Shortest Reconfiguration of Matchings

Nicolas Bousquet[1], Tatsuhiko Hatanaka[2], Takehiro Ito[2], and Moritz Mühlenthaler[2](✉)

[1] CNRS, Laboratoire G-SCOP, Grenoble-INP, Univ. Grenoble-Alpes, Grenoble, France
nicolas.bousquet@grenoble-inp.fr

[2] Fakultät für Mathematik, TU Dortmund University, Dortmund, Germany
moritz.muehlenthaler@math.tu-dortmund.de

Abstract. Imagine that unlabelled tokens are placed on edges forming a matching of a graph. A token can be moved to another edge provided that the edges containing tokens remain a matching. The *distance* between two configurations of tokens is the minimum number of moves required to transform one into the other. We study the problem of computing the distance between two given configurations. We prove that if source and target configurations are maximal matchings, then the problem admits no polynomial-time sublogarithmic-factor approximation algorithm unless $\mathsf{P} = \mathsf{NP}$. On the positive side, we show that for matchings of bipartite graphs the problem is fixed-parameter tractable parameterized by the size d of the symmetric difference of the two given configurations. Furthermore, we obtain a d^{ε}-factor approximation algorithm for the distance of two maximum matchings of bipartite graphs for every $\varepsilon > 0$. The proofs of our positive results are constructive and can hence be turned into algorithms that output shortest transformations. Both algorithmic results rely on a close connection to the Directed Steiner Tree problem. Finally, we show that determining the exact distance between two configurations is complete for the class D^{P}, and determining the maximum distance between any two configurations of a given graph is D^{P}-hard.

Keywords: Matchings · Reconfiguration · Fixed-parameter tractability · Approximation hardness

1 Introduction

A reconfiguration problem asks for the existence of a step-by-step transformation between two given configurations, where in each step we apply some simple modification to the current configuration. The set of configurations may for instance be the set of k-colorings [3,10] or independent sets [15,16,18] of a graph, or the

N. Bousquet—This author was partially supported by ANR project GrR (ANR-18-CE40-0032).

T. Ito—Partially supported by JST CREST Grant Number JPMJCR1402, and JSPS KAKENHI Grant Numbers JP18H04091 and JP19K11814, Japan.

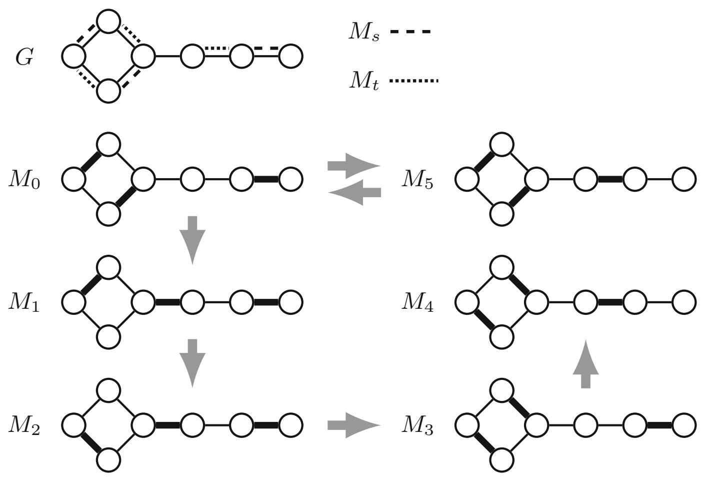

Fig. 1. A reconfiguration sequence $M_s = M_0, M_1, \ldots, M_4 = M_t$ of matchings in a graph G.

set of satisfying assignments of a Boolean formula [12]. A suitable modification may for example alter the color of a single vertex, or the truth value of a variable in a satisfying assignment. For recent surveys on reconfiguration problems the reader is referred to [14] or [20].

Recently, there has been considerable interest in the complexity of finding shortest transformations between configurations. Examples include finding a shortest transformation between triangulations of planar point sets [22] and simple polygons [1], configurations of the Rubik's cube [6], and satisfying assignments of Boolean formulas [19]. For all of these problems, except for the last one, we can decide efficiently if a transformation between two given configurations exists. However, deciding if there is a transformation of at most a given length is NP-complete. In particular, the flip distance of triangulations of planar point sets is known to be APX-hard [22] and, on the positive side, fixed-parameter tractable (FPT) in the length of the transformation [17]. Our reference problem is the task of computing the length of a shortest transformation between two matchings of a graph.

