 2|E(H)| + 2W + 1$. We construct an instance (G, s, t, S, b) of (Shortest) Path Discovery as follows.

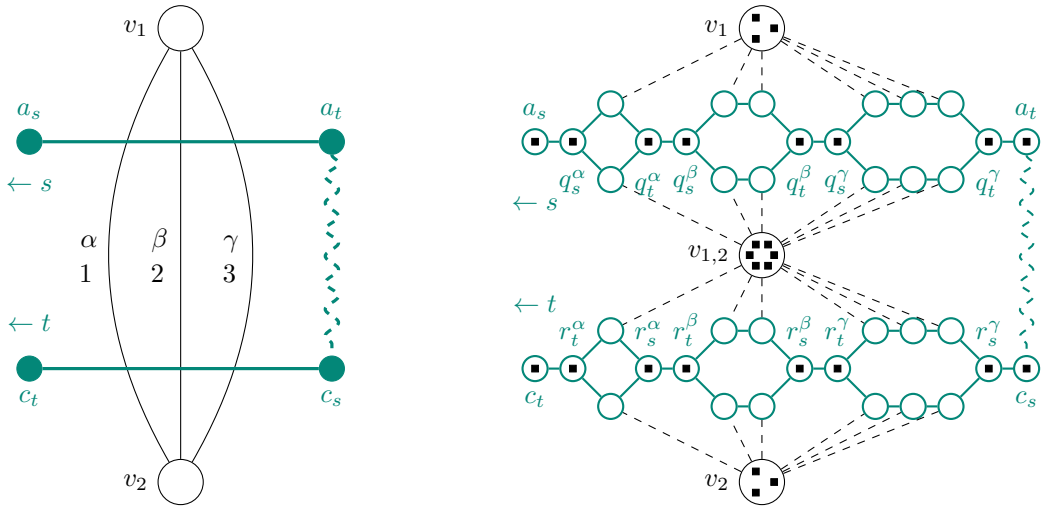

Figure 3 Example of multi-edges α, β, γ of weights 1, 2 and 3 between two vertices in H and a planar gadget that would replace them in G. Only the token capacity for these edges is shown in v_1 and v_2, other adjacent edges would contribute more tokens.

We incrementally construct the planar graph G. It initially contains a copy of $V(H)$. On every vertex $v \in G$, we place $W(v)/2$ tokens to make it a reservoir vertex containing tokens. To avoid token stacking, we could instead add a star of vertices containing a token each and adjust budgets and the further constructions, but this would be more complicated. Now we add the vertices and paths described below. We apply Lemma 12 to find the Jordan curve γ and the path P crossing every edge exactly twice. We add vertices s and t to the graph and place a token on each of s and t. In the following, we will describe on how to use the path P to build and connect planar gadgets for each edge $e \in E(H)$.

For every pair of vertices v_1, v_2 connected by multi-edges $F \subseteq E(H)$, we create a multi-edge gadget. An example for such gadgets is given in Figure 3.

- We create a reservoir vertex $v_{1,2}$ and place tokens equal to the edges capacity $\sum_{e \in F} w(e)$ on it.
- For every $e \in F$, we create two pairs of alternative paths, Q_1^e and Q_2^e, from q_s^e to q_t^e and R_1^e and R_2^e from r_s^e to r_t^e. Each of these alternative paths contains $w(e)$ internal vertices. Choosing one of the two will be equivalent to choosing an edge direction. We place tokens on $q_s^e, q_t^e, r_s^e, r_t^e$.
- We now connect all internal vertices of Q_1^e with v_1, all internal vertices of R_2^e with v_2 and all internal vertices of Q_2^e and R_1^e with $v_{1,2}$ via long subdivided paths of length L. This prevents shortcuts on our intended s-t paths while still being quadratic in size.
- Finally, we connect these gadgets as given by the path P: If P crosses some edge e for the first (second) time just before crossing e' for the first (second) time, we identify q_t^e (r_t^e) with $q_s^{e'}$ ($r_s^{e'}$).

Finally, we now complete our s-t path by connecting s to q_s^e of the first edge e crossed by P and $r_t^{e'}$ to t for the last edge e' touched by P. By our construction, the shortest paths from s to t exactly have length L, and never use any of the reservoir vertices v_1, v_2, and $v_{1,2}$. This construction preserves planarity, see Figure 3.