Reconfiguration of Matchings. A *matching* M of a graph is a set of pairwise independent edges. (Fig. 1 shows the six different matchings of the graph G.) A matching is *inclusion-wise maximal* if it is not properly contained in another matching. We may consider a matching as a placement of (unlabeled) *tokens* on independent edges. Then the *Token Jumping* (*TJ*) operation provides an adjacency relation on the set of matchings of a graph, all having the same cardinality[1]: Two matchings M and M' of a graph G are *adjacent* (under

[1] There is another well-studied operation called Token Sliding (TS). In this paper, we employ TJ as the default operation. However, some of our results apply also to TS.

TJ) if one can be obtained from the other by relocating a single token, that is, if $|M \setminus M'| = 1$ and $|M' \setminus M| = 1$. We say that a sequence $M_0, M_1, \ldots, M_\ell$ of matchings of G is a *reconfiguration sequence* of *length* ℓ from M to M', if $M_0 = M$, $M_\ell = M'$, and the matchings M_{i-1} and M_i are adjacent for each $i \in \{1, 2, \ldots, \ell\}$; see the sequence $M_0, M_1, \ldots, M_4$ in Fig. 1 as an example. The following question is often referred to as the *reachability variant* of the matching reconfiguration problem:

Matching Reconfiguration
Input: Graph G and two matchings M_s, M_t of G.
Question: Is there a reconfiguration sequence from M_s to M_t?

Matching Reconfiguration is known as an early example of a reconfiguration problem that admits a non-trivial polynomial-time algorithm [15]. For Yes-instances, the algorithm from [15] gives a bound of $O(n^2)$ on the length of a transformation. The *distance* between two matchings is the length of a shortest transformation between them (under TJ). If there is no transformation between two matchings, we regard their distance as infinity. In this paper we study the complexity of the following optimization problem related to matching reconfiguration, which is also referred to as the *shortest variant*.

Matching Distance
Input: Graph G and two matchings M_s, M_t of G.
Task: Compute the distance between M_s and M_t.

We also study two related *exact* problems. The first is the exact version of Matching Distance, which takes as input also the supposed distance ℓ of the given matchings.

Exact Matching Distance
Input: Graph G, matchings M_s, M_t of G, and number $\ell \in \mathbb{N}$.
Question: Is ℓ equal to the distance between M_s and M_t?

We can similarly define Exact Matching Diameter: Given a graph G and numbers $k, \ell \in \mathbb{N}$, decide if the maximum distance between any two matchings of G of cardinality k is equal to ℓ.

Related Results. Despite recent intensive studies on reconfiguration problems (see, e.g., [20]), most known algorithmic (positive) results are obtained for reachability variants. However, such algorithms sometimes also give answers to the corresponding shortest variants: if the algorithm constructs an actual reconfiguration sequence which, at any step, transforms an edge of the initial matching into an edge of the target one, then the transformation must indeed be a shortest one.

Generally speaking, finding shortest transformations is more difficult when we need a *detour*, a transformation that touches an element that is not in the symmetric difference of the source and target configurations. For such a detour-required case, only a few polynomial-time algorithms are known for shortest variants, e.g., satisfying assignments of a certain Boolean formulas [19], and independent sets under the TS operation for caterpillars [24]. Note that Matching Reconfiguration belongs to the detour-required case (in the example of Fig. 1, we need to use the edge in $E(G) \setminus (M_s \cup M_t)$ in any reconfiguration sequence).

The reconfiguration of matchings is a special case of the reconfiguration of independent sets of a graph. To see this, recall that matchings of a graph correspond to independent sets of its line graph. Therefore, by a result of Kamiński et al. [16], we can solve Matching Distance in polynomial time if the line graph of a given graph is even-hole-free. Note that in this case no detour is required.

Our Results. Although the reconfiguration of independent sets is one of the most well-studied reconfiguration problems (see, e.g., a survey [20]), to the best of our knowledge, the shortest variant of independent sets under the TJ operation is known to be solvable only for even-hole-free graphs, as mentioned above. Thus, in this paper, we start a systematic study of the complexity of finding shortest reconfiguration sequences between matchings, and more generally, between independent sets of a graph.

Our first result shows that there is no polynomial-time algorithm that computes a sublogarithmic-factor approximation of the distance between two matchings unless $\mathsf{P} = \mathsf{NP}$. The result implies approximation hardness for the length of shortest transformations between b-matchings of a graph and shortest transformations between independent sets on any graph class containing line graphs of bipartite graphs. Note that maximal subclasses of line graphs of bipartite graphs that have been considered in the literature are either trivial (e.g., disjoint unions of cliques) or even-hole-free, so our result implies a sharp complexity bound.

Theorem 1. Matching Distance *admits no polynomial-time $o(\log n)$-factor approximation algorithm, unless* $\mathsf{P} = \mathsf{NP}$*, even for maximal matchings of bipartite graphs of maximum degree three.*

The proof of Theorem 1 is provided in Sect. 2. Our main result is positive. We show that determining the distance between matchings of *bipartite* graphs is FPT, where the parameter is the size d of the symmetric difference of the two input matchings. An outline of the algorithm is provided in Sect. 3.1. We distinguish two main cases: either a shortest reconfiguration sequence contains a non-inclusion-wise maximal matching or not. For the former case we give a polynomial time algorithm. Note that this algorithm implies in particular a polynomial-time algorithm for finding a shortest transformation between two matchings if at least one of the two matchings is not inclusion-wise maximal. In order to deal with the latter case, we give a reduction from Matching Distance to the problem Directed Steiner Tree, where the number of terminals is of the order of the size of the symmetric difference of the source and target matchings. By putting everything together we obtain our main result.

Theorem 2. Matching Distance *in bipartite graphs can be solved in time $2^d \cdot n^{O(1)}$, where d is the size of the symmetric difference of the two given matchings.*

Theorem 2 raises hopes for possible generalizations, e.g., an FPT algorithm for finding shortest transformations between matchings in general graphs or between independent sets of claw-free graphs. For maximum matchings, our reduction to Directed Steiner Tree is approximation-preserving, which implies the following.

Corollary 1. Matching Distance *restricted to maximum matchings in bipartite graphs admits a polynomial-time d^{ε}-factor approximation algorithm for every $\varepsilon > 0$, where d is the size of the symmetric difference of two given matchings.*

The proofs of Theorem 2 and Corollary 1 are given in Sect. 3.

We finally show that there is a polynomial-time algorithm that decides if the maximum distance between any two matchings of a graph is *finite*. In contrast, we also show that the problems Exact Matching Distance and Exact Matching Diameter are both hard for the class $\mathsf{D^P}$, a class containing both NP and coNP.

Theorem 3. *The problem* Exact Matching Distance *is* $\mathsf{D^P}$*-complete and the problem* Exact Matching Diameter *is* $\mathsf{D^P}$*-hard.*

The class $\mathsf{D^P}$ has been introduced by Papadimitriou and Yannakakis in [21] and is a natural complexity class for *exact* problems and *critical* problems. It was proved by Frieze and Teng that the related problem of deciding the diameter of the graph of a polyhedron is also $\mathsf{D^P}$-hard [11]. Section 4 is devoted to the proof of Theorem 3. Due to space restriction, proofs of statements marked by $(*)$ are not included in this extended abstract. We would like to remark that the NP-hardness of Matching Distance has been established independently in [13].

Notation. Let $G = (V, E)$ be a graph. Unless stated otherwise, graphs are simple. For standard definitions and notation related to graphs, we refer the reader to [7]. We denote by $A \triangle B$, the symmetric difference of two sets A and B. That is, $A \triangle B := (A \setminus B) \cup (B \setminus A)$. Let $M \subseteq E$ be a matching of G. A vertex of G that is not incident to any edge in M is called *M-exposed* (or *M-free*), otherwise it is called *matched* or *covered.*

2 Approximation Hardness of Matching Distance

To prove Theorem 1, we first show the following slightly less general result.

Theorem 4. Matching Distance *admits no $o(\log n)$-factor approximation unless $\mathsf{P} = \mathsf{NP}$, even when restricted to maximum matchings on bipartite graphs of maximum degree three.*

We show that a sublogarithmic-factor approximation for Matching Distance yields a sublogarithmic-factor approximation of Set Cover. Since Set Cover is not approximable within a sublogarithmic factor, unless $\mathsf{P} = \mathsf{NP}$ [8], Theorem 4 follows. Note that Theorem 4 also holds for the TS operation if M_1 and M_2 are maximum[2]. To obtain Theorem 1, we need to slightly alter the reduction used in the proof of Theorem 4 to transform maximum matchings into maximal matchings. (We refer the reader to the full version [4] for the detailed construction).