Let S be the resulting multiset of token positions, and set $b := 2W \cdot L$.

In the following, we show correctness of our construction. For any shortest s-t path of length L, there are exactly $2W$ missing tokens. These exactly have to come from the reservoir vertices. As we have introduced subdivided paths of length L and set our budget accordingly, the tokens can only move from their reservoir vertices to their corresponding internal vertices of the alternative gadget paths. Moving a single token further than that would prevent the path from being constructed as we would exceed the budget elsewhere and all tokens are needed to form the path.

We show that the constructed instance (G, s, t, S, b) is positive if and only if (H, w) admits a circulating orientation.

First, note that every s-t path in G must traverse the Q-gadgets and R-gadgets corresponding to edges in order. Inside each edge gadget, such a path must choose exactly one of the two internally disjoint branches. Initially, all vertices $q_s^e, q_t^e, r_s^e, r_t^e$ for every $e \in E(H)$ as well as s and t carry tokens. Thus, any feasible discovered path requires exactly $2W$ additional occupied vertices. The only tokens not already placed on mandatory path vertices are the reservoir tokens.

Each reservoir token sent to an internal gadget vertex must traverse one of the attachment paths, each of length exactly L. Therefore, moving one such token to its destination costs exactly L. Since a feasible discovered path requires exactly $2W$ such internal vertices to be occupied, every feasible solution of cost at most $b = 2W \cdot L$ must move exactly $2W$ tokens, and each of them must use a shortest attachment path. In particular, no token can afford any detour, and no token can first move through one gadget and then continue elsewhere.

Now fix an edge $e = \{v_1, v_2\} \in E(H)$. If the final s-t path uses the branch Q_1^e, then all $w(e)$ internal vertices of that branch must be occupied. By construction, every such token must come from the reservoir v_1. Hence, choosing Q_1^e consumes exactly $w(e)$ tokens from v_1. Similarly, choosing Q_2^e consumes exactly $w(e)$ tokens from $v_{1,2}$. For the gadgets R^e, the situation is analogous. Consequently, a feasible solution determines for every vertex v_1 a set of edges e for which the branches Q_1^e are used. We orient these edges e to v_1. Finally, the edges for which the shortest path uses branch Q_2^e are directed to v_2. Let k be the number of tokens used from v_1 on the branches Q_1^e. Clearly, we use an additional $\sum_{e \in F} w(e) - k$ tokens from $v_{1,2}$. Consequently, we have to use $\sum_{e \in F} w(e) - k$ tokens from v_2 for the R_2^e branches.

As each vertex v initially contains exactly $W(v)/2$ tokens and every feasible solution must use all reservoir tokens, we obtain

$$\sum_{e \text{ oriented out of } v} w(e) = W(v)/2.$$

Thus, the chosen orientations form a circulating orientation of (H, w).

Conversely, suppose that (H, w) admits a circulating orientation. For every edge $e = \{v_1, v_2\}$, if e is oriented from v_2 to v_1, let the final s-t path use the branch Q_1^e and the branch R_1^e; otherwise let it use the branches Q_2^e and R_2^e. This determines a feasible s-t path through the gadget chain and the budget to move the tokens is exactly used.

Therefore, the constructed instance is positive if and only if (H, w) admits a circulating orientation. ◀

4.3 XNLP-hardness for pathwidth and cutwidth even in the undirected one-graph model

We prove that (Shortest) Path Discovery is XNLP-hard already in the classical undirected unweighted token-sliding model when parameterized by pathwidth, and even when parameterized by cutwidth.

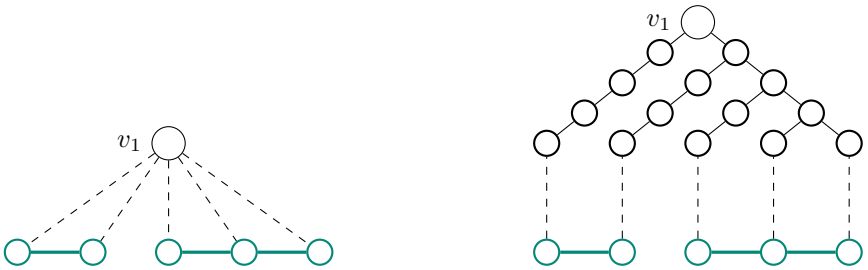

Figure 4 Replacing reservoir vertices with subdivided caterpillars.