Let us briefly recall some definitions related to the Set Cover problem. An instance $I = (U, \mathcal{S})$ of Set Cover is given by a set U of *items* and a family $\mathcal{S}$ of subsets of U. The task is to find the minimum number of sets in $\mathcal{S}$ that are required to cover U. We denote this number by $\mathrm{OPT}(I)$ and let $n := |U|$ and $m := |\mathcal{S}|$. Furthermore, let $d := \max_{S\in\mathcal{S}}\{|S|\}$ be the maximum cardinality of a set in $\mathcal{S}$ and, for each $u \in U$, let $f_u := |\{S \in \mathcal{S} \mid u \in S\}|$ be the *frequency* of u, and let $f := \max_{u\in U}\{f_u\}$ be frequency of I.

Let us now give the reduction and the two main lemmas that guarantee the safeness of our construction.

Reduction. We construct from the Set Cover instance $I = (U, \mathcal{S})$ an instance $I' = (G, M_1, M_2)$ of Matching Distance as follows. For each item $u \in U$, we create a cycle C_u of length four and label one of the vertices by c_u. Then, for each set $S \in \mathcal{S}$, we add a path P_S of length $L := |U|(2 + f + d)$ and label the end-points p_S and q_S. We may assume without loss of generality that L is even. Now, for each set $S \in \mathcal{S}$ and item $u \in U$, we join p_S to c_u by an edge if and only if $u \in S$. The two matchings M_1 and M_2 are constructed as follows. For each $S \in \mathcal{S}$, we leave q_S exposed and add every second other edge to both matchings. For each $u \in U$ there are two different perfect matchings of C_u and we add one to M_1 and the other to M_2. Note that the graph G is bipartite and, since L is even, only the vertices q_S are M_1- and M_2-exposed for $S \in \mathcal{S}$. In order to get the maximum degree down to three, we have to use a slightly more elaborate construction for the part of G that corresponds to the incidence graph of the Set Cover instance. This completes the construction of the instance I' of Matching Distance.

Observe that the construction of I' performed in polynomial time. In order to change the matching on a cycle C_u, it is necessary to move each token on some path P_S such that $u \in S$. Intuitively, we think of L as a very large number, so in order to change the matching M_1 on each cycle C_u, $u \in U$, it is desirable to minimize the number of times we have to move the tokens on the long paths P_S, $S \in \mathcal{S}$. In order to obtain the approximation hardness result in Theorem 4, we establish the following correspondences between reconfiguration sequences from M_1 to M_2 and covers of U by sets in $\mathcal{S}$:

Lemma 1 (∗). *Let $C \subseteq \mathcal{S}$ be a cover of U. Then there is a reconfiguration sequence from M_1 to M_2 of length at most $2L|C| + 2|U|(2 + f + d)$.*

[2] If we delete an edge e of a maximum matching, we can only replace it by an edge e' sharing an endpoint with e. So for maximum matchings, TJ and TS rules are equivalent.

Lemma 2 (∗). *There is a polynomial-time algorithm A' that constructs from a reconfiguration sequence τ from M_1 to M_2 of length $|\tau|$ a cover $C \subseteq \mathcal{S}$ of U of cardinality at most $|\tau|/2L$.*

Combining Lemmas 1 and 2, we have that a $o(\log |I'|)$-factor approximation algorithm for Matching Distance yields a $o(\log n)$-factor approximation algorithm for Set Cover, which contradicts the approximation hardness result from [8].

3 Matching Distance in Bipartite Graphs is FPT

The goal of this section is Theorem 2, which states that the distance of two matchings of a bipartite graph is FPT, where the parameter is the size d of the symmetric difference of the source and target matchings. Let us fix an instance (G, M_s, M_t) of Matching Distance and let us assume that the graph $G = (V, E)$ is bipartite with bipartition $V = (U, W)$. According to [15, Proposition 1] we may check in polynomial time whether a transformation from M_s to M_t exists, so let us assume in the following such transformation exists. In the remainder of this section we denote by $\mathcal{C}$ (resp., $\mathcal{P}$) be the set of (M_s, M_t)-alternating cycles (resp., (M_s, M_t)-alternating paths) in $(V, M_s \triangle M_t)$.

3.1 Overview of the Algorithm

There are two distinct main cases we need to consider.

Case 1 (A shortest transformation from M_s to M_t visits a matching that is not inclusion-wise maximal).
We show that in this case we can find a shortest transformation from M_s to M_t in polynomial time, which may seem quite surprising in the light of the hardness result given in Theorem 1. We first observe that the following holds:

Lemma 3 (∗). *A shortest transformation between two matchings can be computed in polynomial time if at least one of them is not inclusion-wise maximal.*

To prove Lemma 3, we show that the distance of two matchings M_s and M_t, at least one of which is not inclusion-wise maximal, is either $|M_s \triangle M_t|/2$ or $|M_s \triangle M_t|/2 + 1$. We can check in polynomial time which case applies. Note that Lemma 3 also holds if we do not assume that the input graph is bipartite. The hard part of Case 1 is to prove the following lemma (which only holds for bipartite graphs):

Lemma 4 (∗). *There is a polynomial-time algorithm that outputs a shortest transformation from M_s to M_t via a matching M that is not inclusion-wise maximal, or indicates that no such transformation exists.*

Let us briefly summarize the algorithm. Since a shortest transformation from M_s to M_t visits a non-inclusion-wise maximal matching, we have that M_s and M_t cannot be maximum, so there is at least one M_s-augmenting path. Note that, given an M_s-augmenting path P, we may transform M_s into a non-inclusion-wise maximal matching by sliding tokens along P. We may then find a shortest transformation from the resulting matching to M_t in polynomial time according to Lemma 3. We show that it suffices to find an M_s-augmenting path that gives an overall shortest transformation. This task reduces to a shortest-path-computation in a suitable weighted digraph.

Case 2 (No shortest transformation from M_s to *M_t visits a matching that is not inclusion-wise maximal).*
Let us first assume that M_s and M_t are maximum. We reduce the task of finding a shortest transformation from M_s to M_t to the problem Directed Steiner Tree, which is defined as follows.

Directed Steiner Tree
Input: Directed graph $D = (V, A)$, integral arc weights $c \in \mathbb{Z}^A_{\geq 0}$, root vertex $r \in V$, and terminals $T \subseteq V$.
Task: Find a minimum-cost directed tree in D that connects the root r to each terminal.

The reduction to Directed Steiner Tree and its correctness are sketched in Sect. 3.2. The main idea is to construct an instance of Directed Steiner Tree such that each arc of positive cost corresponds to a token move and its cost corresponds to how many times the token has to be moved in a transformation from M_s to M_t. It is known that Directed Steiner Tree parameterized by the number of terminals is FPT [2,9]. Our reduction gives at most $d/2$ terminals. We employ the FPT algorithm from [2] to obtain the following result.

Lemma 5. *Let M_s and M_t be maximum. Then there is an algorithm that finds in time $2^{d/2} \cdot n^{O(1)}$ a shortest transformation from M_s to M_t, or indicates that no such transformation exists.*

In order to deal with the case that M_s and M_t are not maximum, we first recall the following lemma from [15].

Lemma 6 **(**[15, Lemma 1]**).** *If M_s and M_t are maximum then there is a transformation from M_s to M_t if and only if, for each cycle $C \in \mathcal{C}$, there is an M_s-alternating path in G connecting an M_s-exposed vertex to C.*

Note that we assume that for each shortest transformation from M_s to M_t, each intermediate matching is inclusion-wise maximal. Therefore, in a shortest transformation from M_s to M_t, we *have* to use the algorithm from the constructive proof of Lemma 6 to transform the cycles in $\mathcal{C}$. Hence, in such a transformation, we have to consider for each cycle $C \in \mathcal{C}$ the two choices that C is reconfigured using either an M_s-exposed vertex from U or from W. Since each cycle has length at least four, we have that $\mathcal{C}$ contains at most $d/4$ cycles.

We branch over all of the at most $2^{d/4}$ possible choices. For each choice, we reduce the problem to the case where M_s and M_t are maximum as follows. We create two sub-instances: one for the cycles $\mathcal{C}_1$ that have to be reconfigured using exposed vertices in U and one for the cycles $\mathcal{C}_2$ that have to be reconfigured using exposed vertices in W. For the sub-instance of cycles in $\mathcal{C}_1$, we delete all the exposed vertices in W and the matching M_s then becomes maximum. (We perform a similar reduction in the other case). We finally show that no transformation maintaining maximal matchings all along is better than combining the optimal solutions of the two sub-instances and obtain the following result.