We recall the two width parameters that we use. A *path decomposition* of a graph G is a sequence $\mathcal{B} = (B_1, \ldots, B_r)$ of vertex sets, called bags, such that every vertex of G appears in some bag, every edge of G has both endpoints in some bag, and for every vertex $v \in V(G)$, the set of indices i with $v \in B_i$ is an interval. The width of $\mathcal{B}$ is $\max_i |B_i| - 1$, and the *pathwidth* of G, denoted $\mathrm{pw}(G)$, is the minimum width of a path decomposition of G.

A *linear layout* of a graph G is an ordering $\pi = (v_1, \ldots, v_n)$ of $V(G)$. The width of π is

$$\max_{i \in [n-1]} \left| \left\{ \{u, v\} \in E(G) : \pi^{-1}(u) \leq i < \pi^{-1}(v) \text{ or } \pi^{-1}(v) \leq i < \pi^{-1}(u) \right\} \right|.$$

The *cutwidth* of G, denoted $\mathrm{ctw}(G)$, is the minimum width of a linear layout of G.

We use the standard relation between pathwidth and cutwidth. Let $\Delta(G)$ denote the maximum degree of G. Since pathwidth equals vertex separation [10], one obtains

$$\mathrm{pw}(G) \leq \mathrm{ctw}(G) \qquad \text{and} \qquad \mathrm{ctw}(G) \leq \Delta(G) \cdot (\mathrm{pw}(G) + 1).$$

The reduction is from Circulating Orientation parameterized by pathwidth, which is XNLP-complete even when every edge capacity is bounded by a polynomial in $|V(G)|$ [1], and similar to the reduction for planar graphs.

▶ **Theorem 15.** Shortest Path Discovery *is* XNLP*-hard in the classical undirected unweighted token-sliding model when parameterized by the pathwidth of the input graph.*

We now refine the construction to obtain bounded cutwidth. The only obstruction to bounded cutwidth in the construction above is the possible large degree of the reservoir vertices p_v. We replace each such high-degree reservoir by a subdivided caterpillar as in Figure 4 while preserving the crucial property that every useful move from the reservoir to an associated branch vertex has the same length exactly L.

▶ **Theorem 16.** (Shortest) Path Discovery *is* XNLP*-hard in the classical undirected unweighted token-sliding model when parameterized by the cutwidth of the input graph.*

4.4 Further hardness results in the weighted two-graph model

We now record further hardness consequences of the same construction with slight variations. Throughout this subsection, the problem graph G and the movement graph M may differ. We use the construction from the proof of Theorem 15. The reductions below only modify the weights, structure and/or the movement graph. The correctness argument is always the same.

The same reductions can also be made acyclic in the directed two-graph model.

▶ **Proposition 17.** Path Discovery *in the directed two-graph model is para-NP-hard when parameterized by the directed feedback vertex set of the union of problem graph G and movement graph M.*

▶ **Proposition 18.** Path Discovery *in the two-graph model is para-NP-hard when parameterized by the weighted diameter of the problem graph G, even if all edge weights of G are positive integers. In fact, the hardness already holds for instances in which the weighted diameter of G is bounded by an absolute constant.*

▶ **Proposition 19.** *If zero weights are allowed in the problem graph G, then* (Shortest) Path Discovery *is para-NP-hard when parameterized by the weighted diameter of G. In fact, this already holds for instances with weighted diameter* 0.

▶ **Theorem 20.** *If zero weights are allowed in the movement graph M, then* (Shortest) Path Discovery *is para-NP-hard when parameterized by the budget b. In fact, this already holds for $b = 0$.*

5 Conclusion

We introduced a directed weighted two-graph model for solution discovery, in which the graph defining feasibility is separated from the graph defining movement. This separation captures situations in which the combinatorial structure of the desired solution and the constraints governing how tokens, agents, or resources may move are inherently different. Using Path Discovery and Shortest Path Discovery as test cases, we obtained a varied complexity landscape: the model admits efficient algorithms in several natural settings, including for bounded numbers of tokens, token jumping, bounded solution size, and small feedback edge set, while it remains hard under several structural restrictions and for important parameters.