Lemma 7 (∗). *Suppose no shortest transformation from M_s to M_t visits a matching that is not inclusion-wise maximal. Then there is an algorithm that finds in time $2^d \cdot n^{O(1)}$ a shortest transformation from M_s to M_t.*

Hence, Theorem 2 follows from Lemmas 4 and 7. Note that if M_s and M_t are maximum, then we may use the approximation algorithm for Directed Steiner Tree from [5] instead of the exact algorithm from [2]. Since our reduction to Directed Steiner Tree preserves costs, we obtain from an α-approximate solution of the Directed Steiner Tree instance an α-approximate solution to Matching Distance, which implies Corollary 1. Our techniques are not likely to generalize to matchings in non-bipartite graphs in a straight-forward way. We leave as an open problem whether finding a shortest transformation between two matchings in non-bipartite graphs is FPT in the size of the symmetric difference of source and target matchings.

3.2 Proof of Lemma 5: Reduction to Directed Steiner Tree

Let M_s and M_t be maximum matchings of G. We will reduce the task of finding a shortest transformation from M_s to M_t to the Directed Steiner Tree problem. Note that if some edge e is not contained in any maximum matching of G, then we cannot move any token to e and e can be deleted from the graph. Therefore, we may assume that every edge of G is contained in some maximum matching. Let X_s be the set of M_s-exposed vertices of G. By the properties of the Edmonds-Gallai decomposition [23, Ch. 24.4b], we may assume the following[3].

Proposition 1 (∗). *Without loss of generality we have $X_s \subseteq U$.*

Reduction. The main feature of the reduction is that it preserves costs. That is, an optimal Steiner tree of cost α corresponds to a shortest transformation from M_s to M_t of length at most α. We construct an instance $I' := (D, c, r, T)$ of Directed Steiner Tree as follows. The digraph $D = (U', A)$ is given by

$$\begin{aligned} U' &:= \{v \in U \mid \exists \text{ an even-length } M_s\text{-alternating path from } X_s \text{ to } v\} \cup \{r\} \\ A &:= \{uw \mid u, w \in U,\ \exists v \in W :\ uv \in E \setminus M_s,\ vw \in M_s\} \cup R, \end{aligned}$$

[3] Due to space restrictions, the definition of Edmonds-Gallai decomposition is not included in this extended abstract, see [4] for more details.

where r is a new vertex and $R := \{rv \mid v \in X_s\}$. For an arc $uw \in A$, let the weight c_{uw} be given by

$$c_{uw} := \begin{cases} 0 & \text{if } u = r, \\ 1 & \text{if there are two edges } uv \in M_t \text{ and } vw \text{ in } M_s. \\ 2 & \text{otherwise.} \end{cases}$$

The set T of terminals is given by $T := U' \cap \bigcup_{Z \in \mathcal{C} \cup \mathcal{P}} V(Z)$. Note that any two distinct items in $\mathcal{P} \cup \mathcal{C}$ are vertex-disjoint. The root of the Steiner tree is the vertex r. This completes the construction of the instance I'.

Let us now give an outline of the proof of Lemma 5, which states that finding a shortest transformation between two maximum matchings M_s and M_t of a bipartite graph is fixed parameter tractable, where the parameter is the size d of the symmetric difference of M_s and M_t.

We first observe that we may restrict our attention to Steiner trees with some structure on the paths $\mathcal{P}$ and cycles $\mathcal{C}$ of $(V, M_s \triangle M_t)$.

Proposition 2 ($*$). *Let F be a Steiner tree for I'. Then we can obtain in polynomial time a Steiner tree F' for I' of cost at most $c(F)$ with the following properties.*

(i) For each $P \in \mathcal{P}$, the tree F' contains all arcs in $A(P)$.
(ii) For each $C \in \mathcal{C}$, the tree F' misses exactly one arc of $A(C)$.
(iii) For each $P \in \mathcal{P}$, the root r is joined to the M_s-exposed vertex of P.

The proof of next lemma shows how to construct from a Steiner tree F' of cost $c(F')$ a reconfiguration sequence of length at most $c(F')$.

Lemma 8 ($*$). *Let F' be a Steiner tree for I'. Then we can construct in polynomial time a transformation from M_s to M_t of length at most $c(F')$.*

Let us sketch the proof of Lemma 8. If F' does not satisfy the properties of Proposition 2, then we may find in polynomial time a Steiner tree F for I' of cost at most $c(F')$ that does. We perform a DFS traversal of F giving a preference to visiting arcs with largest weight. Note that we visit each arc of F twice, once going "down" the tree and once going "up". Each arc of weight at least one corresponds to a token and the arc-weight specifies how often the corresponding token is moved during the traversal of F. When we traverse an arc of weight one going down the tree, then we move a token to its target destination, so we perform no token move when backtracking. On the other hand, an arc of weight two corresponds to moving a token away from its target position, so we have to undo the move when backtracking. The placement of the terminals on the items in $\mathcal{C} \cup \mathcal{P}$ ensures that after the traversal of F, all tokens have been moved to their target positions.

From Lemma 8 and the next lemma we may conclude that the shortest length of a transformation from M_s to M_t equals the optimal cost of a Steiner tree for I'.

Lemma 9 (∗). *Let $M_0, M_1, \dots, M_m$ be a transformation of length m from M_s to M_t. Then there is a Steiner tree F of I' such that $c(F) \leq m$.*

We combine our previous arguments to prove Lemma 5. The construction of I' can be performed in polynomial time. Moreover, the number of terminals of I' is at most $d/2$. So the Directed Steiner Tree algorithm from [2] computes an optimal Steiner tree F^* of I' in time $2^{d/2} \cdot n^{O(1)}$. Lemma 8 ensures that F^* can be turned into an transformation between M_s and M_t in polynomial time. Lemma 9 ensures that this transformation is of the shortest length. Finally, since the construction is polynomial, the approximation algorithm for Directed Steiner Tree from [5] yields Corollary 1.

4 Exact Distance and Diameter

We consider the problems Exact Matching Distance and Exact Matching Diameter. Before presenting our hardness results for these problems, we first prove that we can decide in polynomial if the maximal distance of any two matchings of a graph is finite. It will be convenient to consider the *reconfiguration graph* $\mathcal{M}_k(G)$ of matchings of a graph G, which is given as follows.

$$V(\mathcal{M}_k(G)) := \{M \subseteq E \mid M \text{ is a matching in } G, |M| = k\}$$
$$E(\mathcal{M}_k(G)) := \{MN \mid M, N \in V(\mathcal{M}_k(G)),\ |M \triangle N| = 2\}$$

We show that for $k \geq 0$ we can decide in polynomial time if $\mathcal{M}_k(G)$ is connected. First suppose that k is less than the size of a maximum matching of G. Then we can transform any matching of size k into one that is not inclusion-wise maximal by sliding tokens along an augmenting path. Hence, by the algorithm given in the proof of Lemma 3, the graph $\mathcal{M}_k(G)$ is connected. Now suppose that k is equal to the size of a maximum matching of G and consider the *Edmonds-Gallai decomposition* A, D, C of the vertex set of G, where C are the vertices that are covered by any maximum matching [23, Ch. 24.4b]. By showing that $\mathcal{M}_k(G)$ is connected if and only if the graph $G[C]$ has a unique perfect matching we obtain the following result.

Theorem 5 (∗). *There is a polynomial-time algorithm that, given a graph G and a number $k \in \mathbb{N}$, decides if $\mathcal{M}_k(G)$ is connected.*

To obtain hardness results for Exact Matching Distance and Exact Matching Diameter it suffices to consider maximum matchings. By using a similar construction to the one from Sect. 2 we show that Exact Matching Distance and Exact Matching Diameter are hard for the class D^P, which is given by $\mathsf{D}^\mathsf{P} := \{L_1 \cap L_2 \mid L_1 \in \mathsf{NP},\ L_2 \in \mathsf{coNP}\}$. We reduce from the problem Exact Vertex Cover, which asks whether the minimum size of a vertex cover of a graph is equal to a given number ℓ. Our construction guarantees that, if we can decide the size of a shortest transformation, then we can decide the size of a minimum vertex cover. Since Exact Matching Distance is in D^P (the question "is the distance of two matchings in $\mathcal{M}_k(G)$ at most ℓ" being in NP), Theorem 3 follows.