These results indicate that the two-graph perspective is not merely a technical generalization of the classical token-sliding and token-jumping models, but a useful framework for understanding how feasibility and movement interact. A natural direction for future work is to study further discovery problems in this model, such as variants of vertex cover, domination, matching, cut, or clustering problems, and to determine which features of the feasibility problem and of the movement graph govern tractability. It would also be interesting to investigate the same separation of feasibility and movement in the classical reconfiguration setting, where both the initial and target solutions are prescribed, and all intermediate configurations are required to remain feasible. We expect that this two-graph viewpoint will lead to new algorithmic questions and to a more refined understanding of reconfiguration and discovery under realistic movement constraints.

References

1 Hans L. Bodlaender and Krisztina Szilágyi. XALP-completeness of parameterized problems on planar graphs. *Discrete Applied Mathematics*, 2026. doi:10.1016/j.dam.2026.01.021.

2 Nicolas Bousquet, Amer E Mouawad, Stephanie Maaz, Naomi Nishimura, and Sebastian Siebertz. On algorithmic meta-theorems for solution discovery: Tractability and barriers, 2025. doi:10.48550/arXiv.2510.17344.

3 Marek Cygan, Fedor V Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, 1 edition, 2015. doi:10.1007/978-3-319-21275-3.

4 Walter Didimo, Giuseppe Liotta, and Maurizio Patrignani. HV-planarity: Algorithms and complexity. *Journal of Computer and System Sciences*, 99:72–90, 2019. doi:10.1016/j.jcss.2018.08.003.

5 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. *Journal of Computer and System Sciences*, 157:103747, 2026. doi:10.1016/j.jcss.2025.103747.

6 Mario Grobler, Stephanie Maaz, Nicole Megow, Amer E. Mouawad, Vijayaragunathan Ramamoorthi, Daniel Schmand, and Sebastian Siebertz. Solution discovery via reconfiguration for problems in P. In *Proceedings of ICALP 2024*, LIPIcs, 2024. doi:10.4230/LIPIcs.ICALP.2024.76.

7 Mario Grobler, Stephanie Maaz, Amer E. Mouawad, Naomi Nishimura, Vijayaragunathan Ramamoorthi, and Sebastian Siebertz. Kernelization complexity of solution discovery problems. In *35th International Symposium on Algorithms and Computation, ISAAC 2024*, LIPIcs, pages 36:1–36:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ISAAC.2024.36.

8 Bart M. P. Jansen, Liana Khazaliya, Philipp Kindermann, Giuseppe Liotta, Fabrizio Montecchiani, and Kirill Simonov. Upward and orthogonal planarity are W[1]-Hard parameterized by treewidth. In *Graph Drawing and Network Visualization*, pages 203–217, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-49275-4_14.

9 Marcin Kamiński, Paul Medvedev, and Martin Milanič. Shortest paths between shortest paths and independent sets. In *Combinatorial Algorithms*, pages 56–67, Berlin, Heidelberg, 2011. Springer. doi:10.1007/978-3-642-19222-7_7.

10 Nancy G. Kinnersley. The vertex separation number of a graph equals its path-width. *Information Processing Letters*, 42(6):345–350, 1992. doi:10.1016/0020-0190(92)90234-M.

11 Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018. doi:10.3390/a11040052.

12 Rin Saito, Anouk Sommer, Tatsuhiro Suga, Takahiro Suzuki, and Yuma Tamura. Solution discovery for vertex cover, independent set, dominating set, and feedback vertex set. In *SOFSEM 2026: Theory and Practice of Computer Science*, pages 432–446, Cham, 2026. Springer Nature Switzerland. doi:10.1007/978-3-032-17801-5_32.

13 Jan van den Heuvel. The complexity of change. In *Surveys in Combinatorics 2013*, volume 409 of *London Mathematical Society Lecture Note Series*, pages 127–160. Cambridge University Press, 2013. doi:10.1017/CBO9781139506748.005.

14 Hanno von Bergen, Larissa Fastenau, Enna Gerhard, Nicola Lorenz, Stephanie Maaz, Amer E. Mouawad, Roman Rabinovich, Nicole Schirrmacher, Daniel Schmand, Sebastian Siebertz, and Mai Trinh. Separating feasibility and movement in solution discovery: The case of path discovery, 2026. doi:10.48550/arXiv.2604.27802.