References

1. Aichholzer, O., Mulzer, W., Pilz, A.: Flip distance between triangulations of a simple polygon is NP-complete. Discret. Comput. Geom. **54**(2), 368–389 (2015). https://doi.org/10.1007/s00454-015-9709-7
2. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Fourier meets Möbius: fast subset convolution. In: Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing, pp. 67–74. ACM (2007). https://doi.org/10.1145/1250790.1250801
3. Bonamy, M., Bousquet, N.: Recoloring graphs via tree decompositions. Eur. J. Comb. **69**, 200–213 (2018). https://doi.org/10.1016/j.ejc.2017.10.010
4. Bousquet, N., Hatanaka, T., Ito, T., Mühlenthaler, M.: Shortest reconfiguration of matchings. CoRR abs/1812.05419 (2018)
5. Charikar, M., et al.: Approximation algorithms for directed Steiner problems. J. Algorithms **33**(1), 73–91 (1999). https://doi.org/10.1006/jagm.1999.1042
6. Demaine, E.D., Eisenstat, S., Rudoy, M.: Solving the Rubik's cube optimally is NP-complete. In: 35th Symposium on Theoretical Aspects of Computer Science. STACS, vol. 96, pp. 24:1–24:13 (2018). https://doi.org/10.4230/LIPIcs.STACS.2018.24
7. Diestel, R.: Graph Theory, Graduate Texts in Mathematics, vol. 173, 3rd edn. Springer, Heidelberg (2005)
8. Dinur, I., Steurer, D.: Analytical approach to parallel repetition. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing, pp. 624–633. ACM, New York (2014). https://doi.org/10.1145/2591796.2591884
9. Dreyfus, S.E., Wagner, R.A.: The Steiner problem in graphs. Networks **1**(3), 195–207 (1971). https://doi.org/10.1002/net.3230010302
10. Feghali, C., Johnson, M., Paulusma, D.: A reconfigurations analogue of Brooks' theorem and its consequences. J. Graph Theory **83**(4), 340–358 (2016)
11. Frieze, A.M., Teng, S.H.: On the complexity of computing the diameter of a polytope. Comput. Complex. **4**(3), 207–219 (1994). https://doi.org/10.1007/BF01206636
12. Gopalan, P., Kolaitis, P.G., Maneva, E.N., Papadimitriou, C.H.: The connectivity of Boolean satisfiability: computational and structural dichotomies. SIAM J. Comput. 2330–2355 (2009). https://doi.org/10.1137/07070440X
13. Gupta, M., Kumar, H., Misra, N.: On the complexity of optimal matching reconfiguration. In: Catania, B., Královič, R., Nawrocki, J., Pighizzini, G. (eds.) SOFSEM 2019. LNCS, vol. 11376, pp. 221–233. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-10801-4_18
14. van den Heuvel, J.: The complexity of change. In: Surveys in Combinatorics 2013, pp. 127–160. Cambridge University Press (2013)
15. Ito, T., et al.: On the complexity of reconfiguration problems. Theor. Comput. Sci. **412**(12–14), 1054–1065 (2011). https://doi.org/10.1016/j.tcs.2010.12.005
16. Kamiński, M., Medvedev, P., Milanič, M.: Complexity of independent set reconfigurability problems. Theor. Comput. Sci. **439**, 9–15 (2012). https://doi.org/10.1016/j.tcs.2012.03.004
17. Li, S., Feng, Q., Meng, X., Wang, J.: An improved FPT algorithm for the flip distance problem. In: 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017), vol. 83, pp. 65:1–65:13 (2017). https://doi.org/10.4230/LIPIcs.MFCS.2017.65

18. Lokshtanov, D., Mouawad, A.E.: The complexity of independent set reconfiguration on bipartite graphs. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 185–195. SIAM (2018)
19. Mouawad, A.E., Nishimura, N., Pathak, V., Raman, V.: Shortest reconfiguration paths in the solution space of Boolean formulas. SIAM J. Discret. Math. **31**(3), 2185–2200 (2017). https://doi.org/10.1137/16M1065288
20. Nishimura, N.: Introduction to reconfiguration. Algorithms **11**(4:52) (2018). https://doi.org/10.3390/a11040052
21. Papadimitriou, C.H., Yannakakis, M.: The complexity of facets (and some facets of complexity). J. Comput. Syst. Sci. **28**(2), 244–259 (1984). https://doi.org/10.1016/0022-0000(84)90068-0
22. Pilz, A.: Flip distance between triangulations of a planar point set is APX-hard. Comput. Geom. **47**(5), 589–604 (2014). https://doi.org/10.1016/j.comgeo.2014.01.001
23. Schrijver, A.: Combinatorial Optimization - Polyhedra and Efficiency, Algorithms and Combinatorics, vol. 24. Springer, Heidelberg (2003)
24. Yamada, T., Uehara, R.: Shortest reconfiguration of sliding tokens on a caterpillar. In: Kaykobad, M., Petreschi, R. (eds.) WALCOM 2016. LNCS, vol. 9627, pp. 236–248. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30139-6_19